Elaboration d'un Système hybride Neuro-Génétique Pour le Diagnostic Médical

D.Yedjour, A.Benyettou, H.Yedjour*

Université des sciences et de la Technologie d'Oran
Faculté des sciences
Département d'Informatique
Email: dyedjour@yahoo.fr

Résumé: Les réseaux de neurones artificiels sont toujours considérés comme des boites noires, qui permettent après un apprentissage à partir d'une base d'exemples incomplète, de classifier de nouveaux exemples, mais sans donner aucune explication sur les résultats. Leurs connaissances sont codées de manière interne par les poids synaptiques, et ne sont pas donc exprimées de manière compréhensible. Les algorithmes génétiques très performants dans les problèmes d'exploration semblent être en mesure de rechercher dans l'espace des ensembles de règles, celui qui représentera le mieux les connaissances d'un RNA. En revanche, ils sont inefficaces lorsqu'il s'agit de trouver la valeur exacte de l'optimum dans cet espace or, c'est précisément ce que les algorithmes exacts d'optimisation réalisent le mieux. Nous présentons dans cet article une nouvelle approche d'extraction de règles à partir d'un réseau de neurones permettant de combiner les deux méthodes métaheuristiques et exactes au sein d'un même système.

Mots-clés: Réseaux de neurones, Algorithme génétique, Extraction de règles, Méthode de Quine Mc-Cluskey

1. Introduction

Les réseaux de neurones sont devenus un outil de plus en plus utilisé (industrie, parole, diagnostic médicale, finance, traitement de signal et dans beaucoup d'autres problèmes), grâce à leur capacité d'apprendre et de généraliser, De plus ils sont peu sensibles aux données approximatives et à la présence de données incorrectes dans la base d'exemple utilisée lors de leur apprentissage. En revanche, les connaissances acquises lors de l'apprentissage sont stockées par le RNA dans sa topologie et les poids de ses connexions, ce qui empêche toute justification des réponses du réseau. Nous pouvons dire que le réseau est une sorte de « boite noire » [10]. Cependant l'extraction de règles pertinentes demeure importante si les résultats du réseau de neurones sont utilisés comme théorie initiale dans des problèmes similaires [2]. Un system combinant le modèle connexionniste et le raisonnement symbolique est dit système hybride intelligent [5]. Il existe deux approches dans l'explicitation des connaissances d'un RNA. l'approche décompositionnelle qui tente d'analyser la topologie et les poids des connexions d'un réseau afin d'en déduire des règles, on cite comme exemple RuleNet[9], RULEX[1], Subset [3], Full–RE [12] , et l'approche pédagogique qui consiste non plus à s'intéresser aux unités d'un réseau, mais simplement à analyser ses réponses par rapport aux entrées, on peut citer quelques travaux qui ont utilisé cette approche VIA(Validity Internal analysis)[13], GEX(crisp Rule EXtraction) and REX (fuzzy Rule EXtraction) [6]. MulGEX [7], CGA[8], BIO-RE (Binary Input-Output Rule Extraction) [12]. Dans cet article, on décrit une nouvelle méthode qui combine les algorithmes métaheuristiques (algorithmes génétiques) avec les méthodes exactes (Quine-Mc-cluskey) afin d'extraire à partir d'un réseau de neurones, les règles binaires de la forme if-then. Les algorithmes génétiques très performants dans les problèmes d'exploration semblent être en mesure de rechercher dans l'espace des ensembles de règles, celui qui représentera le mieux les connaissances d'un RNA. En revanche, ils sont inefficaces lorsqu'il s'agit de trouver la valeur exacte de l'optimum dans cet espace. Or, c'est précisément ce que les algorithmes exacts d'optimisation réalisent le mieux. Il est donc naturel de penser à associer un algorithme exact à l'algorithme génétique de façon à trouver la valeur exacte de l'optimum. On peut aisément le faire en appliquant à la fin de l'algorithme génétique un algorithme exact sur le meilleur élément trouvé. Notre système est testé sur la base de données cancer du sein de l'université de Californie. Les expériences montrent que notre système donne de bons résultats.

