Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Abou Bekr Belkaid
Faculté de Technologie
Département de Génie Biomédical

MÉMOIRE DE FIN D'ÉTUDES

pour obtenir le grade de

Master en Génie Biomédical

Spécialité : **Informatique Biomédicale**

présenté et soutenu publiquement
par

Ben Mazouz Maamr

Khouani Amin

le 15 Juin 2015

Titre:

Optimisation paramétrique d'un classifieur neuronale par méta heuristiques : Application données médicales

Jury

Président du jury.	Mr.HADJ SLIMANE Zine,	MAA UABB Tlemcen
Examinateurr.	Mr. BEHDADA Omar,	MAA UABB Tlemcen
Encadreur.	Mr.CHIKH Mohammed El Amine,	PROF UABB Tlemcen
Co-encadreur.	Mlle.Bekaddour Fatima Zohra,	Doctorant UABB Tlemcen

Je dédié ce mémoire à :

Mes parents

Ma mère, qui a œuvré pour ma réussite, de par son amour, son soutien, tous les sacrifices consentis et ses précieux conseils, pour toute son assistance et sa présence dans ma vie, reçois à travers ce travail aussi modeste soit-il, l'expression de mes sentiments et de mon éternelle gratitude.

Mon père, qui peut être fier et trouver ici le résultat de longues années de sacrifices et de privations pour m'aider à avancer dans la vie. Puisse Dieu faire en sorte que ce travail porte son fruit ; Merci pour les valeurs nobles, l'éducation et le soutient permanent venu de toi.

Mes frères et mes cousins qui n'ont cessé d'être pour moi des exemples de persévérance, de courage et de générosité.

A toute ma famille.

A tous mes collègues de l'UABB.

BEN MAZOUZ.M

Je dédié ce modeste travail

A mes chers parents pour leurs encouragements, leur soutien moral, spirituel et leur tolérance durant toutes mes années d'études, tous les mots restent faibles pour exprimer mes sentiments, qu'ils trouvent a travers ce travail la récompense de leurs efforts. J'espère qu'Allah me donne la force et le courage pour que je puisse rendre leurs sacrifices.

A tous ceux qui m'ont aidé de loin ou de prés.

KHOUANI.A

Remerciements

Nous tenons dans un premier temps à remercier ALLAH le tout puissant de nous avoir donné la volonté, le courage et de nous avoir aidé tout au long de notre parcours éducatif.

Nous tenons à remercier sincèrement Mr Chikh Mohamed.Amine en tant que directeur de mémoire pour son attention aussi particulière, pour ses conseils avisés qui ont été prépondérant pour la bonne réussite de ce projet. Nous avons beaucoup appris de son savoir, sa méthodologie et son expérience. Pour tout cela nous lui seront éternellement reconnaissantes. Avec tous nos respects et notre gratitude.

Nos vifs remerciements vont également à Melle Bekkadour Fatima, en tant que Co-encadreur, pour ses remarques et suggestions pour améliorer la qualité de ce mémoire.

Nous remercions également Mr Hadj Slimane Zine, de nous faire l'honneur de présider ce jury.

Nos remerciements à Mr Benhadada Omar qui a participé à l'examen de ce travail.

Nous remercions infiniment tous les enseignants qui nous ont aidé durant tout notre cycle d'études.

Un grand merci à tous ceux qui ont contribué de près ou de loin pour l'élaboration de ce travail.

Résumé

La spécification correcte des paramètres d'un RNA quasi optimale d'une manière automatique pour un problème spécifique avec une performance satisfaisante est le principale aspect qui est motivé notre approche présent dans ce mémoire de master.

Cette approche emploie une recherche évolutive (AG ou RS) pour effectuer le réglage simultané des poids initiaux, des fonctions d'activation, fonctions d'apprentissage, pas d'apprentissage, et le moment de la PMC. Des expériences ont été effectuées et les résultats obtenue après l'utilisation de cette méthode montrent qu'elle est capable de trouver une meilleure configuration pour effectue une bon classification. Nous avons validé nos résultats expérimentaux sur des bases de données médicales connues : *Pima (Diabètes)*, *TH (Troubles Hépatiques)*, *AP (Appendicite)*.

Mots clés

Algorithmes évolutionnaires, Algorithmes génétiques, Recuit simulé, Optimisation, Métaheuristique, Paramétrage de réseau de neurone artificielle, Les poids initiaux.

Abstract

The correct specification of parameters of a quasi-optimal RNA in an automatic way to a specific problem with a satisfactory performance is the main aspect that motivated our present approach in this master thesis.

This approach employs an evolutionary search (GA or AS) to perform simultaneous tuning of the initial weights, activation functions, learning functions, learning rat, and momentum of MLPNN. Experiments were performed and results obtained after using this method show that it is able to find a better configuration for conducting a proper classification. We validated our experimental results on known medical databases : *Pima (Diabetes), TH (Liver Disorders), AP (appendicitis)*.

Keywords

Evolutionary algorithms, Genetic algorithms, Simulated annealing, Optimization, Metaheuristics, Artificial neural network setting, The initial weights.

Table des matières

Table des figures

Liste des tableaux

Glossaire

AG : Algorithme Génétique
AS :Apprentissage Supervisé
AP : Appendicite(Appendicitis)
BP :Back-Propagation
BDD :Base De Donnée
BUPA : British United Provident Association
FN : Faux Négatif
FP : Faux Positif
Fobj :Fonction Objectif
GA : Genetic Algorithm
HL : Hill climbing
I.A : Intelligence Artificielle
L.F : Logique Floue
PID : Pima Indian Diabets
P.M.C : Perceptron Multi-Couches
PAP : Problème d'Ajustement Parametrique
R.N.A : Réseau de Neurone Artificiel
RS :Récuit simulé
Se : Sensibilité
Sp : Spécificité
SE : Stratégies d'Evolution
Tc : Taux de Classification
TH : Troubles Hépatiques
UCI : University of California, Irvine
VN : Vrai Négatif
VP : Vrai Positif

Introduction générale

Le Réseaux de neurone artificiel (RNA) est un sujet de grand intérêt pour les instituts de recherche du monde entier. Il existe un grand nombre d'articles dans la littérature consacrée à l'analyse et l'utilisation de RNA. Applications dans des domaines médicales, industriels et commerciaux ont contribué plus fortement vers prouvant l'importance, L'efficience et l'efficacité des RNA [7]. Le succès de l'utilisation des RNA dans des domaines tels que la reconnaissance des formes, traitement du signal, la prévision et l'analyse financière etc. est incontestable [8]. Une telle utilisation généralisée de RNA est principalement due à son émulation du comportement du cerveau humain et de sa capacité d'apprentissage. RNA peut être qualifié comme un processus parallèle et distribué qui est constitué d'éléments de traitement multiples. RNA ont émergé comme un outil important en termes de vitesse d'apprentissage, la précision et l'immunité au bruit et la capacité de généralisation.

Beaucoup a été étudié et recherché pour essayer de trouver des moyens systématiques pour ajuster les paramètres d'un modèle RNA pour un problème en particulier visant à optimiser sa performance globale. Ces enquêtes sont basées sur presque tous les aspects de la RNA modélisation tels que les différents types de fonction d'activation, l'initialisation des poids, la collecte de données de formation, pré et post-traitement, les algorithmes de formation, et les fonctions d'erreur. Toutefois, des preuves théoriques et empiriques trouvées dans la littérature montrent la recherche et la définition des paramètres optimale ou quasi-optimale d'un RNA sont les plupart des facteurs importants pour le modèle de calcul des performances, l'efficacité et la précision .

L'apprentissage supervisé (AS) est l'un des algorithmes de formation de poids les plus efficaces, de sorte que des efforts sont faits pour trouver un ensemble optimal des poids de connexions pour un RNA selon certains critères d'optimalité. Un des algorithmes de formation d' AS les plus populaires la rétro-propagation (ou backpropagation BP). Le BP est un algorithme de recherche de descente de gradient. Il est basé sur la minimisation de l'erreur quadratique moyenne totale entre la production réelle et la sortie désirée. Cette erreur est utilisée pour guider la recherche de l'algorithme BP dans l'espace en poids. Le BP est largement utilisé dans de nombreuses applications, en ce qu'il n'y a pas lieu de déterminer la structure exacte et les paramètres de RNA à l'avance. Cependant, le problème de l'algorithme BP est qu'il est très souvent pris au piège dans des minima locaux et l'apprentissage et la vitesse d'adaptation est très lent dans la recherche de minimum global de l'espace de recherche. La vitesse et la robustesse de l'algorithme BP sont sensibles à plusieurs paramètres de l'algorithme et les meilleurs paramètres varient de problèmes vers autre.

Récemment, certains algorithmes évolutionnaires développés, notamment les algorithmes génétiques (AG), ont attiré une grande attention dans les domaines RNA. l'AG est proposé pour optimiser la topologie RN et les poids. L'AG est utilisé pour identifier les

neurones sans importance et supprimer ces neurones pour donner une structure optimal. En travaillant avec une population de solutions, l'AG peut demander beaucoup minima locaux, et donc d'augmenter la probabilité de trouver le minimum global. Cet avantage de l'AG peut être appliquée pour optimiser les paramètres et la topologie des RNA.

Les AGs sont des procédures de recherche stochastiques basés sur les mécanismes de la sélection naturelle, génétique et l'évolution . Depuis ils évaluent simultanément de nombreux points dans l'espace de recherche, ils sont plus susceptibles de trouver la solution globale d'un problème donné. En outre, ils utilisent uniquement une mesure de performance scalaire simple qui ne nécessite pas ou utiliser l'information dérivée, ils sont donc des méthodes d'optimisation d'usage général pour résoudre les problèmes de recherche. Deux principaux domaines majeurs dans lesquels les AGs ont été utilisés : l'optimisation et de l'apprentissage de la machine. Dans l'apprentissage machine, les AGs ont été appliquées avec succès à l'apprentissage des réseaux de neurones.

Un réseau de neurones optimal est un RNA adaptée à un problème spécifique, ayant ainsi une architecture plus petite avec une convergence plus rapide et une meilleure performance. Un RNA presque optimale (ou quasi-optimale) se caractérise par le choix de ses paramètres spécifiques, corrigées pour un problème spécifique, en produisant ainsi une performance satisfaisante. La recherche des paramètres des RNA est généralement effectuée par un développeur (par une procédure d'essai-erreur). Ainsi, l'optimalité ou même quasi-optimalité n'est pas assurée.

Cette technique de réglage manuel est considérée comme une tâche fastidieuse et source d'erreurs. Lorsque la complexité de problème augmente et quand les réseaux quasi-optimales sont souhaitées, la recherche manuelle devient plus difficile et ingérable. La construction manuelle d'un configurations quasi-optimales RNA implique des difficultés comme le nombre exponentiel de paramètres qui doivent être ajustés.

Considérant les problèmes et les difficultés liées à l'utilisation de la méthode manuelle pour accorder les paramètres RNA, l'adoption d'une méthode automatique pour effectuer cette tâche d'accord émerge dans le but d'éviter de tels problèmes.

En résumé, la problématique définie ci-dessus s'énonce par : Comment développer un classifieur neuronal ayant des paramètres optimales pour la classifcation supervisée des données médicales ?

Dans ce travail, nous appliquons un groupe de métaheuristiques pour résoudre ce problème. Un algorithme hybride basé sur une technique AG et RS est proposée pour optimiser les paramètres du réseau neuronal PMC (MLPNN). Dans la première une représentation de chromosome simple est utilisé, qui contient des informations sur la fonction d'activation, fonction d'apprentissage, pas d'apprentissage, et le moment, de la PMC(MLPNN). Le processus d'apprentissage de paramètre, basé sur la technique AG et l'algorithme BP est un processus d'apprentissage en deux étapes. Dans la première étape, les paramètres initiaux sont accordés par l'AG .Dans la deuxième étape, l'algorithme BP est introduit pour former la première RN pour produire des paramètres optimales de la RNA.

Une autre contribution de ce travail ,nous avons construit une fonction objectif (Fobj) qui consiste en une combinaison de deux critères : l'erreur et pénalité. La fonction fobj

est la fonction dont la valeur doit être minimisée par l'algorithmes mis en œuvre. Ainsi, il devient clair que la fobj est composé par la somme des erreurs d'apprentissage et de généralisation multiplié par une pénalité(qui convient de la complexité de l'architecture de RNA et le temps d'apprentissage).

L'erreur de généralisation et d'apprentissage sont multiplié aussi par deux facteurs de pondération (P1 et P2) qui mesurent le degré d'importance de ces erreurs dans le processus l'algorithmique de recherche, ce qui nous a permet de peaufiner les progrès de l'optimisation.

Le plan de ce mémoire est composé de :

- Chapitre 1 Généralités sur les métaheuristiques, Ce chapitre décrit le cadre de l'optimisation et de la métaheuristique.
- Chapitre 2 problème d'optimisation paramétrique d'un classifieur neuronal, nous présentons quelques notions de bases sur les RNA en particulier le Perceptron Multi-couches Puis, Nous nous intéresserons par la suite sur le problème étudie : le PAP (problème d'ajustement paramétrique) des RNAs.Ce chapitre sera terminé par une présentation des quelque travaux existants liée à la résolution du PAP des RNAs qui utilisent les métaheuristiques pour ajuster les paramètres d'un RNA.
- Chapitre 3 Implémentation, décrivant l'implémentation de notre classifieur neuronal avec l'approche utilisé comme hybridation, ce travail comprend deux phases : Phase de réalisation de notre classifieur neuronal,et phase d'amélioration paramétrique par métaheuristique (algorithme génétique et recuit simulé), les résultats obtenus sont discutés.
- En dernier lieu, une conclusion générale et des perspectives de ce travail de Master seront présentées.

Chapitre 1

Généralités sur les métaheuristiques

1 Introduction

L'optimisation est un sujet central en recherche opérationnelle, un grand nombre de problèmes d'aide à la décision pouvant en effet être d'écrits sous la forme de problèmes d'optimisation. Les problèmes d'identification, d'apprentissage supervisé de réseaux de neurones ou encore la recherche du plus court chemin sont, par exemple, des problèmes d'optimisation. Ce chapitre décrit tout d'abord le cadre de l'optimisation et de la métaheuristique dans lequel nous nous plaçons dans ce travail.

2 Optimisation

L'optimisation est une branche des mathématiques et de l'informatique en tant que disciplines, cherchant à modéliser, à analyser et à résoudre analytiquement ou numériquement les problèmes qui consistent à déterminer quelles sont la ou les solution(s) satisfaisant un objectif quantitatif tout en respectant d'éventuelles contraintes.
Un problème d'optimisation combinatoire est généralement caractérisé par un ensemble fini de solutions admissibles W et une fonction objectif f : WßR associant une valeur à chaque solution admissible. La résolution du problème consiste à déterminer la (ou les) solution(s) de W minimisant ou maximisant f, c'est-à-dire l'optimum global [9]. Cependant il peut exister des solutions intermédiaires, qui sont également des optimums, mais uniquement pour un sous-espace restreint de l'espace de recherche : on parle alors d'optimums locaux. Cette notion est illustrée sur la figure 1.1.

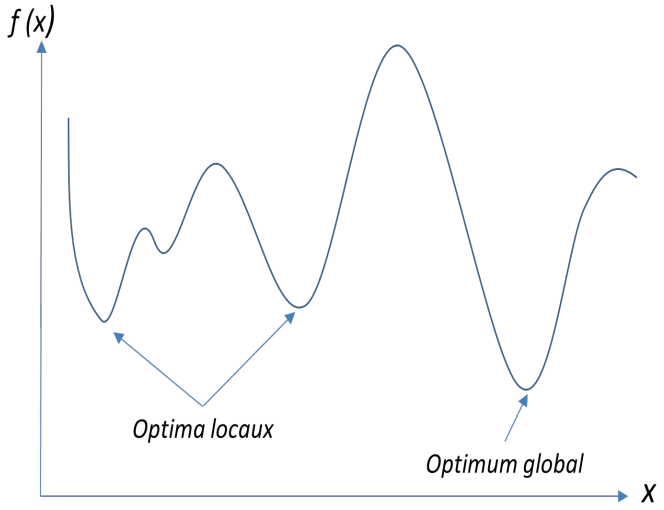

FIGURE 1.1 – Différence entre un optimum global et des optimums locaux [1].

2.1 Les différents problèmes d'optimisation

Il existe plusieurs classes d'optimisation, dont nous citons les plus connues (figure1.2) :

- Optimisation Monovariable et Multivariable : Les problèmes d'optimisation monovariable désignent habituellement tous les problèmes dont la fonction objectif ne dépend que d'une seule variable. Les problèmes d'optimisation multivariable rassemblent les problèmes dont la fonction objective est définie par plus d'une variable.

- Optimisation Uni-modale et Multimodale : Un problème dont f ne compte qu'un seul optimum est appelé problème uni-modale. Dans ce cas, un optimum local est aussi un optimum global. A l'inverse, Lorsqu'une fonction admet plusieurs optima locaux, elle est dite multimodale.