2. SYSTEME MC-RULEGEN

La figure1 présente l'architecture de notre système MC-RULEGEN, il est décomposé en 04 modules: le module perceptron multicouches, le module génétique, le module simplification de règles, et enfin le module system à base de règles.

2.1. Module Apprentissage du réseau de neurones

Les données doivent être dans un format binaire, sinon une procédure de binarisation (1) sera appliqué sur les données non-binaires [12].

$$y_i = \begin{cases} 1 & si \;\; x_i \geq u_i \\ 0 & \sin on \end{cases} \qquad (1)$$

où xi est la valeur de l'attribut Xi, ui est la valeur moyenne de Xi et yi est la valeur binaire correspondante.

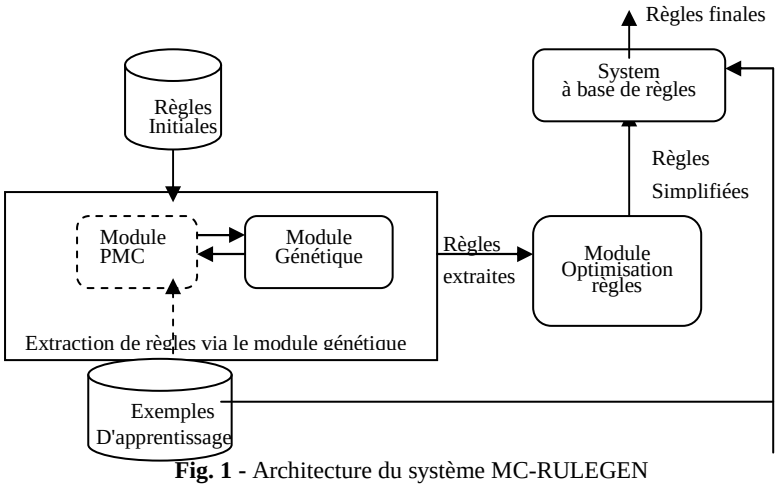

Fig. 1 - Architecture du système MC-RULEGEN

Le PMC est appris à partir d'une base d'exemple, chaque vecteur d'entrée, est associe à un vecteur de sortie (apprentissage supervisé), nous avons utilisé l'algorithme de la rétro-propagation. L'apprentissage sert à déterminer les valeurs optimales des poids (la matrice des poids), les connaissances du réseau sont contenues dans cette matrice. La phase de l'apprentissage nécessite la manipulation de plusieurs paramètres (momentum, fonction d'activation, fréquence d'apprentissage,..) afin d'aboutir au résultat voulu.

2.2. Module génétique

Les connaissances du réseau de neurones sont difficilement interprétables par un être humain. Pour remédier à cela, il convient d'expliciter ces connaissances, c'est-à-dire les traduire sous une forme intelligible, une approche pédagogique basé sur les algorithmes génétiques est utilisée. La règle extraite doit avoir la forme suivante

`if [not]`x_1 `and [not]`x_2 `. . . then C`

[.] est facultatif

Mesure de qualités des règles extraites

Les règles extraites doivent être précises et compréhensibles [4], [6]. La précision (2) mesure la proportion des exemples correctement classés par la règle parmi tous les exemples d'apprentissage

$$précision = \frac{\text{nombre des exemples correcteme nt classés}}{\text{nombre total des exemples}} \qquad (2)$$

La fidélité se calcule de la manière suivante: chaque individu est passé dans le réseau de neurone pour classification, le pourcentage des bonnes réponses est la valeur de la fidélité associé à l'individu. La compréhensibilité calcule le nombre de règles ainsi que le nombre de prémisses dans chaque règle. Enfin la généralisation est définit par (3)

$$\text{Generalisa tion} = 1 - \frac{\text{nombre de règles}}{\text{nombre des exemples}} \qquad (3)$$

Algorithme génétique pour l'extraction de règles

Les algorithmes génétiques (AG) sont des algorithmes d'optimisation s'appuyant sur des techniques dérivées de la génétique et de l'évolution naturelle. ils utilisent la selection, le croisement et la mutation.

Algorithme

- Générer aléatoirement une population initiale P(0),
- Calculer la fonction fitness fi(m) pour chaque individu m de la population P(t),
- Définir des probabilités de selection pour chaque individu m dans P(t),
- Générer la nouvelle population P(t+1) en appliquant les opérateurs génétiques de croisement et de mutation,
- Repéter l'étape 2 jusqu'à ce que le résultat final est le meilleur individu généré durant la recherche ou bien si le nombre maximal de générations soit atteint.

Dans cet article, l'approche génétique est utilisée pour générer les règles symboliques interprétant le résultat du réseau de neurones, c'est pourquoi ces règles doivent être représentées sous forme de chromosomes.

La forme du choromosomes

Le chromosome est composé d'un ensemble de gènes, chaque gène correspond à une règle, cela dit que le chromosome code un ensemble de règles (figure2)

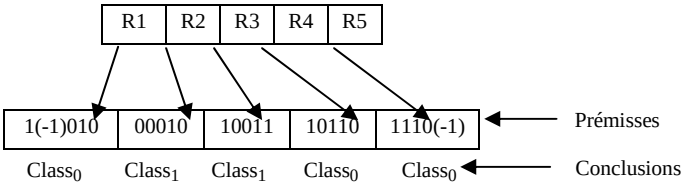

Fig. 2 - Forme du chromosome dans MC-RULEGEN

-1 veux dure que l'attribut n'est pas activé.
0 veux dire que l'attribut x s'ecrit not (x) dans la règle générée
1 veux dire que l'attribut x s'ecrit (x) dans la règle générée
on suppose que les attributs se lisent de la gauche vers la droite alors le dernier gène devient :
if x_1 and x_2 and x_3 and not(x_4) then $class_0$, x_1, x_2 et x3 sont dits attributs positifs et x_4 un attribut négatif, x_5 un attribut inactif

Population initiale

La population initiale de règles est choisie à partir de la table de vérité, qui doit contenir toutes les combinaisons possibles de valeurs d'entrées (attributs), les valeurs de sorties sont générées aléatoirement.

Fonction fitness "mesure de performance"

La fonction *fitness* permet d'évaluer les individus (chromosomes), et donc de déterminer la qualité de la solution. Les meilleurs individus sont mutés et croisés pour produire une nouvelle génération. Dans cet article deux mesures de *fitness* sont utilisées: la fidélité et la compréhensibilité.

Opérateurs génétiques

Les trois opérateurs de base utilisés dans les AG sont: la selection, le croisement et la mutation. La méthode de la selection utilisée est celle de la roulette (roulette wheel selection).

- Croisement*:* permet de combiner deux chromosomes (parents) afin de produire un nouveau chromosome (offspring). La figure 3 explique le croisement (| est le point de croisement):

Chromosome 1	11011 | 00100110110
Chromosome 2	11011 | 11000011110
Offspring 1	11011 | 11000011110
Offspring 2	11011 | 00100110110

Fig. 3 - Exemple du croisement

- *Mutation:* le rôle de la mutation est d'apporter «du nouveau» dans les chromosomes manipulées afin que la recherche ne soit pas cloisonnée dans une partie de l'espace exploré. La mutation consiste juste à choisir aléatoirement un caractère d'une chaîne et à le modifier. Dans notre travail, la mutation peut basculer de la valeur 1 à 0/(-1) ou de 0 à 1/(-1) ou -1 à 0/1. (voir figure 4)

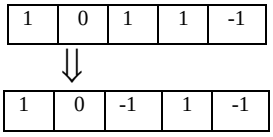

Fig. 4 - Exemple de la mutation

Evaluation de règles

C'est au cours de l'évaluation des règles que vont se réaliser les interactions entre le module génétique et le RNA. nous avons modifié l'algorithme de la rétropropagation (developed by Rumelhart hinton, wiliams [11]), de telle sorte que les attributs inactifs (valeur=-1) soient omis lors du calcul (voir figure5).