- Optimisation Mono Objectif et Multi Objectif : Les problèmes d'optimisation mono-objectif regroupent les problèmes n'ayant qu'un objectif unique à optimiser. Les problèmes d'optimisation multi-objectif possèdent plus d'un objectif à optimiser.

- Optimisation continue et discrète : A la différence de l'optimisation continue qui recherche une solution dans un ensemble infini d'objets (ex : l'ensemble des nombres réels), l 'optimisation discrète regroupe les problèmes dont la solution recherchée est une combinaison d'éléments parmi un ensemble fini d'objets.

- Optimisation avec et sans contraintes : Les problèmes d'optimisation sans contraintes sont des problèmes pour lesquels toute solutions appartenant à S est considérée comme une solution faisable et acceptable. Les problèmes d'optimisation avec contraintes sont des problèmes dont une solution s appartenant à S ne peut être considérée comme solution faisable que si elle satisfait l'ensemble des contraintes prédéfinie.

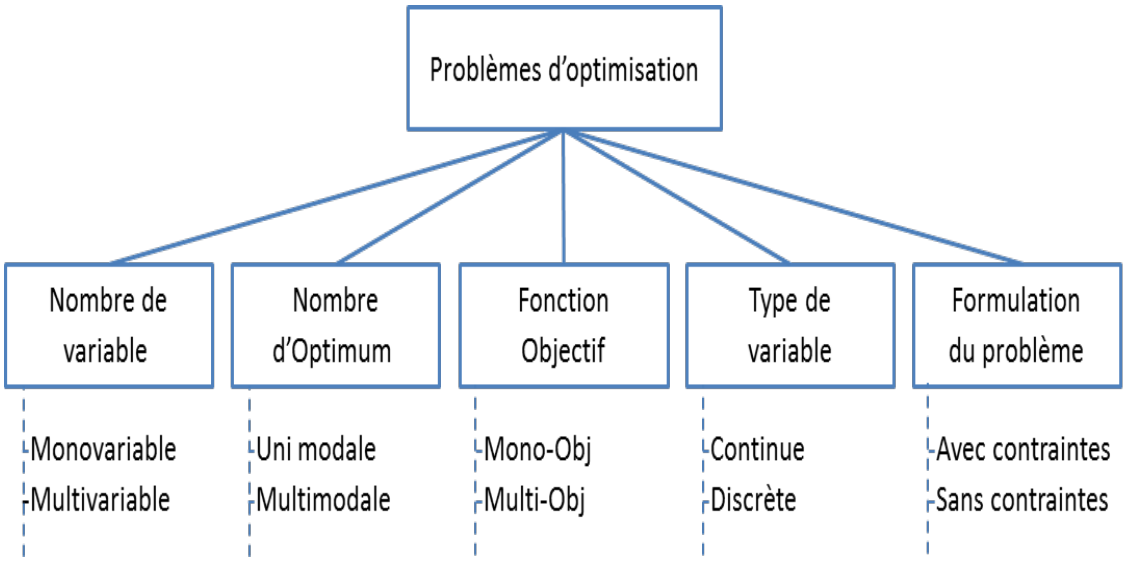

FIGURE 1.2 – Classification des problèmes d'optimisation [2].

2.2 Classification des méthodes d'optimisation

Deux types de méthodes sont souvent mis en opposition, dans la résolution des problèmes d'optimisation (Figure 1.3) : les méthodes exactes et les méthodes approchées. Les méthodes exactes ont l'avantage d'obtenir des solutions dont l'optimalité est garantie .Par contre, pour les problèmes NP-complets, ces algorithmes sont très coûteuses en temps d'exécution (algorithmes à temps non polynomial). Les méthodes approchées génèrent des solutions de haute qualité en un temps de calcul raisonnable, mais il n'existe aucune garantie de trouver la solution optimale [3].

Méthodes exacts

Les méthodes exactes ont l'avantage d'obtenir des solutions dont l'optimalité est garantie. Ces méthodes sont génériques et demandent souvent une particularisation vis-à-vis d'un problème spécifique [9]. Parmi les méthodes exactes, on peut citer :

- La programmation dynamique : consistant à placer le problème dans une famille de problèmes de même nature mais de difficulté diffiérente puis à trouver une relation de récurrence liant les solutions optimales de ces problèmes [9].

- Le Branch & Bound : consistant à faire une énumération implicite en séparant le problème en sous-problèmes et en évaluant ceux-ci à l'aide d'une relaxation (continue ou lagrangienne principalement) jusqu'à ne plus avoir que des problèmes faciles à résoudre ou dont on sait avec certitude qu'ils ne peuvent pas contenir de solution optimale [9].

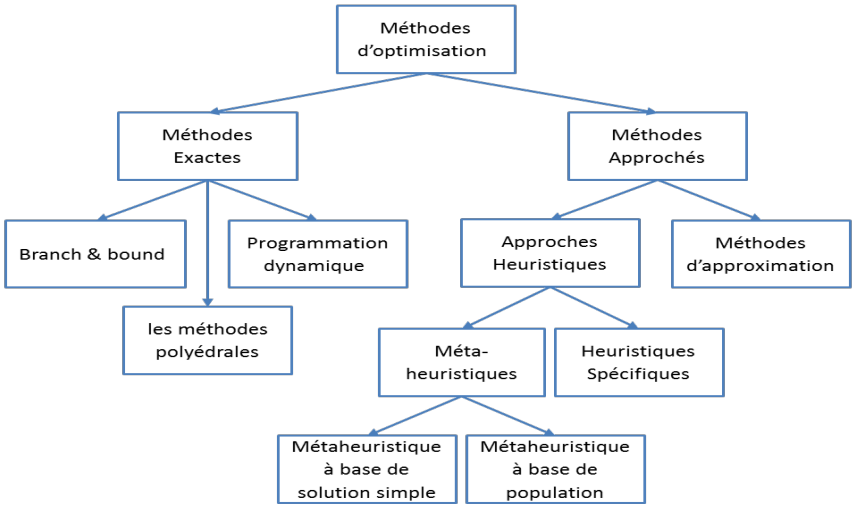

FIGURE 1.3 – Classification des méthodes d'optimisation [3].

– les méthodes polyédrales : consistant à ajouter progressivement des contraintes supplémentaires an de ramener le domaine des solutions admissibles à un domaine convexe (sans en enlever la ou les solutions optimales bien évidemment) [9].
Il existe aussi des outils logiciels génériques (AMPL, CPLEX, LINDO, MPL, OMP, XPRESS...) permettant de résoudre l'ensemble des problèmes pouvant s'écrire sous la forme algébrique d'un problème en variables binaires, entières ou mixtes.

Méthodes approchées

Pour ces méthodes, on abandonne la contrainte de trouver une solution optimale et on cherche un algorithme polynomial qui trouve une solution, quelque fois optimale souvent acceptable. Ces méthodes approchées permettent néanmoins de trouver une bonne solution en un temps de calcul raisonnable comparés aux méthodes exactes. Ces méthodes approchées sont divisées en deux sous-catégories : les algorithmes d'approximation et les heuristiques. Contrairement aux heuristiques, les algorithmes d'approximation garantissent la limite (bound) de la solution trouvée par rapport à l'optimal. Enfin, il existe encore deux sous-classes de méthodes heuristiques : lesheuristiques spécifiques à un problème donné, et les méta-heuristiques.

– Approches heuristiques Une heuristique d'optimisation est une méthode approchée se voulant simple, rapide et adaptée à un problème donnée. Sa capacité à optimiser un problème avec un minimum d'informations est contrebalancée par le fait qu'elle n'offre aucune garantie quant à l'optimalité de la meilleure solution trouvée. Du point de vue de la recherche opérationnelle, ce défaut n'est pas toujours un problème, tout spécialement quand seule une approximation de la solution optimale est recherchée.

– Les métaheuristique Parmi les heuristiques, certaines sont adaptables à un grand nombre de problèmes différents sans changements majeurs dans l'algorithme, on parle alors de méta -heuristiques. La plupart des heuristiques et des métaheuristiques utilisent des processus aléatoires comme moyens de récolter de l'information et de faire face à des problèmes comme l'explosion combinatoire. En plus de cette base stochastique, les métaheuristiques sont généralement itératives, c'est-à-dire

qu'un même schéma de recherche est appliqué plusieurs fois au cours de l'optimisation, et directes, c'est-à-dire qu'elles n'utilisent pas l'information du gradient de la fonction objectif. Elles tirent en particulier leur intérêt de leur capacité à éviter les optima locaux, soit en acceptant une dégradation de la fonction objectif au cours de leur progression, soit en utilisant une population de points comme méthode de recherche (se démarquant ainsi des heuristiques de descente locale) [5].

3 Principes du métaheuristique

Etymologiquement, les métaheuristiques sont des abstractions des heuristiques. Il s'agit d'algorithmes basés sur des heuristiques et qui s'appuient sur 3 principes fondamentaux (Figure1.4)

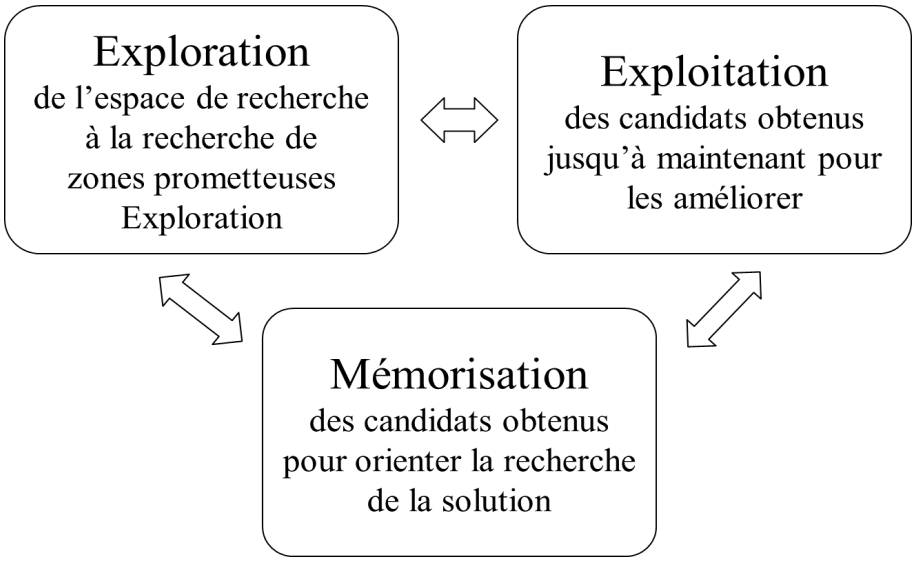

FIGURE 1.4 – les trois principes des métaheuristiques.

Contrairement aux algorithmes de recherche locale, les métaheuristiques disposent de mécanismes d'exploration qui cherchent, dans l'espace des solutions, des zones prometteuses où pourrait se trouver la solution. Cependant, l'algorithme ne doit pas seulement explorer l'espace, mais aussi exploiter les candidats obtenus pour tenter de les améliorer. Exploitation et exploration ne sont possibles qu'avec un mécanisme de mémorisation qui permet de conserver certaines informations passées.

Il n'existe actuellement aucune définition communément acceptée pour les métaheuristiques. [10] a définit la métaheuristique comme :
« Une métaheuristique est formellement défini comme un processus itératif qui guide une heuristique subordonnée en combinant intelligemment différents concepts d'exploration et d'exploitation de l'espace de recherche. Des stratégies d'apprentissage sont utilisées pour structurer l'information afin de trouver efficacement des solutions optimales, ou presque-optimales »

Selon [3] il y'a deux questions de conception liées à toutes les métaheuristiques itératives :

– Représentation de solutions traitées(encodage).

– Définition de la fonction objectif qui guidera la recherche.

4 Propriétés des métaheuristiques

Selon [11], les auteurs résument les propriétés fondamentales attachées à la notion de métaheuristique :

– Les métaheuristiques sont des stratégies qui guident le processus de recherche.
– L'objective des métaheruristiques est d'explorer efficacement l'espace de recherche afin de trouver des solutions (proche de) optimale.
– Les techniques mises en œuvre dans les métaheuristiques vont de la simple recherche locale à des procédures complexes d'apprentissage.
– Les métaheuristiques sont des algorithmes approximatifs et généralement de nature non-déterministe.
– Elles peuvent intégrer des mécanismes évitant d'être piégé dans une zone de l'espace de recherche.

5 Principales métaheuristiques

Il y a différents critères pour classifier et décrire les algorithmes métaheuristiques, mais généralement, les métaheuristiques sont classées en deux groupes :les Métaheuristiques de Trajectoire(figure1.5) et les métaheuristiques à base de population (figure 1.6). Pour chaque groupe nous donnons un aperçu des méta-heuristique les plus répandues :

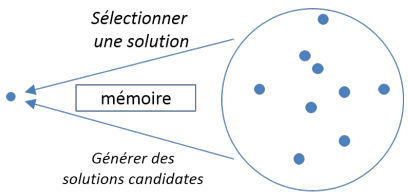

FIGURE 1.5 – Illustration des principes généraux d'une métaheuristique à base de solution unique .

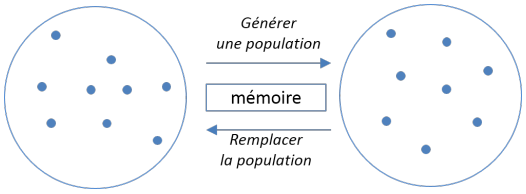

FIGURE 1.6 – Illustration des principes généraux d'une métaheuristique à base de population.

5.1 Métaheuristiques de trajectoire

les Métaheuristiques de trajectoire regroupent ensemble des méthodes consiste à faire évoluer une solution initiale en s'en éloignent progressivement, pour réaliser une trajectoire, un parcours progressif en tentant de se diriger vers des solutions optimales (ex : Hill climbing, recherche Tabou).

Recherche par descente (Hill Climbing)

La méthode de recherche locale la plus élémentaire est la méthode de la descente, appelée aussi recherche par descente ou par escalade(Hill Climbing). Elle peut être décrite comme suit. À partir d'une solution quelconque, on cherche une solution dans le voisinage et on accepte cette solution si elle améliore la solution courante (Hoos et Stützle, 2004) [12].

Nous présentons dans l'algorithme 1.1 une méthode de descente simple (méthode de la première amélioration). Partant d'une solution initiale s, on choisit une solution $s0$ dans le voisinage $N(s)$. Si la solution s0 à un coût meilleur que s, alors on accepte cette solution comme nouvelle solution$s$ et on recommence le processus jusqu'à ce qu'un optimum local soit atteint.

Une version plus efficace de la méthode de la première amélioration est la méthode de la meilleure amélioration. Au lieu de choisir la première solution voisine $s0$ de qualité supérieure à s, on choisit toujours la meilleure solution $s0$ du voisinage s. L'algorithme 1.2 illustre cette méthode.

Algorithme 1.1 : Méthode de la première amélioration

Entrées : $S, \mathcal{N}, f$
Sorties : une solution s correspondant à un optimum local
début
  initialiser : générer une solution initiale $s \in S$
  répéter
    rechercher dans le voisinage : choisir une solution $s' \in \mathcal{N}(s)$
    si $f(s') > f(s)$ **alors**
      $s \leftarrow s'$
    fin
  jusqu'à $f(s') \leq f(s), \forall s' \in \mathcal{N}(s)$
  retourner s
fin

Algorithme 1.2 : Méthode de la meilleure amélioration

Entrées : $S, \mathcal{N}, f$
Sorties : une solution s correspondant à un optimum local
début
  initialiser : générer une solution initiale $s \in S$
  répéter
    rechercher dans le voisinage : choisir une solution $s' \in \mathcal{N}(s)$ tel que $f(s') > f(s)$ et $\forall s'' \in \mathcal{N}(s), f(s'') \leq f(s')$
    $s \leftarrow s'$
  jusqu'à $f(s') \leq f(s), \forall s' \in \mathcal{N}(s)$
  retourner s
fin

Recherche tabou

Les racines de la recherche tabou remontent aux années 70 [13]. Cette méthode a d'abord été présentée par Glover (Glover, 1986) [14] ; les idées de base ont aussi été esquissées par (Hansen, 1986) [15]. D'autres efforts supplémentaires de la formalisation de cette méthode sont présentés dans (Glover, 1989 ; Glover, 1990a ; Glover et Laguna, 1998) [16], [17], [18].

La recherche avec tabous, ou simplement ''recherche tabou'' se base sur l'utilisation de structures de mémoire. Le principe de base consiste à choisir la meilleure solution du voisinage de la solution courante, parfois moins bonne que la solution courante. Par conséquent, la recherche ne s'arrête pas au premier optimum local trouvé. Le danger est alors de revenir à des solutions déjà explorées. Pour éviter des cycles, on mémorise les dernières solutions visitées dans une liste tabou et on interdit tout mouvement qui conduit à une solution de cette liste. La liste tabou est donc une sorte de mémoire à court terme. Tout mouvement qui nous mène de la solution courante à une solution de la liste tabou est appelé mouvement tabou.

Lorsque la liste tabou est trop complexe ou occupe une grande place de mémoire, on mémorise plutôt des attributs des solutions ou des modifications des solutions que des solutions entières. La mémorisation de ces attributs ou modifications permet certes un gain en place mémoire, mais elle n'a pas que des avantages.