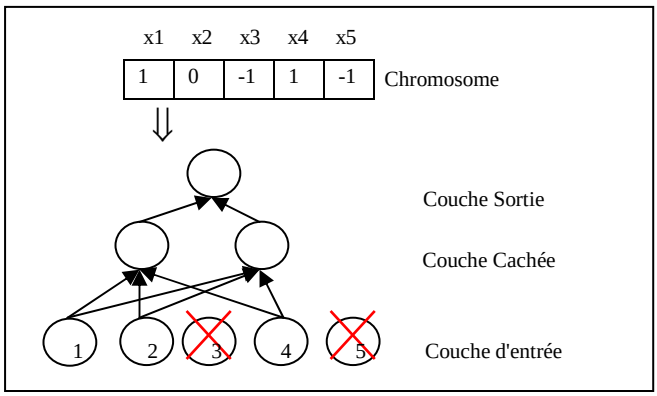

Fig. 5 - Algorithme de la rétropropagation modifié

2.3. Module d'optimisation de règles

L'ensemble de règles générée par AG (supposé optimal) est exprimé sous forme de chaînes de "un", de "zéro" ou de "-1". On applique l'algorithme de Quine-McCluskey sur cet ensemble afin d'arriver à la solution exacte et simplifiée. La méthode de Quine consiste, en partant de la décomposition canonique disjonctive de f, à utiliser systématiquement la formule de simplification $x + \bar{x} = 1$ plusieurs fois jusqu'à ce que aucune paire de termes ne peut être combinées [14].

Considérrons l'exemple suivant :
$f(a, b, c) = a.b + b.\bar{c} + a.\bar{c}$
La décomposition canonique disjonctive de f est :
$f(a, b, c, d) = a.b.c + \bar{a}.b.c + a.b.\bar{c} + ab\bar{c} + a\bar{b}c \mp a\ b\ c$

Algorithme de Quine modifié:

La méthode de quine MC-Cluskey permet de simplifier une fonction en partant de sa forme canonique, nous avons modifié l'algorithme pour qu'il commence avec n'importe quels termes, cela permet de réduire la complexité de l'algorithme.

Comment utiliser l'algorithme de Quine McCluskey dans notre approche:

1. Initialement $R_{NM} = \{\}$, $R_{FIN} = \{\}$
2. Regrouper les règles générées par le module génétique dans des classes, chaque classe englobe tous les règles ayant le même nombre des attributs inactifs (valeur=-1),
3. Numéroter chaque classe, ex: class0 contient les règles avec zéro (0) attribut négatif, class1 contient les règles avec un seul (1) attribut négatif, etc … (si le nombre des sous classes de la classe0=n ➔ nbre maximum de classes (nbrclass)=n)
4. Trier chaque classe suivant le nombre des attributs positifs (valeur=1), construire alors des sous classes, chacune d'elles contient les règles ayant le même nombre de 1, deux sous classes de la classe i sont dites adjacentes si la première contient m attributs positifs et la $2^{ème}$ contient m+1 attributs positifs,
5. Commencer par la classe0, (i←0)
6. Apparier les sous classes adjacentes de la classei deux à deux, appliquer la règle $x + \bar{x}$ (voir figure 6) ,
7. Les règles qui ont participé à la génération des nouvelles règles sont marquées,
8. Les règles non marquées sont insérées dans R_{NM}
9. Les nouvelles règles sont insérées dans la classe i+1,
10. i←i+1
11. si i< n alors aller à l'étape 5 sinon $R_{FIN} = R_{NM}$ fsi
12. Déterminer quels sont les implicants premiers essentiels à partir de R_{FIN} [14].

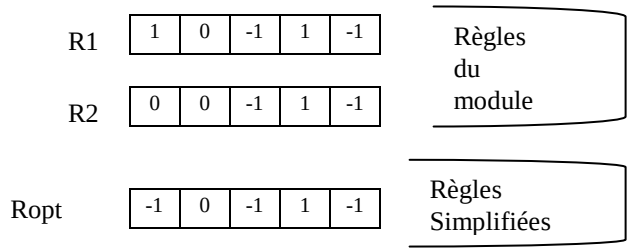

Fig. 6 - Exemple d'application de Quine Mc-cluskey sur les Rè**gles extraites**

Dans cet exemple les règles R1 et R2 sont extraites par le module génétique alors que Ropt est obtenu en appliquant l'algorithme de Quine sur les deux règles

2.4. Système à base de règles

Les règles obtenues à partir des deux modules précédents (génétique et simplification) sont utilisées par le module "système à base règle" dans le but d'obtenir un ensemble final et réduit de règles pertinentes qui couvre le maximum des exemples de test.

Méthodologie:

- On suppose que E= l'ensemble de règles finale, initialement E=∅,
- pour chaque règle, la précision est calculée en comptant le nombre des exemples correctement classés par cette règle,
- les règles sont triées dans l'ordre décroissant, selon leurs valeurs de précision,
- choisir les règles dont la précision est supérieur à la valeur maxaccuracy définie par l'utilisateur,
- si cette règle existe alors:
  - calculer NB le nombre de prémisses dans chaque règle de l'étape 3,
  - Choisir uniquement les règles dont la valeur NB<prem, (prem est une valeur définie par l'utilisateur)
  - Déplacer la règle dans E
- Sinon, l'algorithme fait des combinaisons des deux meilleurs règles au sens de précision, ce processus continu jusqu'à ce qu'on trouve un ensemble de règles vérifiant le critère précision >=maxaccuracy et NB<prem, cet ensemble de règles est déplacé dans E.

3. Résultats Expérimentales

Notre système MC-RULEGEN est testé sur la base de données cancer du sein, nous avons utilisé le modèle PMC avec une seule couche cachée. Les résultats sont comparés avec d'autres approches.

Base de données cancer du sein [15]: Contient 699 exemples répartis sur 02 classes (458 pour bénigne et 241 pour maligne). Chaque exemple est composé de neufs attributs, chacun d'eux prend des valeurs entre 1 et 10. Les exemples sont utilisés dans l'apprentissage et le test.

3.1. Apprentissage du réseau de neurones

Le meilleur résultat est obtenu en utilisant (table 1): 9 neurones dans la couche d'entrée, 03 neurones dans la couche cachée et deux neurones dans la couche de sortie, nous avons utilisé la fonction logistique entre les couches, le momentum = 0.95 et la fréquence d'apprentissage=0.4. Après 500 époques, le taux de classification est égale à 98%.

3.2. Extraction de règles par l'algorithme génétique

*Fonction Fitness:*MC-RULEGEN cherche des règles avec:

- Une meilleure précision,
- Un nombre minimal de prémisses.

Nombre de neurones			Classification (%)	
Entrées	Cachées	Sorties	Training	Test
9	3	2	98.53	98

Table 1. Liste des styles définis dans le présent document.

La taille de la population est initialisée à la valeur *"popsize"*, celle du chromosome (l'individu) est égale "*indiv_length"*, le nombre de gènes (règles) dans le chromosome est égale à *"rulesize"*. Chaque règle contient *"sizeinput"* attributs. La méthode de la roulette est utilisée dans la selection des individus. La prochaine génération est créee à partir de la population courante en utilisant les opérateurs de croisement (avec une probabilité pCross) et de mutation (avec une probabilité pMut). Les meilleurs chromosomes au sens de fitness survivent et participent à la création de la nouvelle population. La population continue à évoluer vers les meilleurs valeurs de fitness. Après plusieurs générations, l'algorithme converge vers le meilleur chromosome. La figure7 montre l'évolution des valeurs de la fonction fitness des meilleurs individus de chaque génération. La compréhensibilité dans chaque individu est calculée en comptant le nombre des attributs inactifs, si ce nombre augumente alors la compréhensibilité augmente. Nous avons utilisé dans notre travail les valeurs suivante: *popsize=30, rulesize*=15, *sizeinput*=9; *indiv_length = sizeinput*rulesize*; pMut=0.2; pCross=0.8; nombre de génération égale à 10000;

3.3. Procédure d'Optimisation

En appliquant la procédure d'optimisation, deux règles sont éliminées.