Le stockage des attributs ou des modifications des solutions, permet d'identifier une solution de bien meilleure qualité avec un statut tabou. Accepter tout de même cette solution revient à dépasser son statut tabou, c'est l'application du critère d'aspiration. Ce mécanisme permet alors de lever le statut tabou d'une solution, sans pour autant introduire un risque de cycles dans le processus de recherche. La fonction d'aspiration la plus répandue est celle qui consiste à révoquer le statut tabou d'un mouvement tabou si ce dernier permet d'atteindre une solution de qualité supérieure à celle de la meilleure solution trouvée jusqu'alors.

Un facteur critique est la taille de la liste tabou. Si sa valeur est trop petite, des cycles

peuvent se produire dans le processus de recherche. Au contraire, si sa valeur est trop grande, des mouvements de bonne qualité peuvent être interdits et mènent à l'exploration de solutions de qualité plus basse. Certains auteurs, comme (Taillard, 1991) [19], proposent l'utilisation d'une liste tabou de longueur variable. D'autres auteurs, comme (Batitti et Techioiii, 1994) [20], préfèrent les listes tabou réactives qui varient selon les résultats de la recherche.

Algorithme 1.3 : Méthode de recherche tabou

Entrées : $S, \mathcal{N}, f$
Sorties : une solution s^* correspondant à un optimum

début
  initialiser : générer une solution initiale $s \in S$
  $s^* \leftarrow s$
  $\mathcal{T} \leftarrow \oslash$
  répéter
    construire $\mathcal{N}^*(s) = \{s'' \in \mathcal{N}(s) \setminus \mathcal{T}\}$
    choisir une solution $s' \in \mathcal{N}^*(s) : f(s') \geq f(s''), \forall s'' \in \mathcal{N}^*(s)$
    mettre à jour $\mathcal{T}$ avec s
    si $f(s') > f(s^*)$ **alors** $s^* \leftarrow s', s \leftarrow s^*$
  jusqu'à *satisfaire un critère d'arrêt*
  retourner s^*
fin

Recuit simulé

Le recuit simulé est fondé sur une analogie entre un processus physique (le recuit) et le problème de l'optimisation(voir le tableau1.1). Le recuit simulé en tant que métaheuristique [21] s'appuie en effet sur des travaux visant à simuler l' évolution d'un solide vers son état d'énergie minimale [22].

Problème d'optimisation	Système physique
Solution	État du système
Fonction objectif	Énergie libre (E)
Paramètres du problème	Coordonnées des particules
Trouver une bonne configuration	Trouver les états à basse énergie
Optimum global	État stable ordonné
Optimum local	État métastable
Recherche locale	Trempe rapide
Le paramètre T	La température

TABLE 1.1 – Analogie entre un problème d'optimisation et un système physique [5], [3].

Le principe de base du recuit simulé est alors emprunté à la physique du solide : on chauffe un morceau de métal, puis on le laisse refroidir lentement. Si le refroidissement est trop brusque, les atomes peuvent s'éloigner de leur état d'équilibre et causer une imperfection au morceau de métal. Le refroidissement du métal est donc lent et régulier pour

permettre aux atomes de se stabiliser peu à peu dans une position d'énergie minimale.

L'algorithme de recuit simulé consiste en une recherche aléatoire de l'espace d'état, à favoriser les descentes, mais sans interdire tout à fait les remontées. Cependant, une augmentation du niveau d'énergie permet de sortir des minima locaux (figure 1.7).

L'algorithme montre qu'on tend très rapidement vers un minimum local proche de la meilleure solution.

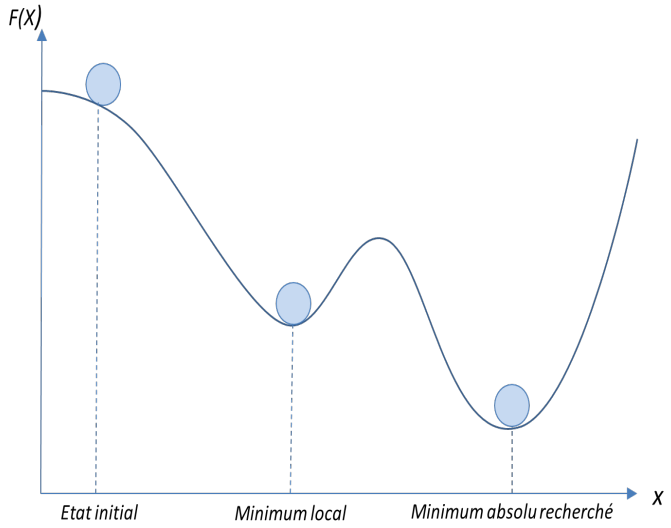

FIGURE 1.7 – La recherche dans l'espace d'état avec le recuit simulé [4].

Au début de l'algorithme 1.4, la probabilité est proche de 1 et presque toutes les variations ΔE sont acceptables. Lorsque la température diminue, les remontées sont de plusen plus difficiles et très peu de variations peuvent être acceptées.

Algorithme1.4 : recuit simule

Entrées :

s=s_0 ; *// Génération aléatoire d'une solution s*

T=T_{max} ; *// Température initiale*

Sortie : meilleur solution trouvée

1. **Début :**
2. **Répéter**
3. Génération aléatoire d'une solution s' voisine de s
4. $\Delta E = f(s') - f(s)$;
5. **Si** ($\Delta E < 0$) alors s ←s'
6. **Sinon** faire s ←s' avec une probabilité $e^{\frac{f(s)-f(s')}{T}}$
7. Diminuer T
8. **Jusqu'à** (Critère d'arrêt satisfait)
9. **Fin**

5.2 Métaheuristiques de population

Contrairement aux méthodes précédentes, les approches à base de population consistent à travailler avec un ensemble de solutions simultanément, que l'on fait évoluer graduellement. Ceci permet d'améliorer l'exploration de l'espace de recherche.
Dans cette seconde catégorie, on recense : Les algorithmes évolutionnistes (EvolutionaryAlgorithms) qui forment une classe importante des métaheuristiques à base de population.

Algorithmes génétiques (AG)

Les algorithmes génétiques ont été proposés par Holland (1975) [23] puis approfondis par Goldberg en 1989 [24]. Ils ont été inspirés des principes de la génétique naturelle et de la théorie d'évolution (La présence ou l'absence des gènes et de leur ordre dans le chromosome décident des caractéristiques d'une espèce. Aussi, différents traitements sont passés d'une génération à la prochaine à travers différents processus biologiques qui déterminent la structure génétique).
Dans un AG(Figure 3.3), chaque solution est stockée dans un chromosome artificiel représenté par un code (binaire, entiers...). Chacun de ces chromosomes est défini par deux caractéristiques.

- Le premier est leur génotype, qui est la chaîne des symboles qui définit le chromosome. Il s'appelle comme ceci en raison de l'analogie avec un ordre génétique dans la biologie.

- Le second est le phénotype, le phénotype est une solution du problème dans une représentation naturelle. Avec chacun des chromosomes, les paramètres sont décodés et évalués par la fonction fitness pour déterminer la qualité du phénotype.

De nouveaux candidats sont produits d'un ensemble de population existant en appliquant des opérateurs artificiels génétiques choisis (croisement et mutation)

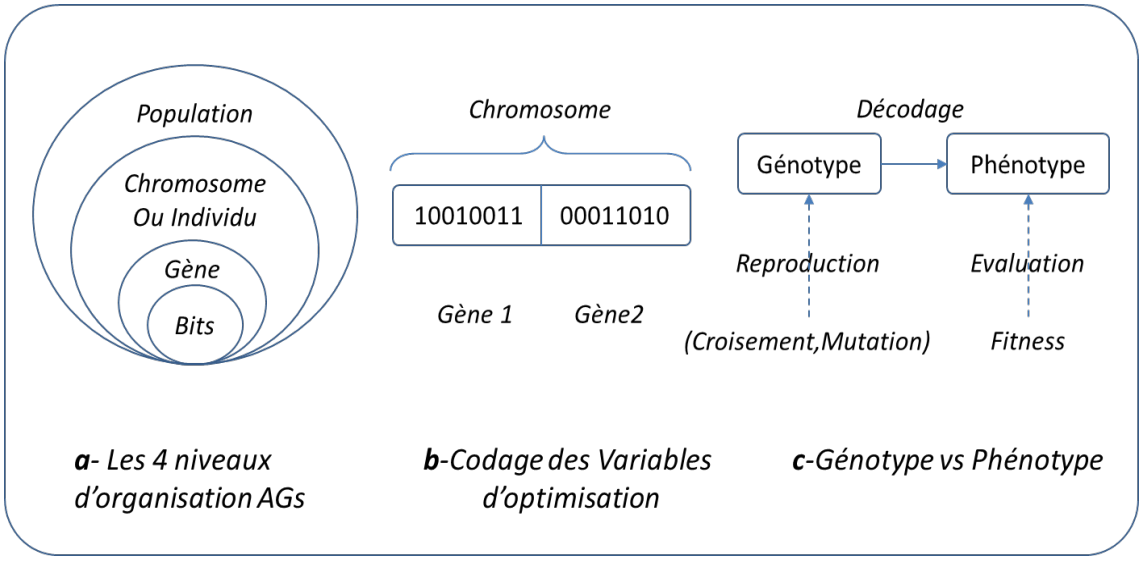

FIGURE 1.8 – Les concepts principaux utilisés dans les algorithmes génétiques [3].

Principe de fonctionnement d'un algorithme génétique : Les algorithmes génétiques sont des méthodes de recherche probabilistes, basés sur la théorie darwinienne d'évolution.

les individus soumis à l'évolution sont des solutions appartenant à l'espace de recherche du problème à optimiser. L'ensemble des individus traités simultanément par l'algorithme génétique constitue une population. L'algorithme itère et passe d'une génération à l'autre jusqu'à la satisfaction d'un critère d'arrêt. Durant chaque génération, un ensemble d'opérateurs est appliquée aux individus d'une population pour engendrer la nouvelle population à la génération suivante. Le principe d'un algorithme génétique se décrit simplement. Un ensemble de N points dans un espace de recherche, choisi àpriori au hasard, constitue la population initiale ; chaque individu x de la population possède une certaine performance, qui mesure son degré d'adaptation à l'objectif visé : dans le cas de la minimisation d'une fonction objectif f, x est d'autant plus performant que $f(x)$ est plus petit. Un AG consiste à faire évoluer progressivement par générations successives, la composition de la population en maintenant sa taille constante. Au cours des générations, l'objectif est d'améliorer globalement la performance des individus ; on essaie d'obtenir un tel résultat en mimant les deux principaux mécanismes qui régissent l'évolution des êtres vivants, selon la théorie de Darwin [25] :

- La sélection, qui favorise la reproduction et la survie des individus les plus performants.
- La reproduction, la recombinaison et les variations des caractères héréditaires des parents, pour former des descendants aux potentialités nouvelles.

La figure 1.9 suivante présente le processus d'un algorithme génétique simple.

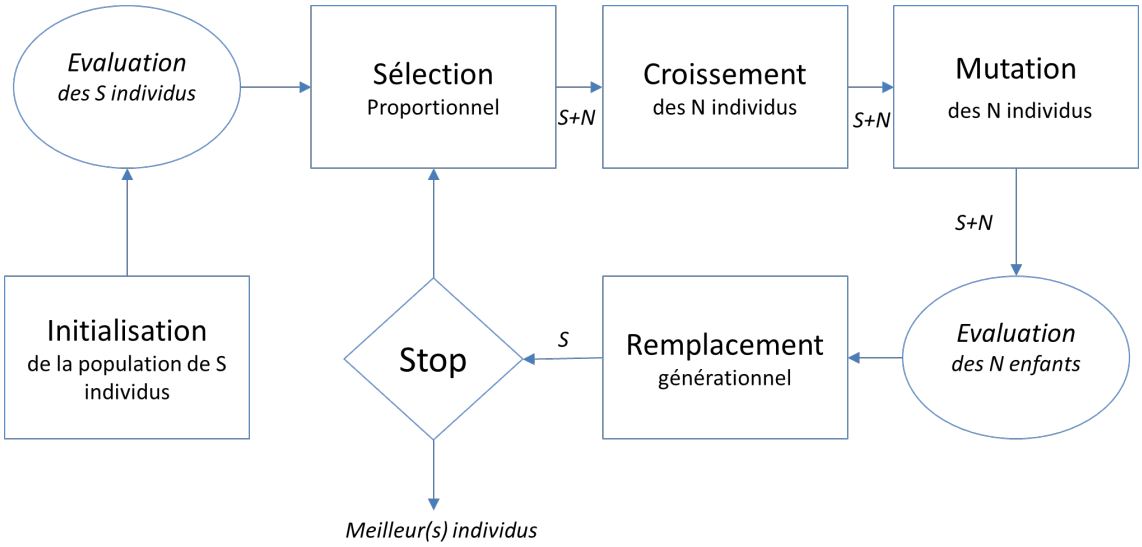

FIGURE 1.9 – le schéma d'un algorithme génétique.

Stratégies d'évolution

Les stratégies d'évolution constituent une méthode d'optimisation développée en 1964 par Rechenberg et Schwefel. Elle est à base de population, inspirée de mécanismes biologiques (en particulier génétiques), approximative, stochastique, d'ordre 0 et globale.Cependant, cet algorithme est considéré comme un algorithme évolutionnaire (ou évolutionniste), au même titre que les Algorithmes Génétiques (GA), du fait qu'il utilise des mécanismes inspirés de la théorie de l'évolution.

Les Stratégies d'Evolution ont été initialement conçues pour résoudre des problèmes à variables continues. Elles utilisent généralement la méthode de remplacement d'élitiste

(consiste à conserver le meilleur individu de la population actuelle dans la population de génération suivante).L'opérateur de croisement est rarement utilisé. L'opérateur de sélection est déterministe et basé sur la sélection proportionnelle au rang. Leur principale avantage est leur efficacité temporelle.

6 Conclusion

Les méthodes d'optimisation existent donc sous diverses formes, chacune étant adaptée à un type de problème particulier. Certains problèmes difficiles ne peuvent d'ailleurs pas être résolus de manière optimale par une machine. Pour pallier ce problème, des techniques ont été présentées, et notamment des méthodes de la famille des métaheuristiques. Ce premier chapitre a été l'occasion de présenter le cadre générale des métaheuristiques. Nous constatons que les métaheuristiques représentent un bon outil qui peut être appliquées pour la résolution du problème d'optimisation paramétrique des classifieurs. Dans le chapitre suivant, nous présentons un état de l'art de ce problème..

Chapitre 2

Difficultés d'optimisation paramétrique d'un classifieur neuronal

1 introduction

Il y'a un grand nombre d'articles dans la littérature consacrée à l'analyse et /ou l'utilisation de réseaux de neurones artificiels (RNA). Les RNAs ont connus un grand succès dans le domaine de la reconnaissance des formes et de traitement du signal. Cependant, un problème majeur concernant l'utilisation des réseaux de neurones dans la pratique reste comme un défi. Ce problème est lié à la construction correcte des réseaux de neurones adaptés à un problème spécifique. Ceci est considéré comme une tâche extrêmement importante pour atteindre le succès dans une application qui utilise les RNAs.
Dans ce chapitre, nous présentons quelques notions de bases sur les réseaux de neurones artificiel en particulier le Perceptron Multi-couches Puis, nous présentons la phase d'apprentissage d'un réseau de neurones. Nous nous intéresserons par la suite au problème d'ajustement paramétrique (PAP) des RNAs. Ce chapitre sera terminé par une présentation des quelque travaux qui utilisent les métaheuristiques pour ajuster les paramètres d'un RNA.

2 Historique

En 1943, Mac Culloch et Pitts ont proposé le premier modèle de neurone biologique (Dav1993) [26]. Ce dernier, appelé aussi neurone à seuil, a été inspiré des récentes découvertes en biologie. Ce sont des neurones logiques (0 ou 1). Ces deux physiciens ont montré que les neurones formels peuvent réaliser des fonctions logiques.

En 1949, le psychologue Donald Hebb a introduit le terme connexionnisme pour parler des modèles massivement parallèles et connectés . Il a proposé de nombreuses règles de mise à jour des poids dont la plus connue à cette époque est la "règle de Hebb" [26].

En 1958, le psychologue Frank Rosenblatt a développé le modèle du perceptron [27].Il s'agit d'un réseau de neurones, capable d'apprendre à différencier des formes simples et à calculer certaines fonctions logiques. Il est inspiré du système visuel. Il a réussi à l'appliquer pour la reconnaissance des formes.

Au début des années 60, les travaux de Rosenblatt ont vécu un grand intérêt dans le milieu scientifique. Mais en 1969, deux scientifiques américains appelés Minsky et Papert [26] ont publié un livre dans lequel ont démontré les limites du perceptron proposé par Rosenblatt. En particulier, son incapacité réside à la résolution des problèmes non linéaires.