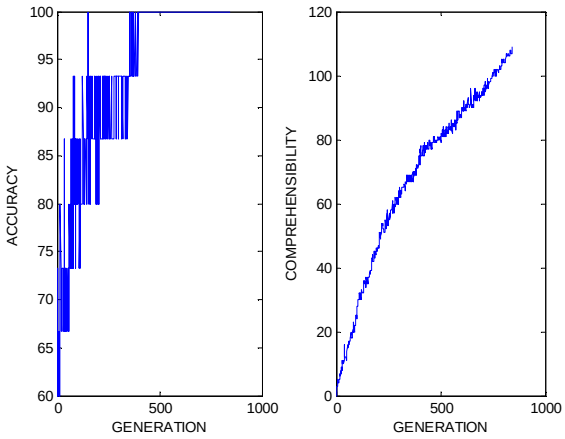

Fig. 7. Evolution de la fonction Fitness

3.4. Système à base de règles

La table 2 montre les 03 meilleurs règles obtenues pour chaque classe. Pour la classe bénigne, en se basant sur la première règle, uniquement 5 parmi 229 d'exemples de test sont malclassées, si on rajoute à cette règle la deuxième, la précision saute à 100% (voir table 3), les autres règles sont alors omises.

Règles Benigne	Précision	Règles Maligne	Précision
R_{B1}	224/229	R_{M1}	96/120
R_{B2}	219/229	R_{M2}	101/120
R_{B3}	200/229	R_{M3}	97/120

Table 2: Nombre des exemples correctement classés par les 03 meilleurs règles

Règles Bégnine	R_{B1}	R_{B2}	$R_{B1}+R_{B2}$
Précision (Test %)	97.82	95.63	100
Précision (Train %)	92.86	95.98	98.66
Nombre prémisses / Nombre de règles	1/1	1/1	2/2

(a)

Règles Malignes	R_{M1}	R_{M2}	R_{M3}	$R_{M1} + R_{M2}$	$R_{M1}+R_{M2}+R_{M3}$
Précision (Test %)	80	84.17	80.83	98.33	100
Précision (Train %)	81.2	82.91	74.36	94.87	97.44
Nombre prémisses / Nombre de règles	1/1	1/1	1/1	2/2	3/3

(b)

Table 3: Qualité des règles extraites (a) pour la classe bénigne et (b) pour la classe maligne)

Les règles finaux obtenues sont données par:
Règles de la classe Bénigne
if (v(8) Normal Nucleoli <2.77) then benin
if (v(6) Bare Nuclei <3.45) then benin

Règles de la classe Maligne
if (v(8) Normal Nucleoli ≥2.77) then malignant
if (v(6 Bare Nuclei ≥3.45) then malignant
if (v(4) Marginal Adhesion ≥ 3) then malignant

Notre base de règles est comparée avec le réseau de neurones (NN), les résultats des table 4 et table 5 montrent que les cinq règles extraites par l'approche MC-RULEGEN permet de couvrir tous les exemples de test, le nombre maximal d'attributs dans chaque règle est égale à "1". Notre système utilise seulement 03 attributs v(8), v(6) et v(4) qui sont suffisants pour représenter toutes les connaissances du NN.

	Bénigne	**Maligne**	**Précision (Test)**	**Nombre des attributs (prémisses)**
NN (Exemples)	229	120	98%	9
MC-RULEGEN (Règles)	2	3	100%	1

Table 4: Comparaison des résultats du NN et de la base de règles

La précision est passée de 98% à 100% en utilisant seulement 05 règles et 03 attributs.

	Réseau de neurones		Règles Extraites	
	Ratio	%	Ratio	%
Apprentissage	336/341	98.53	335/341	98.24
Test	342/349	98	349/349	100
Total	678/690	98.26	684/690	99.13

Table 5: Comparaison entre NN et la base de règles

4. Comparaisons avec d'autres travaux

Dans cette section, nous avons comparé notre approche avec d'autres travaux soit qu'ils utilisent l'approche globale (Bio-Re, GEX, CGA) ou locale (Partial-Re, Full-RE, NeuroRule, C4.5rules). Les résultats de classification des règles obtenues, le nombre de règles trouvé ainsi que le nombre maximale de prémisses par règle sont illustrés dans la table 6. les résultats montrent que les règles extraites par Mc-RULEGEN sont plus performantes et plus compréhensibles par rapport à celles extraites par les autres techniques. MC-RULEGEN est capable d'extraire un ensemble de règles d'une meilleur performance.