Les travaux se sont ralentis considérablement jusqu'aux années 80. En 1982, Hopfield a fini par démontrer l'intérêt des réseaux entièrement connectés [26].
Parallèlement, Werbos a conçu un mécanisme d'apprentissage pour les réseaux multicouches de type perceptron : la rétropropagation (Back-Propagation). Cet algorithme qui permet de propager l'erreur vers les couches cachées sera popularisé en 1986 dans un livre "Parallel Distributed Processing" par Rumelhart et al. [28].

3 Réseaux de neurones artificielle

3.1 Définition

Les réseaux de neurones artificiels (RNA) sont des processeurs élémentaires fortement connectés fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit [Dre2002].

D'un point de vue modélisation mathématique, on peut définir un réseau de neurone artificiel par les quatre éléments suivants :

- La nature des entrées et des sorties : Elles peuvent être :« Binaire : (-1 ;+1) (0,1) ou Réelles »
- La fonction d'entrée totale qui définit le prétraitement effectué sur les entrées : Elle peut être : « Booléenne ,Linaire ,Affine, Polynomiale de degré supérieur à deux »
- La fonction d'activation du neurone qui définit son état en fonction de son entrée totale . Il existe plusieurs types des fonctions d'activations tel qu'il est montrer dans les figures 2.1, 2.2, 2.3, 2.4.
- La fonction de sortie qui calcule la sortie du réseau en fonction de son état d'activation.

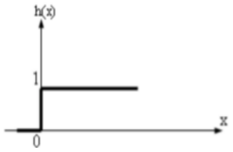

FIGURE 2.1 – Fonction de Heaviside

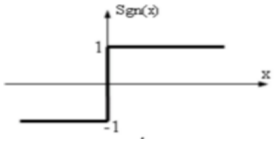

FIGURE 2.2 – Fonction signe

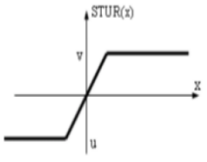

FIGURE 2.3 – Fonction linéaire à seuil

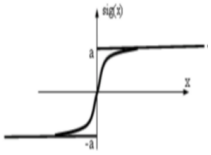

FIGURE 2.4 – Fonction sigmoïde

On distingue principalement deux types de réseaux de neurones : Les réseaux non bouclés et les réseaux bouclés.

3.2 Réseaux de neurones bouclé (ou récurrent)

Un réseau de neurones bouclé est schématisé par un graphe des connexions qui est cyclique. Lorsqu'on se déplace dans le réseau en suivant le sens des connexions, il est possible de trouver au moins un chemin qui revient à son point de départ (un tel chemin est désigné sous le terme de "cycle").
Ainsi, un retard entier multiple de l'unité de temps choisie est attaché à chaque connexion d'un réseau de neurones bouclé (ou à chaque arrête de son graphe). Une grandeur à un instant donné ne peut pas être fonction de sa propre valeur au même instant.
Tout cycle du graphe du réseau doit avoir un retard non nul. La figure 3.1 représente un exemple de réseau de neurones bouclé. Les chiffres dans les carrés indiquent le retard attaché à chaque connexion, exprimé en multiple de l'unité de temps.
Ce réseau contient un cycle, qui part du neurone 3 et revient à celui-ci en passant par le neurone 4. La connexion de 4 vers 3 ayant un retard d'une unité de temps.

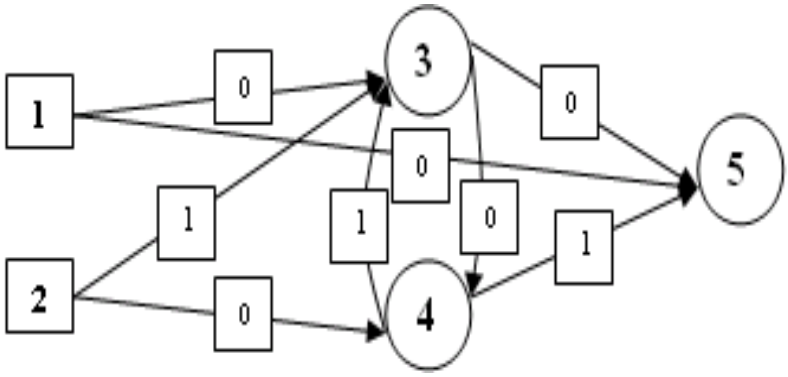

FIGURE 2.5 – Un réseau de neurones bouclé à deux entrées.

3.3 Réseaux de neurones non bouclé

Un réseau de neurones non bouclé est représenté graphiquement par un ensemble de neurones connectés entre eux, l'information circulant des entrées vers les sorties sans "retour en arrière" [29]. On constate que le graphe d'un réseau non bouclé est acyclique comme le montre la figure 2.6.
En effet, si on se déplace dans ce type de réseau à partir d'un neurone quelconque en suivant les connexions, on ne peut pas revenir au neurone de départ.

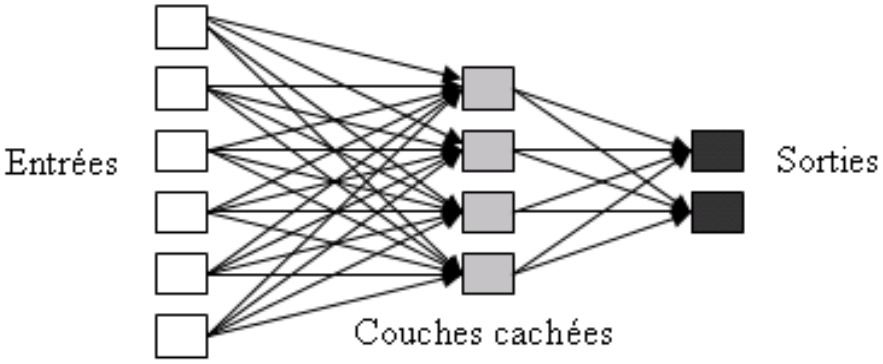

FIGURE 2.6 – Structure d'un réseau de neurones non bouclé.

Les réseaux de neurones non bouclés à couches dont les neurones cachés ont une fonction d'activation sigmoïde, sont souvent appelés des perceptrons multicouche (ou MLP pour Multi-Layer Perceptron) [30].

4 Perceptron Multicouches (PMC)

4.1 Architecture

Le perceptron multicouche (PMC) est un réseau composé de couches successives. Une couche d'entrée lit les signaux entrant, un neurone par entrée x_j , une couche en sortie fournit la réponse du système. Selon les auteurs, la couche d'entrée qui n'introduit aucune modification n'est pas comptabilisée. Une ou plusieurs couches cachées participent au transfert. Un neurone d'une couche cachée est connecté en entrée à chacun des neurones de la couche précédente et en sortie à chaque neurone de la couche suivante.

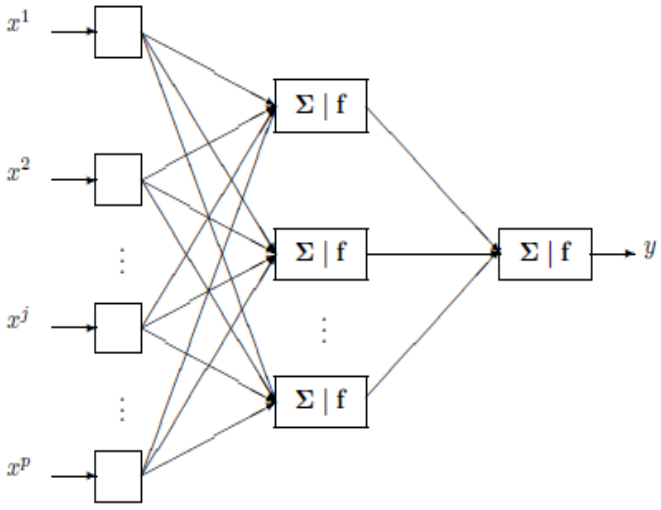

FIGURE 2.7 – : Exemple de perceptron multicouche élémentaire avec une couche cachée et une couche de sortie. .

4.2 Fonction de transfert

- Les neurones d'entrées n'ont pas de fonctions de transferts, leurs états étant imposés par l'extérieurs.
- Les neurones cachées ont des fonctions de transferts sigmoïdes.
- Les neurones de sorties, suivant les applications , ont des fonctions de transfert : sigmoïdes, linéaires, exponentielles ou autres

4.3 Apprentissage

On distingue deux types d'apprentissage : Un apprentissage "supervisé" et un apprentissage "non supervisé" [31].

L'apprentissage supervisé

L'apprentissage "supervisé" pour les réseaux de neurones, consiste à calculer les coefficients synaptiques de telle manière que les sorties du réseau soient, pour les exemples utilisés lors de l'apprentissage, aussi proches que possibles des sorties "désirées". Ils peuvent être la classe d'appartenance de :

- La forme que l'on veut classer,
- La valeur de la fonction que l'on veut approcher,
- La sortie du processus que l'on veut modéliser,
- La sortie souhaitée du processus à commander.

On connaît donc, en tout point ou seulement en quelques points les valeurs que doit avoir la sortie du réseau en fonction des entrées correspondantes : C'est en ce sens que l'apprentissage est "supervisé". Cela signifie qu'un "professeur" peut fournir au réseau des "exemples" de ce que celui-ci doit faire. Ce type d'apprentissage sera adopté par la suite dans notre algorithme d'optimisation.

La plupart des algorithmes d'apprentissage des réseaux de neurones formels sont des algorithmes d'optimisation : Ils cherchent à minimiser par des méthodes d'optimisation non linéaire une fonction coût qui constitue une mesure de l'écart entre les réponses réelles du réseau et ses réponses désirées.

Cette optimisation se fait de manière itérative, en modifiant les poids en fonction du gradient de la fonction coût : Le gradient est estimé par une méthode spécifique aux réseaux de neurones, dite méthode de rétropropagation, puis il est utilisé par l'algorithme d'optimisation proprement dit, les poids sont initialisés aléatoirement avant l'apprentissage, puis modifiés itérativement jusqu'à obtention d'un compromis satisfaisant entre la précision de l'approximation sur l'ensemble d'apprentissage et la précision de l'approximation sur un ensemble de validation disjoint du précédent.

L'apprentissage non supervisé

Un réseau de neurones non bouclé peut être également utilisé dans un but de visualiser ou d'analyser des données : On dispose d'un ensemble de données, représentées par des vecteurs de grande dimension et l'on cherche à les regrouper selon des critères de ressemblance qui sont inconnus à priori. Ce type de tâches est connu en statistique sous le nom de méthodes "d'agrégation".

On peut utiliser les réseaux de neurones non bouclés pour réaliser une tâche assez voisine : A partir des données décrites par des vecteurs de grande dimension, on cherche à trouver une représentation de ces données dans un espace de dimension beaucoup plus faible (typiquement de dimension 2) tout en conservant les "proximités" ou "ressemblances" entre ces derniers. Il n'y a pas là, donc de "professeur", puisque c'est au réseau de découvrir les ressemblances entre les éléments de la base de données, et de les traduire par une proximité dans la "carte" de dimension 2 qu'il doit produire.

4.4 Calcul de l'erreur

Notation et définitions

Si l'on considère un réseau de neurones numérotés de 0 à $n-1$ (soit $v = n-1$) , on définit le vecteur colonne des états, le vecteur potentiels V et la matrice synaptique W. L'état du réseau est défini par le vecteur d'états. Le coefficient w_{ij}est la valeur du poids synaptique associé à la connexion qui lie la sortie du neurone j à une entrée du neurone i. La i^{ieme} ligne de la matrice W contient ainsi tous les coefficients synaptiques du neurone i. La matrice W est une matrice carrée n.n . Si le neurone i n'est pas connecté à la sortie du neurone j, le coefficient synaptique w_{ij}est maintenu à 0.

Il conviendra que toutes les entrées synaptiques x_j d'un neurone i sont connectées aux sorties de chaque neurone j du réseau, ainsi pour tout i, $x_j = \sigma_j$. Le réseau doit pouvoir être sensible à son environnement, il dispose pour cela de neurone d'entrée à entrée particulière dite externe. Une entrée externe n'est pas connectée à la sortie d'un autre neurone. Le réseau agit sur son environnement par l'intermédiaire d'un ensemble de neurones matérialisant ses sorties.

Dans le modèle neuronal que nous allons utiliser, l'entrée externe d'un neurone est non pondérée, elle s'ajoute à ses entrées synaptiques. Le potentiel de ces neurones est alors défini :

$$V_i(t+\tau) = \sum_{j=1}^{n-1}(W_{ij}.\sigma(t)) + li(t) \tag{2.1}$$

Il devrait donc exister deux types de neurones dans un réseau : Ceux disposant d'une entrée externe et les autres, mais cela alourdirait les notations et les expressions analytiques pour peu de choses. Aussi, on considère sauf indication contraire que le potentiel de tout neurone est défini par l'équation (1.1). Lorsqu'un neurone ne dispose pas d'entrée externe, la variable Ii(t) est toujours nulle.

Le fait d'utiliser des notations différentes pour distinguer les neurones de sortie des autres neurones, apporterait aussi des difficultés de notation. Pour cela, on définit un vecteur booléen de n composantes ω_i, chacune étant associée à un neurone, tel que : ω_i= 0 si le neurone i n'est pas un neurone de sortie et ω_i= 1 si le neurone i est un neurone de sortie.

Tout neurone d'un réseau peut en être une entrée ou une sortie du réseau sans qu'il y ait de contrainte. Un neurone qui n'est ni une entrée ni une sortie est un neurone dit caché du réseau.

Il est utile de reconnaître les synapses actives de celles qui ne le sont pas. Par exemple ils peuvent nous en servir pour la reconnaissance des coefficients synaptiques modifiables durant un apprentissage ou alors, d'un point de vue algorithme pour éviter d'effectuer

d'inutiles multiplications par 0 durant l'utilisation du réseau. Dans ce but, on associe généralement à un réseau de neurones un graphe d'adjacence dont les sommets représentent les neurones et les arcs les synapses. Les arcs sont orientés de la sortie d'un neurone vers l'entrée d'un autre.

On définit de même une matrice d'adjacence C, telle que $_{Cij}$ est égale à 1 s'il existe une connexion du neurone pré-synaptique j vers le neurone post-synaptique$i$.

Fonctionnement d'un réseau

Un neurone artificiel fait une somme pondérée des potentiels d'actions qui lui parviennent (chacun de ces potentiels est une valeur numérique qui représente l'état du neurone qui l'a émis), puis s'active suivant la valeur de cette sommation pondérée. Si cette somme dépasse un certain seuil, le neurone est activé et transmet une réponse (sous forme de potentiel d'action) dont la valeur est celle de son activation. Si le neurone n'est pas activé, il ne transmet rien.

En 1943, W. S. Mc Culloch et W. E. Pitts ont proposé un modèle mathématique très simplifié du neurone biologique. Ce modèle, appelé neurone artificiel, est un automate capable de prendre deux états, 0 (inactif) ou 1 (actif). Il dispose d'une sortie correspondant à l'axone et transmettant son état à d'autres neurones artificiels et des entrées pondérées.

Ils correspondent aux dendrites recevant des états qui proviennent d'autres neurones ou des grandeurs provenant des capteurs. Le nombre qui pondère chaque entrée est appelé "coefficient synaptique" (figure 3.3). Soit W_{ij} le coefficient synaptique, ou poids synaptique, correspondant à la synapse transmettant l'activité du neurone$j$ vers le neurone i. Soient x_j (t) les valeurs présentées sur chacune des v entrées du neurone i à l'instant t. Soit θ_i une quantité associée au neurone i appelé "seuil". On définit V_i(t) le potentiel d'action du neurone i à l'instant t de la façon suivante [32] :

$$V_i(t) = \sum_{j=1}^{n} (W_{ij}.x_j(t)) \tag{2.2}$$

L'état σi(t + τ) du neurone i à l'instant $t + \tau$ s'écrit :

$$\sigma i(t + \tau) = 0 si V_i(t) <= f(\theta_i) \tag{2.3}$$

$$\sigma i(t + \tau) = 1 si V_i(t) > f(\theta_i) \tag{2.4}$$

La durée τ est caractéristique du neurone. On considèrera pour ce rapport que la durée est la même pour tous les neurones d'un réseau. L'état σi(t + τ) constitue le signal qui est transmis à d'autres neurones.

Il est commode de traiter le seuil θi comme un coefficient synaptique, la synapse correspondante étant considérée comme connectée à un neurone dont l'état est maintenu à -1.

L'écriture des expressions analytiques s'en trouve souvent légèrement simplifiée. Ce neurone comporte donc v+1 entrées. Supposons que le seuil correspond au coefficient synaptique W_{i0}, on aura : $W_{i0} = \theta_i$ et $x_0 = -1$

On définit alors l'évolution de l'état d'un neurone par les expressions suivantes :

$$V_i(t) = \sum_{j=0}^{v} (W_{ij}.x_j(t)) \tag{2.5}$$

$$\sigma i(t+\tau) = f(V_i(t)) \tag{2.6}$$

Dans le cas des neurones de Mc Culloch et Pitts, la fonction f est l'échelon de Heaviside

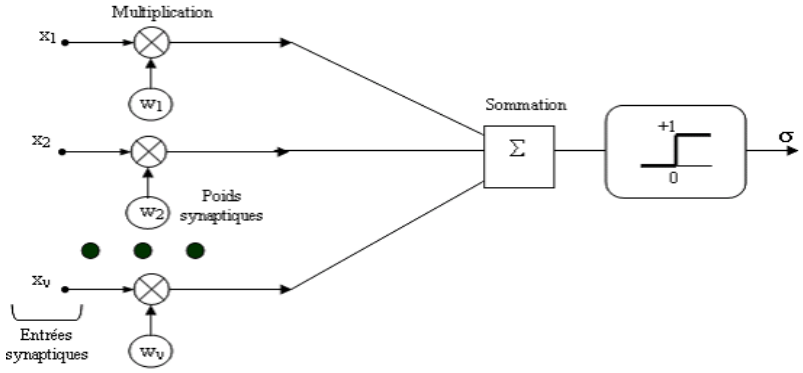

FIGURE 2.8 – Le neurone artificiel de Mc Culloch et pitts.