	Précision	**Nombre de règles**	**Nombre maximale de prémisses par règle**
MC-RULEGEN	100	05	01
Bio-Re	96.63	10+default rule	04
Partial-Re	96.49	09	3
Full-RE	96.19	05	2
NeuroRule	97.21	3+ default rule	04
C4.5rules	97.21	7	04
GEX	97,81	19,64 ± 2,33	-
CGA	98,3±0,5	18±1,5	2,5±0,1

Table 6: performances de l'ensembles de règles extraites par les différentes techniques

(Dans CGA et GEX, valeur moyenne ± deviation standard est utilisé)

5. Conclusion

Ce papier présente une nouvelle approche d'extraction de règles à partir d'un réseau de neurones. Notre approche combine les deux approches métaheuristiques (algorithme génétique) et les exactes (Quine Mc-cluskey) au sein d'un même système afin d'extraire les règles binaires de la forme if-then, les règles obtenues sont passées dans un système à base de règles pour raffinage. Les résultats expérimentaux montrent que notre approche MC-RULEGEN génère des règles de très haute performance.

REFERENCES

1. Andrews R, and Geva S, Rule Extraction from a Constrained Error Back-Propagation MLP. In: Proceedings of the 6th Australian Conference on Neural Networks, p.9-12, Brisbane Queensland, 1994.

2. Geoffrey G. Towell, Jude W. Shavlik,, "Extracting Refined Rules From Knowledge-Based Neural Networks", Machine Learning (Vol. 13, N°1) (PP. 71-101), 1993.

3. Geoffrey G. Towell, "Symbolic Knowledge and Neural Networks: Insertion, Refinement and Extraction", Ph.D. Thesis, Computer Sciences Departement, University of Wisconsin, Madison, 1991.

4. Huysmans J, Baesens B, and Vanthienen J, “Using Rule Extraction to Improve the Comprehensibility of Predictive Models,” K.U.Leuven KBI, Research 0612, 2006

5. Li Min Fu, "Knowledge-Based Connectionism for Revising Domain Theories", IEEE Transactions on Systems, Man and Cybernetics, Vol.23, N.1, Janvier/Février 1993

6. Markowska-Kaczmar, Evolutionary approaches to rule extraction from neural networks, studies in computational intelligence (SCI) 82, 117-209, 2008.

7. Markowska-Kaczmar U, Mularczyk K (2006) GA-based pareto optimization, Vol. 16 of Studies in computational intelligence. Springer, Berlin Heidelberg, Newyork

8. Markowska-Kaczmar, Pawel Wnuk-Lipinski: Rule Extraction from Neural Network by Genetic Algorithm with Pareto Optimization. ICAISC 2004: 450-455

9. McMillan C, Mozer M.C, and P.smolensky. the Connectionist Scientist Game: Rule Extraction and Refinement in a Neural Network. In: Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society, Hillsdale, NJ, 1991.

10. Raul T. Santos, Jlio C. Nievola, Alex A. Freitas,"Extracting Comprehensible Rules from Neural Networks via Genetic Algorithms", Proc.2000 IEEE Symp. On Combination of Evolutionary Algorithm and Neural Network (2000).

11. Rumelhart, D. E., Hinton, G. E., and McClelland, J. L. A general framework for Parallel Distributed Processing In Rumelhart, D. E. and McClelland, J. L., editors, Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations, MIT Press, Cambridge, MA. pp 45-76, 1986.

12. Taha I, Ghosh J (1999) Symbolic interpretation of artificial neural networks. IEEE Transactions on Knowledge and Data Enginnring 11(3):448–463.

13. Thrun SB (1995) Extracting rules from artificial neural networks with distributed representations. In G. Tesauro, D. Touretzky and T. Leen, editors, Advances in Neural Information Processing Systems (NIPS) 7, Cambridge,MA, 1995. MIT Press.

14. Algèbre de Boole www.iut-info.univ-lille1.fr/~iovleff/pub/Teaching/MathInfo1/Poly3.pdf

15. UCI Repository of Machine Learning Databases. University of California, Irvine, Department of Information and Computer Sciences.