L'algorithme de rétro-propagation des erreurs

Il s'agit donc dévaluer la dérivée de la fonction coût en une observation et par rapport aux différents paramètres. Soit $Z_{ki} = f(\alpha_{k0} + \alpha_{k'xi})$ et $Z_i = (Z_{li}, \ldots.., Z_{qi})$. Les dérivées partielles de la fonction perte quadratique s'écrivent :

$$\frac{\partial Q_i}{\partial \beta k} = -2(y_i - \phi(x_i))(\beta' z_i) zki = \delta_i zki$$

$$\frac{\partial Q_i}{\partial \beta k} = -2(y_i - \phi(x_i))(\beta' z_i)\beta k f'(\alpha' k.x_i) xip = Skixip$$

Les termes δi et ski sont respectivement les termes d'erreur du modèle courant à la sortie et sur chaque neurone caché. Ces termes d'erreur vérifient les équations dites de rétro-propagation :

$$Sk_i = f'(\alpha' kx_i)\beta_i\delta_i$$

Dont les termes sont évalués en deux passes. Une passe « avant», avec les valeurs courantes des poids, l'application des différentes entrées $x_i$au réseau permet de déterminer les valeurs ajustées ϕ (x_i). La passe « retour» permet ensuite de déterminer les δi qui sont «rétro-propagés» afin de calculer les ski et ainsi obtenir les évaluations des gradients.
L'algorithme suivant montre une structure algorithmique générale de la rétro-propagation.

ALGORITHME II.1 : Apprentissage Par Rétro-Propagation

-Initialisation

Des poids b_{jk} par tirage aléatoire selon une loi uniforme sur [0 ; 1].
Normaliser dans [0 ; 1] les données d'apprentissage.

Tant que *(Q* > err_{max}) Ou (n_{iter} < $iter_{max}$) Faire

Ranger la base d'apprentissage dans un nouvel ordre aléatoire.

Pour chaque élément i = 1,…, n de la base Faire

Calculer $\varepsilon(i) = yi - \emptyset(x_i^1, \ldots, x_i^P$; (b) (i-1)) en propageant les entrées vers l'avant.

L'erreur est "rétro-propagée" dans les différentes couches afin d'affecter à chaque Entrée une responsabilité dans l'erreur globale.
M-à-J de chaque poids $b_{jkl}(i) = b_{jkl}(i - 1) + \Delta b_{jkl}(i)$

Fin Pour

Fin Tant que

4.5 Algorithmes d'optimisation

Sachant évaluer les gradients, différents algorithmes, plus ou moins sophistiqués, sont implémentés. Le plus élémentaire est l'utilisation itérative du gradient : en tout point de l'espace des paramètres, le vecteur gradient de Q pointe dans la direction de l'erreur croissante. Pour faire décroître Q il suffit donc de se déplacer en sens contraire. Il s'agit d'un algorithme itératif modifiant les poids de chaque neurone selon :

$$\beta_k^{(r+1)} = \beta_k^{(r)} - \tau \sum_{i=1}^{n} \frac{\partial Q_i}{\partial \beta k}$$

$$\alpha_{kp}^{(r+1)} = \alpha_{kp}^{(r)} - \tau \sum_{i=1}^{n} \frac{\partial Q_i}{\partial \alpha_{kp}^{(r)}}$$

Le coefficient de proportionnalité τ est appelé : Taux d'Apprentissage. Il peut être fixe (à déterminer par l'utilisateur), ou encore varier en cours d'exécution selon certaines heuristiques.

D'autres méthodes d'optimisation ont été adaptées à l'apprentissage d'un réseau : méthodes du gradient avec second ordre utilisant une approximation itérative de la matrice hessienne (algorithme BFGS, de Levenberg- Marquardt) ou encore une évaluation implicite de cette matrice par la méthode dite du gradient conjugué.
La littérature sur le sujet, propose plusieurs approches, destinées à l'amélioration de la vitesse de convergence de l'algorithme ou bien d'éviter à rester collé à une solution locale défavorable. D'autres techniques de type heuristiques proposent d'ajouter un terme d'inertie afin d'éviter des oscillations de l'algorithme.

5 Problème d'ajustement paramétrique (PAP) des RNAs :

5.1 Définition de problème

Il existe plusieurs paramètres qui doivent être ajustés pour chaque méthode de classification. L'ajustement des paramètres doit être fait avec soins, car ils ont une grande

influence sur l'efficacité et l'effectivité de l'approche de classification. Un ensemble de valeurs optimal et universelle pour les différentes valeurs des paramètres d'une méthode de classification donné est inexistante. Il existe principalement deux stratégies d'ajustement des paramètres :

- Initialisation hors ligne (off-line) : Dans cette stratégie d'initialisation, les valeurs des différents paramètres sont fixées avant l'exécution de l'approche de la classification. Généralement, le développeur de l'approche ajuste un paramètre à la fois, et ces valeurs optimales sont déterminer empiriquement.
- Initialisation en ligne (online) : Dans cette stratégie d'initialisation, les paramètres sont contrôler et mètre a jour durant l'exécution de l'approche de classification. Par exemple, une modification d'une valeur d'un paramètre est accomplie par utilisation de métaheuristique.

Dans ce qui suit, le problème d'ajustement paramétriques des RNAs sera désigné par l'acronyme : ***PAP*** .

5.2 Choix des paramètres

L'ajustement des paramètres d'un réseau de neurones artificiel doit être fait avec soin. Ceci à une grande influence sur la performance du classifieur neuronal. Il existe plusieurs paramètres qui doivent être ajustés :

- **Variables d'entrée / sortie :** leur faire subir comme pour toutes méthodes statistiques, d'éventuelles transformations.
- **Architecture du réseau :** le nombre de couches cachées (en général une ou deux) qui correspond à une aptitude à traiter des problèmes de non-linéarité, le nombre de neurones par couche cachée. Ces deux choix conditionnent directement le nombre de paramètres (de poids) à estimer et donc la complexité du modèle. Le contrôle de la complexité du modèle ou plus généralement d'un sur-apprentissage peut être déterminé en optimisant le nombre de neurones par minimisation d'une estimation de l'erreur de prévision par exemple par validation croisée. Aussi, le choix optimale de l'architecture des RNAs participent à la recherche d'un bon compromis biais/variance c'est-à-dire à l'équilibre entre qualité d'apprentissage et qualité de prévision.
- **Durée de l'apprentissage :** la stratégie la plus simple (SAS Enterprise Miner) consiste à considéré un échantillon indépendant de validation et à arrêter l'apprentissage lorsque l'erreur sur cet échantillon de validation commence à se dégrader tandis que celle sur l'échantillon d'apprentissage ne peut que continuer à décroître.
- Terme de régularisation : Une option importante car efficace pour éviter le sur-apprentissage consiste à introduire un terme de pénalisation ou régularisation, comme en régression ridge, dans le critère à optimiser. Celui-ci devient alors : Q(θ) + $\gamma \, ||\theta||^2$: Le paramètre γ (decay) doit être réglé par l'utilisateur ; plus il est important et moins les paramètres ou poids peuvent prendre des valeurs "chaotiques" contribuant ainsi à limiter les risques de sur-apprentissage. Une stratégie simple et sans doute efficace, surtout si la taille de l'échantillon ne permet pas de prendre en compte une partie validation, consiste à introduire un nombre plutôt grand de neurones puis à optimiser le seul paramètre de régularisation (decay) par validation croisée
- **Autres paramètres :** Il existe d' autres paramètres qui interviennent également sur l'obtention de ce compromis : le nombre maximum d'itérations, l'erreur maximum tolérée et le taux d'apprentissage. En renforçant ces critères, la qualité de l'apprentissage peut etre amélioré, ce qui peut se faire au détriment de celle de la prévision.

En pratique, tous ces paramètres ne peuvent être réglés simultanément par l'utilisateur. Celui-ci est confronté à des choix concernant principalement le contrôle du sur-apprentissage : limiter le nombre de neurones ou la durée d'apprentissage ou encore augmenter le coefficient de pénalisation de la norme des paramètres. Ceci nécessite de déterminer un mode d'estimation de l'erreur : échantillon validation ou test, validation croisée ou bootstrap. Ces choix sont souvent pris par défaut dans la plupart des logiciels commerciaux. Il est important d'en connaître les implications.

5.3 Méthodes utilisées pour la résolution du PAP

– ***Approches manuelle :*** La recherche des paramètres des RNA est généralement effectuée par un développeur (par une procédure d'essai-erreur). Ainsi, l'optimalité ou même quasi-optimalité n'est pas assurée, comme l'espace exploré est seulement une petite partie de la totalité de l'espace de recherche et le type de recherche est assez aléatoire [33] .

  Cette technique de réglage manuel est considérée comme un sous-productive, tâche fastidieuse et source d'erreurs [34], [35], [8]. Lorsque la complexité d'un domaine problème augmente et quand réseaux quasi-optimales sont souhaitées, la recherche manuelle devient plus difficile et ingérable [8]. Un réseau de neurones optimal est un RNA adaptée à un problème spécifique, ayant ainsi une architecture plus petite avec une convergence plus rapide et une meilleure performance de généralisation [34], [35], [8]. Un RNA presque optimale se caractérise par le choix de ses paramètres spécifiques, corrigées pour un problème spécifique, en produisant ainsi une performance satisfaisante.

  La construction de configurations quasi-optimales RNA implique des difficultés comme le nombre exponentiel de paramètres qui doivent être ajustés, la nécessité d'une connaissance a priori du domaine du problème et RNA fonctionnement afin de définir ces paramètres, et la présence d'un expert quand cette connaissance fait défaut.
– ***Approches automatiques :*** Considérant les problèmes et les difficultés liées à l'utilisation de la méthode manuelle pour accorder les paramètres RNA, l'adoption d'une méthode automatique pour effectuer cette tâche d'accord émerge dans le but d'éviter de tels problèmes.

  Un grand nombre de documents se trouvent dans la littérature consacrée à la construction de méthodes automatiques pour ajuster les paramètres RNA. Ceux-ci peuvent être classées en tant que méthodes d'évolution et non évolutifs. Un genre de technique évolutive, l'algorithme génétique (AG) [36], est souvent utilisé pour rechercher des modèles quasi-optimales RNA avec l'optimisation des paramètre RNA . D'autres incluent des fonctions de transfert, poids initiaux et les règles d'apprentissage .

  Les méthodes automatiques qui utilisent des techniques non-évolutives sont axées principalement sur la manipulation d'architectures et de poids RNA.

5.4 Travaux connexes

Dans cette section, nous présentons un tableau II.1 qui montre quelques recherches consacrées à la résolution du PAP lié aux RNAs . La première et quatrième colonnes de ce tableau montrent les problèmes étudier d'un réseau de neurone artificiel et les références des travaux dédiés qui sont extraits de la littérature. La deuxième et la troisième colonne présente les méthodes de résolution adopter ainsi que l'algorithme metaheuristique utilisé.

Problème Etudié	Système physique	Metaheuristique Utilisée	Référence
Estimation des paramètres		Firefly Algorithm with Predation (FAP) A l'aide du code BRAMS	(Eduardo2013) [37]
Optimisation Structurelle d'un RNA	* Proposition d'une approche automatisée pour optimiser l'architecture des RNAs.* Estimation de la complexité (computational) des RNAs .	Plusieurs métaheuristiques telles que RVV, SA, GA ...	(Carvalho 2010) [38]
la configuration automatique pour les deux types de réseaux de neurones (PMC) et récurrent Elman (sont des réseaux de neurones)	Les réseaux auto-configuration sont appliquées pour effectuer assimilation de données.	AAM (Multiple particules algorithme de collision)	(Juli 2014) [39]
combinaisons entre RNA et AE	utiliser (AE) pour évoluer des poids de connexion RNA, des architectures, l'apprentissage des règles, et les caractéristiques d'entrée	AE(algorithmes évolutionnaires)	(Xin Yao1999) [40]
réduire le temps nécessaire pour effectuer la recherche des paramètres RNA pour un problème spécifique(paramètres de l'algorithme d'apprentissage, le nombre de couches cachées et les nœuds cachés par couche les fonctions de transfert et poids initiaux)	les résultats montre que cette méthode nécessite seulement 15 minutes pour effectuer la recherche des paramètres RNA contre 33,63 heures pour son antécesseur(NNGA-DCOD (RNA + AG- encoder direct))	GANNTune(est constitué d'un GA travaillant directement avec des paramètres RNA)	(Leandro 2008) [41].
Reglage parametrique d'un PMC		AG	(Meng09) [42]

TABLE 2.1 – Principaux travaux qui utilisent les métaheuristiques pour la résolution du PAP .

5.5 conclusion

Dans ce chapitre nous avons détaillé le problème d'optimisation paramétrique d'un classifieur neuronal. Au début, nous avons exposé les principes et les propriétés fondamentales des réseaux de neurones artificielles, en particulier le Perceptron Multi-couches (PMC). Nous avons définit aussi le problème PAP auquel nous nous s'intéressons dans ce mémoire de master. En suite, nous avons présenté les principaux paramètres affectant les classifieurs neuronales. Finalement, nous avons donné un aperçu sur les principaux travaux publiés qui utilisent les métaheuristiques comme approches d'optimisation pour ajuster les paramètres d'un RNA. A la base de ce qui a été montré dans ces deux premiers chapitres, nous entamons la partie développement d'un algorithme d'optimisation. Cet algorithme sera basé sur ces outils qui feront l'objet d'un troisième chapitre.

Chapitre 3

Implémentation de l'algorithme OP-RNA

1 Introduction

La mise en œuvre en pratique c'est le moyen le plus efficace pour caractériser et acquérir un concept théorique. Notre cadre de travail dès le début se base sur l'élaboration d'un système automatique capable d'ajuster les paramètres d'un classifieur neuronel, En parcourant l'état de l'art nous avons choisis l'algorithme génétique AG et le récuit simuler RS pour améliorer la robustesse de notre classifieur (amélioration paramétrique).
Notre travail comprend deux phases :

- Phase de réalisation de notre classifieur neuronal.
- Phase d'amélioration paramétrique par métaheuristique (algorithme génétique et recuit simulé).

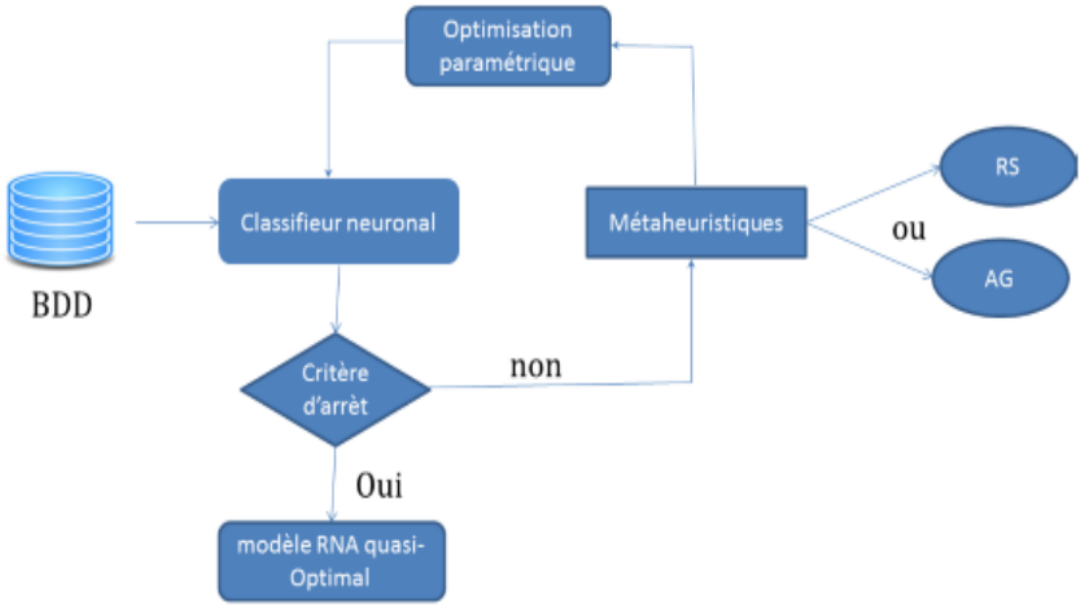

FIGURE 3.1 – Schéma représentatif de notre procédure.

2 Description du jeu de données

Dans ce mémoire de master nous utilisons trois bases de données médicales. Ces bases de données sont souvent utilisés par les méthodes de ''Data Mining'' pour analyser et valider ces approches. Les principales caractéristiques de ces ensembles de bases sont représentés dans le tableau suivant :

Bases de données	Nbr d'attributs	Nbr de cas	cas positifs	cas négatifs	Taille-Base d'apprentissage	Taille-Base detest
PID (Diabete)	8	768	286	500	576	192
Troubles Hépatiques	6	345	145	200	276	69
Appendicite	7	106	85	21	85	21

TABLE 3.1 – Les trois bases de données utilisées dans cette étude .

2.1 Description de la base de données de diabète

Dans ce mémoire de Master nous utilisons la base de données médicale réelle (Indians Diabetes Pima).L'ensemble de données a été choisi du dépôt d'UCI [43] qui réalise une étude sur 768 femmes Indiennes Pima, Ces mêmes femmes, qui ont stoppé leurs migrations en Arizona, États-Unis, adoptant un mode de vie occidentalisé, développent un diabète dans presque 50% des cas. [44] Dans les 768 cas présentés Il y a 500 échantillons provenant des patients qui n'ont pas le diabète et 268 échantillons provenant des patients qui sont connus d'avoir le diabète. Le diagnostic est une valeur binaire variable «classe» qui permet de savoir si le patient montre des signes de diabète selon les critères de l'Organisation Mondiale de la Santé. Les huit descripteurs cliniques sont :

- 1. Npreg : nombre de grossesses,
- 2. Glu : concentration du glucose plasmatique,
- 3. BP : tension artérielle diastolique,
- 4. SKIN : épaisseur de pli de peau du triceps,
- 5. Insuline : dose d'insuline,
- 6. BMI : index de masse corporelle,
- 7. PED : fonction de pedigree de diabète (l'héridité).
- 8. Age : âge

2.2 Description de la base de données du Troubles hépatiques

Cette base provient du centre de recherche médical BUPA : British United Provident Association (Richard S. Forsyth, Mapperley Park, Nottingham, UK). Elle est composée de 345 patients dont 145 cas présentent un diagnostic positif et 200 cas présentent un diagnostic négatif. Un diagnostic indique si une personne est atteinte d'une hépatite ou pas .Les principales caractéristiques de cette base de données sont présentées dans le tableau suivant :

Attributs	Description
VGM	Volume globulaire moyen
Alkphos	Alkaline phosphatase
SGPT	Sérum glutamo pyruvique transaminase
SGOT	Sérum glutamo oxalacétique transaminase
Gammagt	Gamma glutamyl transpeptidases
Drinks	Consommation alcoolique (number of half-pinf equivalents of alcoholic beverages drunk per day)

TABLE 3.2 – Description des attributs de la base T.H (Troubles hépatiques) .

Les cinq premiers descripteurs correspondent à des tests sanguins donnant la sensibilité des troubles du foie.

2.3 Description de la base de données d'Appendicite

Appendicite (Appendicitis) . Cette base de données crée par Shalom Weiss de l'université Rutgers est composée de 106 exemples avec sept descripteurs numériques décrit dans le tableau III.4. qui représente des tests de laboratoires .Ces exemples sont répartie en deux classe : 80.2% des données appartient a la clasSe '1' (diagnostic positive) et 19.8% appartient a la classe '2' (diagnostic négative).

Attributs	Description
1	WBC1
2	MNEP
3	MNEA
4	MBAP
5	MBAA
6	HNEP
7	HNEA

TABLE 3.3 – Description des attributs de la base AP (Appendicite) [6].

3 Environnement de développement

Notre choix du langage java a été guidé par les avantages qu'offre la programmation orientée objet. Le langage Java examine le programme au fil de l'exécution et libère la mémoire automatiquement. Cette fonctionnalité diminue les reprises de panne du programme et ne laisse pas la possibilité d'une mauvaise utilisation de la mémoire.

Notre application est réalisée sous Netbeans version 8.0 sous windows. Le choix de la programmation sous windows a été pris à cause de l'interface graphique qu'offre cet environnement.

Le choix de Netbeans est fondamental puisque c'est un logiciel standard de développement utilisé pour la création des interfaces très puissantes. Par contre les phases de calcul sont réalisées sous Matlab. Notons que Matlab et Netbeans sont reliés par la bibliothèque JMI.

4 Critères d'évaluation

Les performances de classification des données ont été évaluées par le calcul des vrais positifs (VP),vrais négatifs (VN),faux positifs (FP), et faux négatifs (FN), le pourcentage de sensibilité SE,la spécificité (SP),et le taux de classification (TC),leurs définitions respectives sont les suivant :

- VP : malade classé malade.
- VN :non malade classé non malade.
- FP : non malade classé malade.
- FN : malade classé non malade.
- **Sensibilité (SE%) :** [SE = 100 * VP / (VP + FN)] la sensibilité (Se) du test est sa capacité de donner un résultat positif quand la maladie est présente.
- **Spécificité (SP %) :** [SP = 100* VN / (VN + FP)] la spécificité du test est cette capacité de donner un résultat négatif quand la maladie est absente.
- **Taux de classification (TC%) :** [TC = 100 * (VP + VN) / (VN + VP + FN +FP)] est le pourcentage des exemples correctement classées.

Dans ce travail, nous traitons un problème d'optimisation combinatoire complexe où nous sommes intéressés à trouver les paramètres qui montrant une meilleure performance avec moins de surcharge de calcul possible. Pour cela, nous avons construit une fonction objectif qui consiste en une combinaison de deux critères pour l'erreur et une sanction qui peut être exprimé par :

$$F_{obj} = Penalitex * (\alpha_1 * Eapp + \alpha_2 * Egen)/(\alpha_1 + \alpha_2)$$

[38]
où α_1 et α_2 (α_1,α_2 >0 ; α_1,α_2 ϵ R) sont des facteurs d'ajustement utilisés pour mesurer le degré d'importance de l'erreurs de d'apprentissage et de généralisation, respectivement. pour ce travail, nous utilisons trois ensembles de valeurs pour α_1, α_2

Exp	α_1	α_2
♯1	1.0	0.1
♯2	1.0	1.0
♯3	0.1	1.0

TABLE 3.4 – Les valeurs de α_1, α_2 appliquées dans chacune des expériences effectuées dans ce travail

La fonction f_{obj} est la fonction dont la valeur doit être minimisée par algorithmes mis en œuvre. Ainsi, il devient clair que f_{obj} est composé par la somme des erreurs dans les phases d'apprentissage et de généralisation multiplié par une pénalité (complexité de réseau)

Dans ce travail, nous supposons que l'expression globale de la pénalité est venue de la complexité du modèle où x est le nombre de poids dans les connexions et y le nombre d'itération nécessaires à la formation du réseau :

$$\text{Pénalité} = P1 + P2$$

$$\text{Pénalité} = \underbrace{1 * 10^{-3} * x}_{P1} + \underbrace{10^{-2} * y + 1}_{P2}$$

où x est le nombre de poids dans les connexions, et y le nombre d'itération nécessaires à la formation du réseau.

5 Le modèle OP-RNA

5.1 Expérimentation1 (construction de notre classifieur neuronal)

Architecture du modèle et choix des paramètres

Dans la premiére expérimentation nous avons utiliser un perceptron multicouche avec la topologie [8 :6 :1] et [8 :6 :3 :12] qui a comme entrée les paramètre des base (PIMA,TH ,et AP). avec les paramètres par défaut suivant : un pas d'apprentissage=0.5 , maximum d'itération = 1000, une fonction d'activation = Log-sigmoïde(ou Hyperbolique tangent sigmoïde),et une algorithme d'apprentissage : Gradient descente(ou Levenberg-Marquardt). Nous utilisons aussi trois ensembles de valeurs pour α_1, α_2.

Résultats et discussion

L'ensemble de données a été divisé aléatoirement dans deux sous-ensembles apprentissage 75% et test 25%.

BDD	Arch	α_1, α_2	E_{app}	E_{test}	Se	Sp	Tc	F_{obj}
Pima	[8 6 1]	[1 0.1]	0.18109	0.17713	0.8278	0.6285	0.7552	0.4251069
		[1 1]	0.17702	0.15845	0.8606	0.700	0.8020	0.5693042
		[0.1 1]	0.1508	0.1442	0.816	0.803	0.8125	1.065004
	[8 6 3 12]	[1 0.1]	0.1575	0.14757	0.8360	0.7142	0.7916	0.2194557
		[1 1]	0.16115	0.14504	0.9180	0.600	0.8020	0.1843072
		[0.1 1]	0.160123	0.13726	0.9098	0.6571	0.8177	0.1907581
TH	[8 6 1]	[1 0.1]	0.207905	0.18923	0.875	0.600	0.7790	0.715042
		[1 1]	0.17713	0.18438	0.9464	0.4666	0.7790	0.2213846
		[0.1 1]	0.2483	0.2399	0.896	0.333	0.721	1.2827797
	[8 6 3 12]	[1 0.1]	0.2027	0.2141	0.9285	0.400	0.7441	0.3256212
		[1 1]	0.14713	0.15175	0.9107	0.700	0.8372	0.2024915
		[0.1 1]	0.20610	0.21374	0.9464	0.2666	0.7093	0.3575030
AP	[8 6 1]	[1 0.1]	0.172025	0.201182	0.7142	0.7368	0.7307	0.2293044
		[1 1]	0.08863	0.16506	0.5714	0.8974	0.8076	0.1921137
		[0.1 1]	0.09	0.1751	0.55	0.88	0.769	0.3268484
	[8 6 3 12]	[1 0.1]	0.252541	0.240998	0.4285	0.9473	0.8076	0.2948144
		[1 1]	0.06478	0.20801	0.5714	0.9473	0.8461	0.2433484
		[0.1 1]	0.12276	0.174005	0.5714	0.8947	0.8076	0.2181190

TABLE 3.5 – Résultat pour la première expérimentation

-Les résultats de cette expérimentation pour la base de donnée PIMA montre que la spécificité du système est élevé ce qui veut dire que le système a fait un bon apprentissage pour les données négative. Donc lorsqu'un patient est non diabétique notre modèle le détecte avec succès. Même chose pour la sensibilité, ce qui veut dire que le système a fait une bonne reconnaissance des données positives. Donc les beaucoup de patient diabétique ont détecte aussi avec succès. Avec ces performances, nous pouvons dire que le modèle a donné un bon taux de classification.

-Les résultats de cette expérimentation pour la base de donnée TH montre que la sensibilité du système est élevé ce qui veut dire que le système a fait un bon apprentissage pour les données positive. Donc lorsqu'un patient est malade notre modèle le détecte avec succès. Par contre la spécificité du système est faible ce qui veut dire que le système a fait une mauvaise reconnaissance des données négative. Donc beaucoup de patient non malade ont été reconnu comme malade. Ce qui peut générer un danger pour la santé du patient. Avec ces performances, nous pouvons dire que le modèle a donné un taux de classification moyen et une bonne sensibilité. Par contre il a donné une faible spécificité. Ce qui reste un inconvénient a étudié.

- Les résultats de cette expérimentation pour la base de donnée AP montre que la sensibilité du système est moyen ce qui veut dire que le système a fait une moyen reconnaissance des données positives. Donc la moyenne des patients malades ont été reconnu comme non malade. Par contre la spécificité du système est élevé ce qui veut dire que le système a fait une bonne reconnaissance des données négatives. Donc beaucoup de patient non malade ont été détecte avec succès. Avec ces performances, nous pouvons dire que le modèle a donné un taux de classification acceptable et une bonne spécificité et une sensibilité moyenne. Ce qui reste un inconvénient a étudié.

-une autre remarque c'est que : lorsqu'on donne l'importance a l'erreure de teste ($\alpha 1 < \alpha 2$) le taux de classification augmente.

D'après les résultats obtenus nous avons observé qu'il faut établir une procédure d'optimisation qui permet de choisir les paramètres les plus pertinents pour avoir un bon modèle de classification.

Diverses approches ont été employées pour optimiser les paramètres de RNA. L'algorithme génétique et le recuite simulé reste parmi les techniques les plus utilisées.

Dans la partie suivante nous allons proposer une approche qui utilise ces technique pour optimiser les paramètres les plus pertinents avec une amélioration de la robustesse de notre classifieur neuronal.

5.2 Expérimentation2 (Amélioration par métaheuristique)

Dans le but d'améliorer la performance de l'expérimentation précédente, nous avons appliqué deux métaheuristiques (l'Algorithme génétique AG et Recuit simulé RS) sur les même architectures de l'expérimentation 1 [8 :6 :1] et [8 :6 :3 :12].

Amélioration par Algorithme Génétique

Nous allons établir ou concevoir un algorithme génétique ayant comme but d'optimiser les paramètres suivant : fonction d'activation, fonction d'apprentissage, pas d'apprentissage, le moment, et les poids de connexion. Les valeurs possibles pour les paramètres sont indiqué dans le tableau ci-dessous :

Paramètre	Valeur
fonction d'activation	Tanh Hiperbolic tangent, log-sigmoid, lineaire
fonction d'apprentissage	descent Gradient, Levenberg-Marquardt, Resilient, BFGS quasi newton
pas d'apprentissage	[0 ,1]
le moment	[0 ,1]
les poids de connexion	$[\frac{-1}{\sqrt{N}}, \frac{+1}{\sqrt{N}}]$ avec N : nombre des exemples

TABLE 3.6 – valeurs possibles pour les paramètres

L'algorithme hybride proposé est représenté par le diagramme de la Figure 3.2. Ce modèle représenté un algorithme d'apprentissage de PMC hybride avec l'AG afin d'optimiser les paramètres du réseau. Tous les paramètres du réseau sont codés pour former un long chromosome et régler par l'AG. Puis, à la suite du processus GA, l'algorithme BP est utilisé pour former le réseau. La procédure de l'hybride BP algorithme d'apprentissage est présentée comme suit :

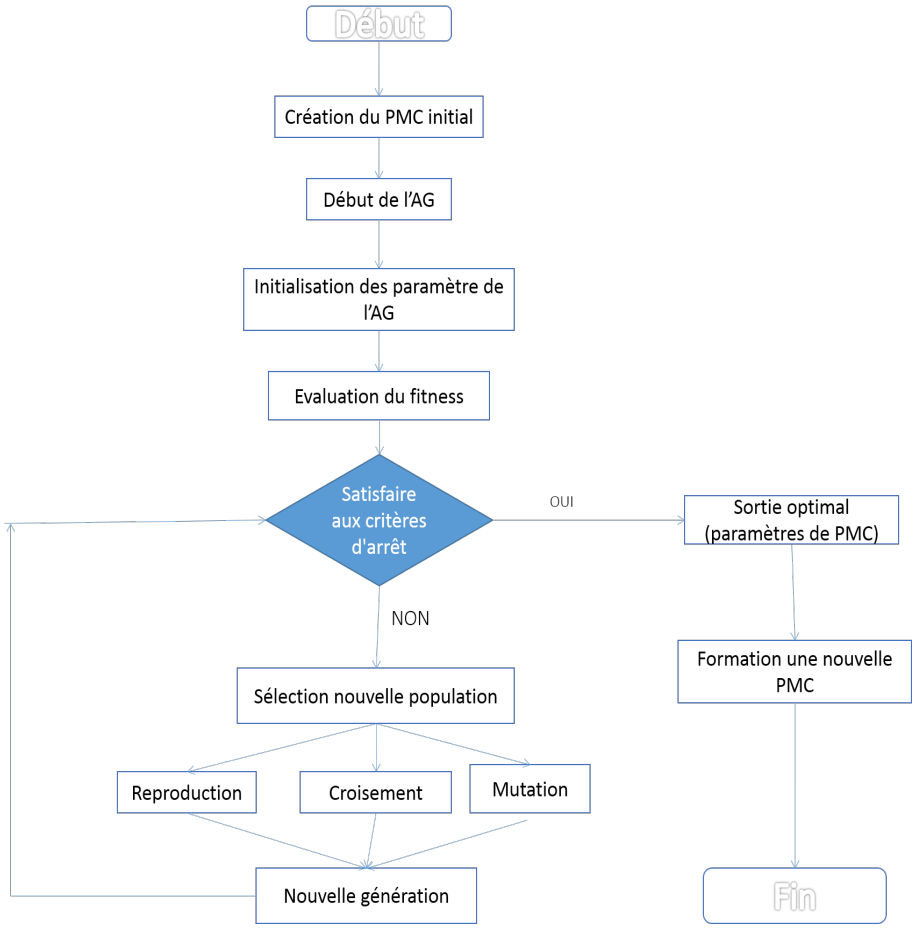

FIGURE 3.2 – organigramme de l'algorithme d'apprentissage proposé.

codage

– ***La représentation des chromosomes*** Tous les paramètres sont mémorisés par un vecteur ligne C =(C_i) , i = 1,2,..N : où N est le nombre de tous les paramètres de RN. Nous pouvons écrire le premier chromosome comme suit :

Fct d'activation	Fct d'apprentissage	Pas d'app	Moment	P= [$W_1, W_2, W_n, B_1, B_2, B_m$]
2 bits	2 bits	6 bits	4 bits	n bits

Tel que le premier et le deuxième groupe de deux bits correspondent à la fonction d'activation et la fonction d'apprentissage utilisé dans le réseau, respectivement. Nous représentons le pas d'apprentissage (6 bits) dans la troisième. Le quatrième groupe de 4 bits est en relation avec le moment.

Le dernier groupe est écrite comme suit : P= [$W_1, W_2, W_n, B_1, B_2, B_m$] où W_1 désigne les poids de connexion entre la couche d'entrée et la première couche cachée, W_2 est les poids de connexion entre la première couche caché et la deuxième couche cachée. Et n le nombre de couche caché choisi.
B_1= [b_1,b_2,. . . .b_k] :sont les biais de neurones pour la premier couche cachée.
B_m =[b_1,b_2,. . . .b_k] :sont les biais de neurones pour la dernier couche cachée.

– ***Fonction de fitness*** La fonction de fitness dépend de problème et est utilisé pour évaluer la performance de chaque individu. Dans ce travaille nous utilisons comme fonction de fitness la fonction objective. $F_{fitness} = F_{objectif}$

Opérateurs génétiques

– ***Sélection*** Opérateur de sélection consiste à sélectionner les individus de la population pour la reproduction sur la base de la valeur de fitness relative de chaque individu. L'extraction peut être réalisée de plusieurs façons. Dans ce travail nous utilisons la fonction de sélection « par tournois », Le principe de la sélection par tournoi consiste à choisir k individus aléatoirement, l'individu sélectionné est le vainqueur du tournoi c'est-à-dire celui qui possède la meilleur performance, donc il y aurait autant de tournoi que d'individu a sélectionné.

– ***Croisement*** Pour le croisement nous avons choisis le croisement à un point tel que nous affectons à Chaque individu la probabilité de croisement généralement supérieur à 70%. Alors pour qu'un individu soit sélectionné pour le croisement, nous générons une valeur aléatoire β dans [0,1], que nous allons comparer à la probabilité de croisement. - Si $\beta <$ Pc, (Pc : probabilité de croisement), alors l'individu est sélectionné pour le croisement, sinon il passe directement à la population qui participe à la mutation. Pour chaque pair des éléments sélectionnés nous définissons de façon aléatoire le point de croisement h (nous utilisons le croisement "un point").

– ***Mutation*** Après croisement, les nouveaux individus sont soumis à une mutation. La mutation empêche l'algorithme d'être pris au piège dans un minimum local. Une variable est choisi avec une certaine probabilité et sa valeur est modifiée par une valeur aléatoire. Ici, nous choisissons la méthode de mutation non-uniforme. Mutation non-uniforme modifie l'un des gènes du parent basé sur une distribution de probabilité non-uniforme.

– ***Test d'arrêt*** Le test d'arrêt d'une manière générale est défini selon le type de problème étudié (heuristique), dans notre cas nous allons utiliser comme test le nombre d'itération pour gagner dans le temps de calcul.

résultats obtenu Après les tests sur trois banques de donnée nous avons obtenu le résultat suivant

BDD	Arch	α_1, α_2	Paramètre optimale				E_{app}	E_{test}	Tc	F_{obj}
			Fct act	Fct app	Pas	Mom				
Pima	8 6 1	[0.1 1]	purelin	trainbfg	0.4375	0.3125	0.1757	0.1636	0.8372	3,9593
		[1 1]	tansig	trainlm	0.6875	0.6875	0.1490	0.1344	0.8177	1,1157
		[1 0.1]	purelin	trainlm	0.8281	0.9375	0.1578	0.1468	0,4756	0,8125
	8 6 3 12	[0.1 1]	tansig	trainlm	0.8281	0.6875	0.1567	0.1321	0.8333	1.1044
		[1 1]	tansig	trainlm	0.3125	0.0625	0.1475	0.1403	0.8125	1.0179
		[1 0.1]	purelin	trainrp	0.2656	0.75	0.1602	0.1374	0.8177	2.6810
TH	8 6 1	[0.1 1]	tansig	trainlm	0.1406	0.125	0.1812	0.2000	0.7558	2.0269
		[1 1]	tansig	trainbfg	0.4531	0.4375	0.2100	0.1944	0.7906	3.9593
		[1 0.1]	tansig	traingd	0.8437	0.1875	0.2143	0.2070	0.7441	34.533
	8 6 3 12	[0.1 1]	purelin	trainrp	0.2812	0.0625	0.2198	0.2071	0.6976	5.4613
		[1 1]	tansig	trainrp	0.75	0.5625	0.2476	0.2438	0.7209	1.7626
		[1 0.1]	purelin	trainbfg	0.6718	0.8125	0.0992	0.1678	0.8076	0.6818
AP	8 6 1	[0.1 1]	purelin	trainbfg	0.6718	0.8125	0.0992	0.1678	0.8076	0.6818
		[1 1]	purelin	trainbfg	0.6406	0.8125	0.1287	0.1770	0.80769	0.21026
		[1 0.1]	purelin	trainrp	0.8281	0.375	0.0968	0.1610	0.3441	0.8076
	8 6 3 12	[0.1 1]	purelin	trainbfg	0.625	0.0.25	0.0625	0.1520	0.8461	1.4765
		[1 1]	purelin	trainbfg	0.0027	0.2279	0.1214	0.157	0.8076	0.4955
		[1 0.1]	logsig	trainlm	0.6406	0.625	0.1346	0.1891	0.5928	0.8076

TABLE 3.7 – Résultat pour la deuxième expérimentation (AG)

Amélioration par Recuite simulé

Dans le cas du recuit simulé le réglage des paramètres n'est pas évident. Les paramètres choisis pour le recuit simulé sont :

- Température initiale : Si la température initiale est trop élevée, le début de la recherche ne sert à rien. Donc nous choisissons la température initial T0 (par default) : T0=100.
- Fonction de recuit :Annealing Boltz
- Détermination de la probabilité P :La probabilité P d'accepter X* quand cette solution est moins bonne que la solution courante décrois avec le temps de l'algorithme, cette probabilité P est une fonction P(T,δ F) dépendant d'un paramètre T, appelé température, par analogie avec le recuit métallurgique, et de la dégradation de l'objectif δ F=F(X*)-F(Xn). La forme de la loi de probabilité P est dictée par la distribution de Boltzmann P (T, δ F)= e-(δ F/T) et elle est comprise entre 0 et 1 comme une probabilité.
- Critère d'arrêt : Le critère d'arrêt est le nombre maximal d'itérations : Max_{iter}=100

Nous présentons dans le tableau suivant les résultats de l'hybridation de recuit simulé RS et le Perceptron Multi-couches PMC :

BDD	Arch	α_1, α_2	Paramètre optimale				E_{app}	E_{test}	Tc	F_{obj}
			Fct act	Fct app	Pas	Mom				
Pima	8 6 1	[0.1 1]	tansig	trainlm	0.3726	0.8482	0.1485	0.1361	0.8281	0.9907
		[1 1]	tansig	trainlm	0.7068	0.7375	0.1490	0.1344	0.8177	1,1157
		[1 0.1]	tansig	trainrp	0.8537	0.2875	0.2546	0.1186	0,8092	4.533
	8 6 3 12	[0.1 1]	**tansig**	**trainlm**	**0.6574**	**0.8721**	0.1510	0.1329	0.8333	0.8371
		[1 1]	tansig	trainlm	0.3425	0.0831	0.1376	0.1403	0.8125	1.0179
		[1 0.1]	purelin	trainrp	0.2642	0.7930	0.1633	0.1374	0.8263	1.7451
TH	8 6 1	[0.1 1]	logsig	trainlm	0.0975	0.0016	0.1518	0.1709	0.8837	3.4205
		[1 1]	tansig	trainbfg	0.4531	0.4375	0.1644	0.1880	0.8109	1.8482
		[1 0.1]	tansig	trainlm	0.3564	0.1875	0.1755	0.1912	0.7441	2.533
	8 6 3 12	[0.1 1]	**tansig**	**trainbfg**	**0.9331**	**0.7679**	0.2615	0.2173	0.7674	0.2695
		[1 1]	tansig	trainrp	0.6963	0.8106	0.2376	0.2486	0.7209	1.7626
		[1 0.1]	purelin	trainbfg	0.1406	0.3152	0.2032	0.2155	0.7906	0.6370
AP	8 6 1	[0.1 1]	**purelin**	**trainbfg**	**0.0027**	**0.2279**	0.1214	0.1572	0.8461	0.4955
		[1 1]	purelin	trainbfg	0.7216	0.7855	0.1389	0.1670	0.8381	0.3256
		[1 0.1]	logsig	trainrp	0.8612	0.4560	0.1241	0.1320	0.8544	0.9454
	8 6 3 12	[0.1 1]	**purelin**	**trainbfg**	**0.0051**	**0.0513**	0.1079	0.1594	**0.8846**	0.3432
		[1 1]	purelin	trainrp	0.0043	0.1645	0.1214	0.157	0.8045	0.6541
		[1 0.1]	logsig	trainbfg	0.0187	0.465	0.1346	0.1645	0.8901	0.6076

TABLE 3.8 – Résultat pour la deuxième expérimentation (RS)

6 Aperçue sur l'application

Les figures suivantes montrent un schéma général de notre application (OP-RNA). La première fenêtre qui s'affiche si on exécute l'application monte dans la figure 3.3

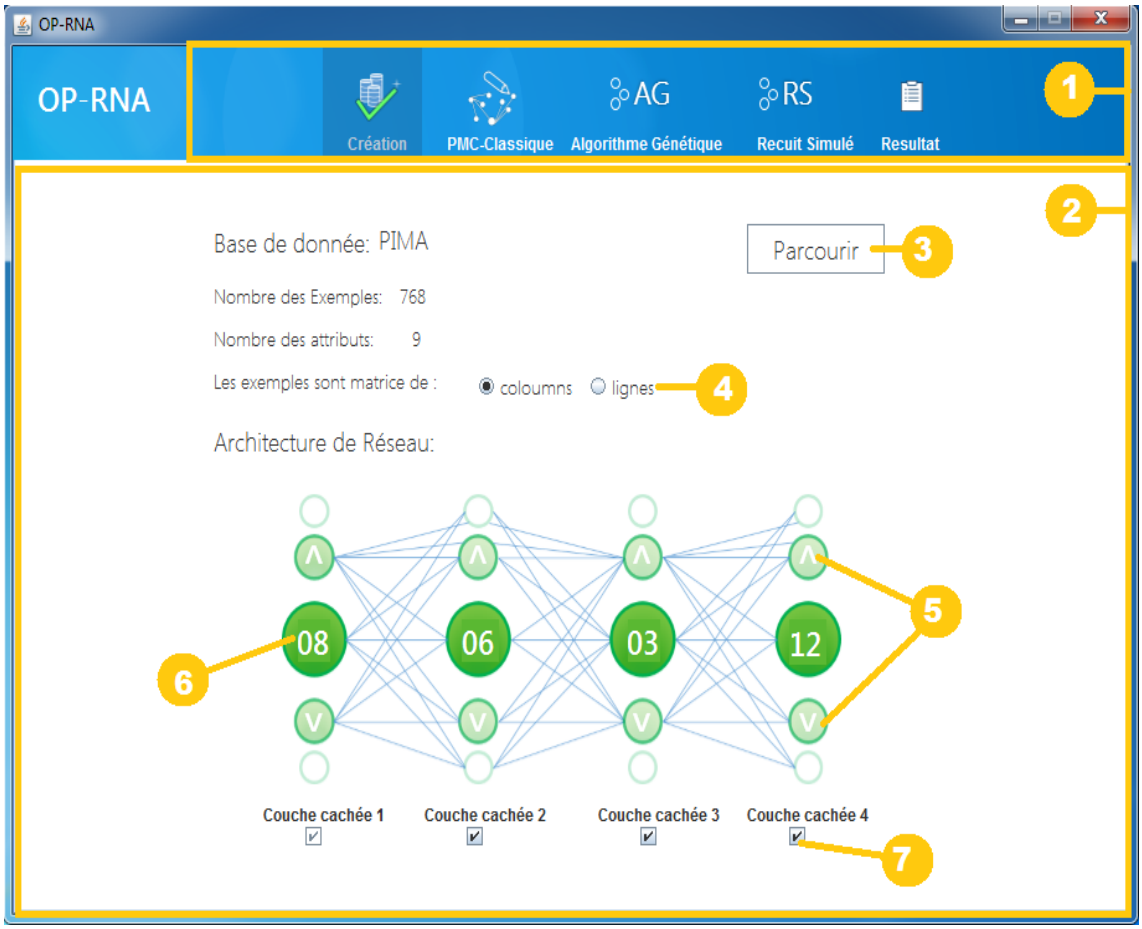

FIGURE 3.3 – Interface Principale du OP-RNA.

- 1 : Barre de menu : contient les outils suivant :(de gauche à adroite respectivement)
  - Création : permet de charger la base de données et construire la topologie du réseau de neurone.
  - PMC-classique : permet de saisir manuellement les paramètres spécifiés pour le perceptron multicouches.
  - Algorithme Génétique : permet de sélectionner les paramètres d'AG afin de construire un perceptron multicouches amélioré.
  - Recuit simulé : permet de sélectionner les paramètres de RS afin de construire un perceptron multicouches amélioré.
  - Résultat : permet de voir historique des résultats des expériences, et de tester des cas nouveaux en utilisant des modèles déjà enregistrés.
- 2 : Fenêtre d'édition permet d'affiche ensemble des objets de chaque outil sélectionné pour l'édition.

- 3 : permet de chargement de la base de donnée.

- 4 : permet de sélectionner de quelle façon les individus de la base sont arrangés.

- 5 : permet d'augmenter ou diminuer le nombre de neurone dans chaque couche cachée.

– 6 : permet saisir le nombre de neurone dans chaque couche cachée immédiatement.

– 7 : permet d'activer ou d'annuler la couche cachée.

La figure3.4, affiche la fenêtre de création Modèle PMC-classique ,elle montre les différents champs des paramètres de perceptron multicouche, ainsi une Botton "Réinitialiser" pour étalier des paramètres par défaut, et une Botton vert "Lancer" pour lancer l'exécution de ce modèle.

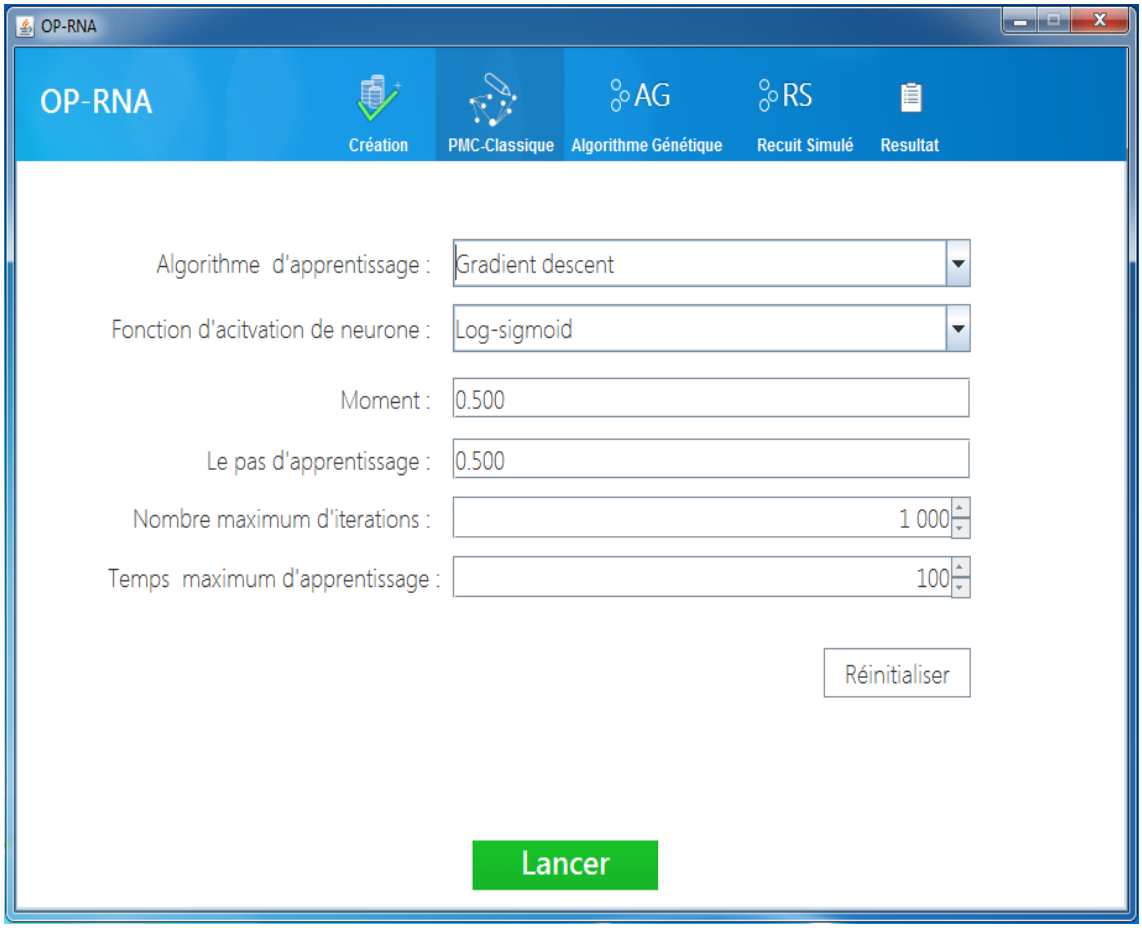

FIGURE 3.4 – Fenêtre de création Modèle PMC-classique.

La figure 3.5, affiche la fenêtre de création Modèle PMC+AG, elle montre les différents champs des paramètres de l'AG. L'option facultative "Optimiser la base" permet de réduire nombre des exemples de la base entrée de telle façon pour améliorer la performance de système (éliminer les données redondants).

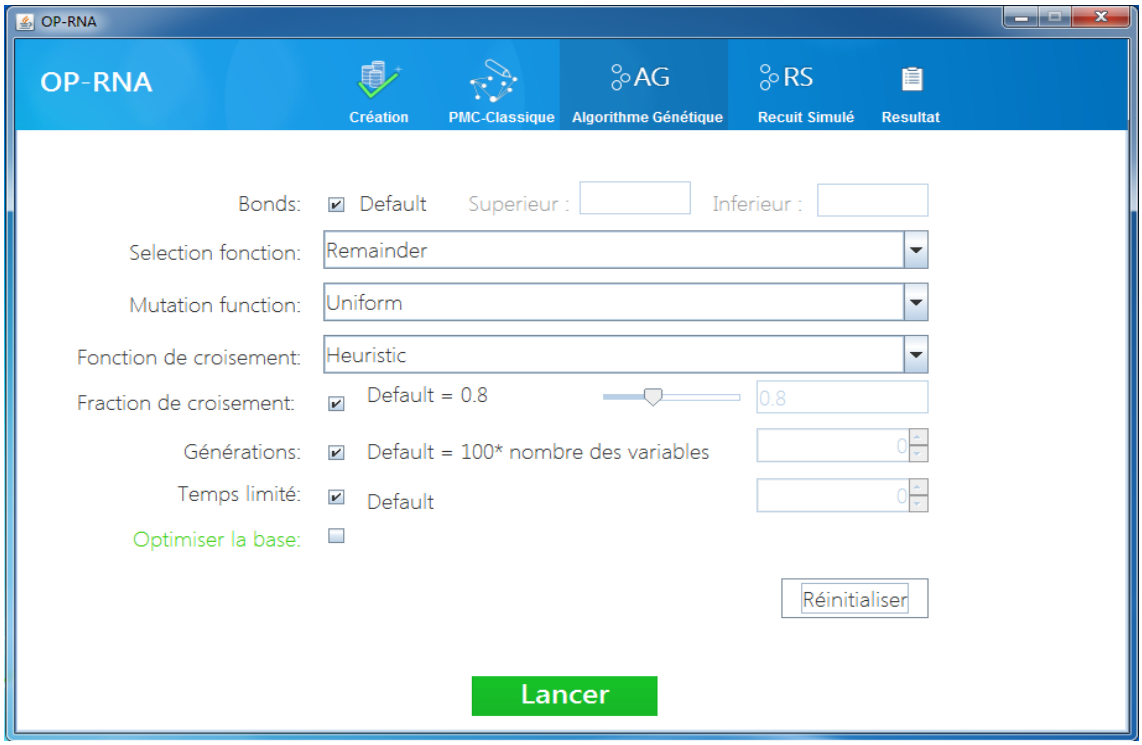

FIGURE 3.5 – Fenêtre de création Modèle PMC+AG.

La figure suivante3.6, affiche la fenêtre de création Modèle PMC+RS, elle montre les différents champs des paramètres de l'RS. L'option "Optimiser la base" déjà expliqué dans le Menu de l'AG.

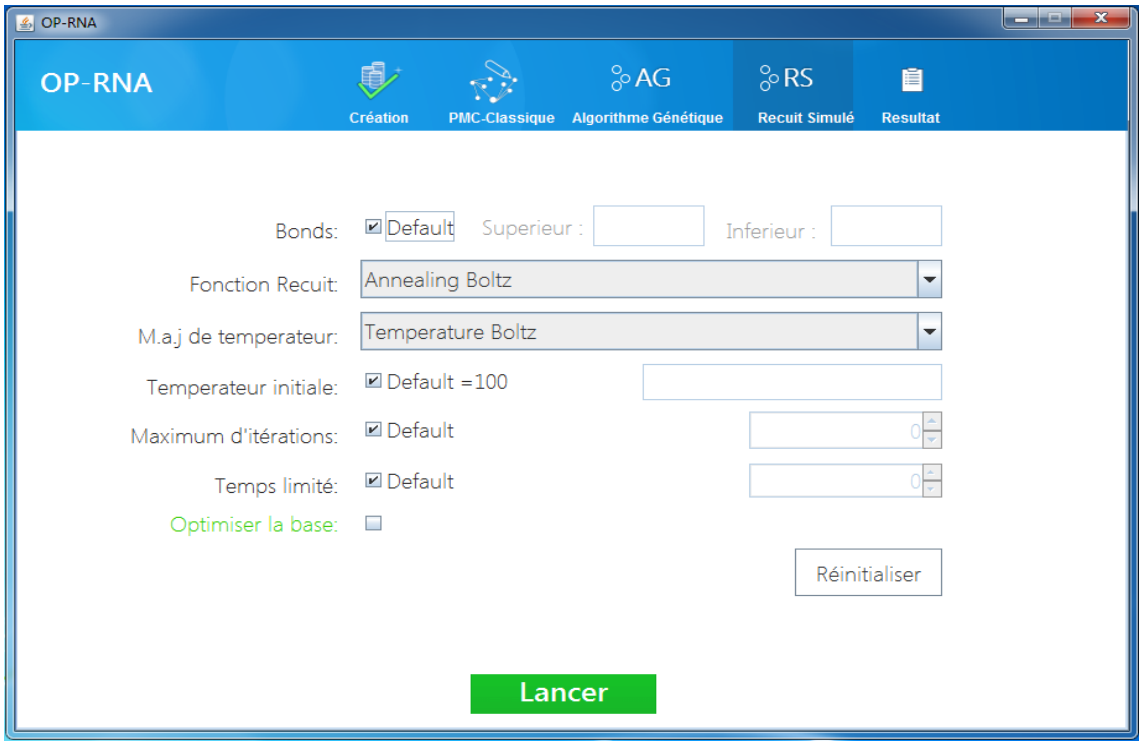

FIGURE 3.6 – Fenêtre de création Modèle PMC+RS.

La figure 3.7, permet de visualiser le graphe de performance avec les différents paramètres obtenu par la méthode utiliser. Le titre de la fenêtre montre le nom de la méthode utilisée.

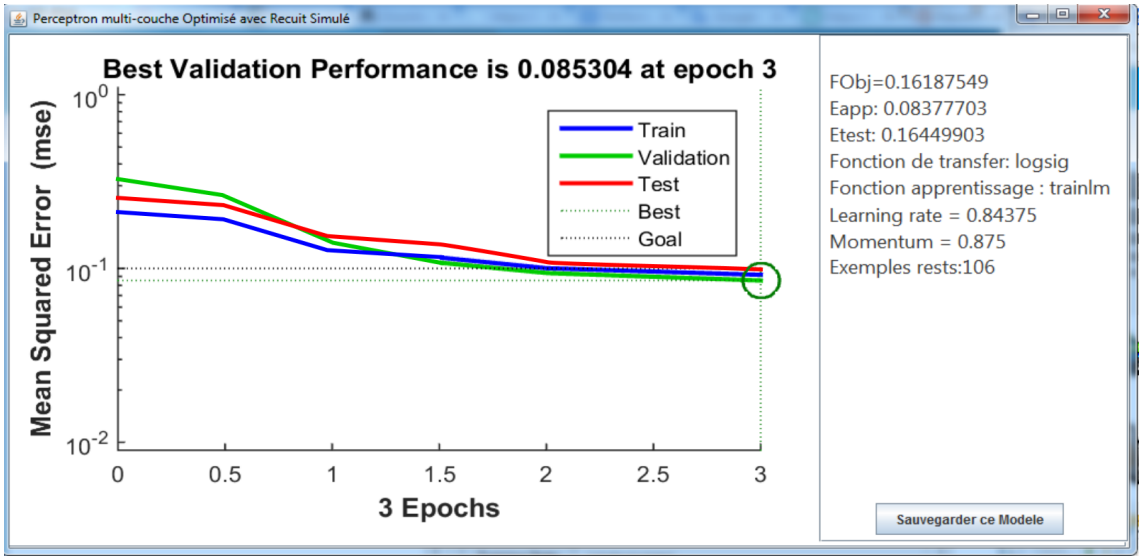

FIGURE 3.7 – Fenêtre de Visualisation des résultats.

La figure suivante3.8, montre à gauche l'interface de menu 'Resultat' qui contient deux buttons :

- 1 : ouvrir la fenêtre (se montre à droite ce Figure) qui permet de faire le diagnostique sur des cas nouveaux on utilisant un modèle déjà sauvegardé (coté expert).
- 2 :permet d'ouvrir et visualiser l'historique des résultats par un navigateur web.

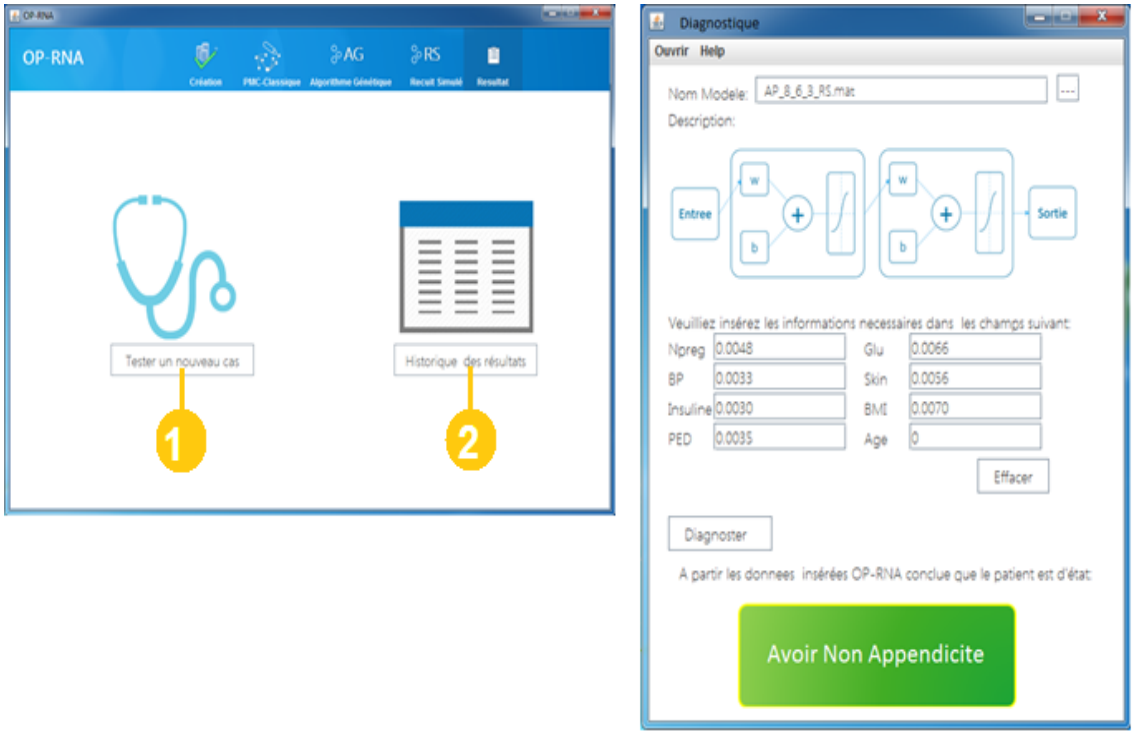

FIGURE 3.8 – Fenêtre de diagnostique.

7 Etude comparative

7.1 Comparaison des résultats

Dans cette section nous comparons les résultats obtenus par un classifieur PMC standard et des résultats obtenus par un classifieur PMC optimisé par différent métaheuristqiues.
Pour la table de comparaison nous utilisons les résultats de l' ensemble de valeurs pour [α1,α2]=[0.1 1] (qui donne l'importance pour l'erreur de test) :

		[8 :6 :6]			[8 :6 :3 :12]		
BDD	Méthode	Erreur		Tc	Erreur		Tc
		App	Test		App	Test	
Pima	PMC	0.1810	0.1771	0.7552	0.1575	0.1475	0.7916
	PMC+AG	0.1757	0.1636	0.8372	0.1567	0.1321	0.8333
	PMC+RS	0.1485	0.1361	0.8281	0.1510	0.1329	0.8333
TH	PMC	0.20	0.1892	0.7790	0.2027	0.2141	0.7440
	PMC+AG	0.1812	0.2000	0.7558	0.2198	0.2071	0.7976
	PMC+RS	0.1518	0.1709	0.8837	0.2615	0.2173	0.8474
AP	PMC	0.1720	0.2011	0.7307	0.2520	0.2409	0.8076
	PMC+AG	0.0992	0.1678	0.8076	0.0695	0.1520	0.8461
	PMC+RS	0.1214	0.1572	0.8461	0.1079	0.1594	0.8846

TABLE 3.9 – Table de comparaison entre les méthodes avec et sans métaheuristique

La comparaison des algorithmes du réseau de neurone optimisé (avec méta-heuristique) et la méthode classique (sans méta-heuristique) permet de mettre en évidence la puissance des premières. Elles présentent un taux de classification et une erreur meilleure à celui des méthodes classique pour les trois bases de données médicales Utilisées.

En ce qui concerne le temps de calcul, les méthodes hybrides (PMC+AG & PMC+RS) sont plus lentes que l'autre (PMC classique), mais offrent une plus grande sécurité pour trouver le minimum absolu.

Les résultats de la méthode RS-PMC a montré une meilleure amélioration par rapport à la méthode d'AG-PMC. Ces résultats de l'hybridation sont encourageants et montrent l'efficacité de ces algorithmes relativement aux algorithmes du PMC seuls. Cependant, cette efficacité dépend des paramètres de l'algorithme Génétique et la Recuit simulé choisi.

Avec ces résultats nous constatons que l'utilisation d'algorithme génétique et le recuit simulé pour l'optimisation d'un classifieur neuronal a amélioré la robustesse de notre classifieur d'une manière considérable.

Pour bien visualiser la différence entre le modèle classique et notre approche nous avons réalisé les courbe de performance montre dans les figures (3.9 3.14) suivant :

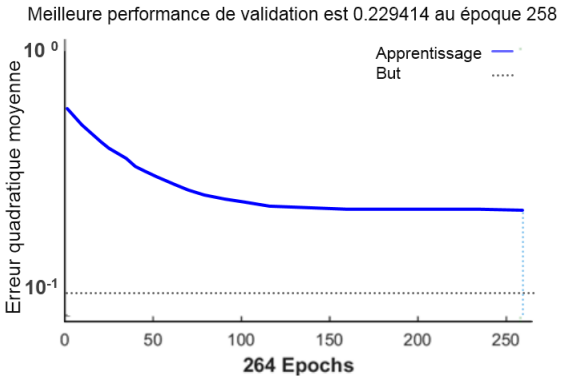

FIGURE 3.9 – La performance d'apprentissage Sans méta-heuristique (PIMA).

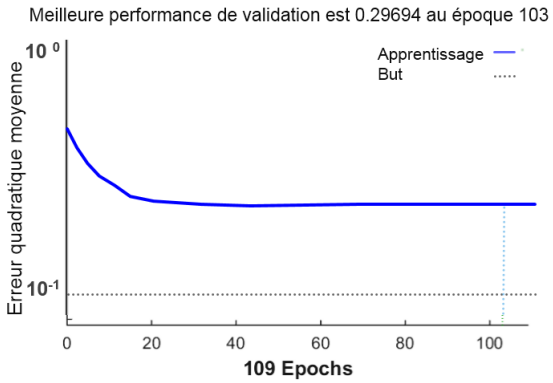

FIGURE 3.10 – La performance d'apprentissage Sans méta-heuristique(TH).

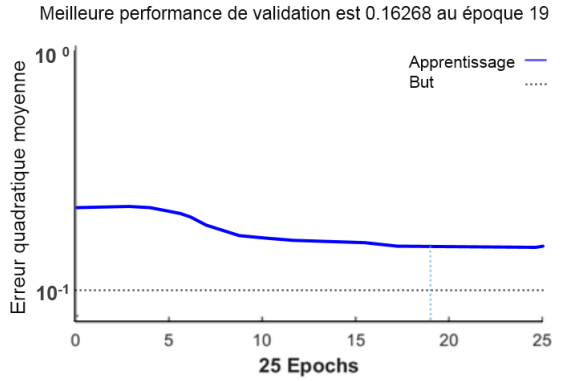

FIGURE 3.11 – La performance d'apprentissage par AG (PIMA).

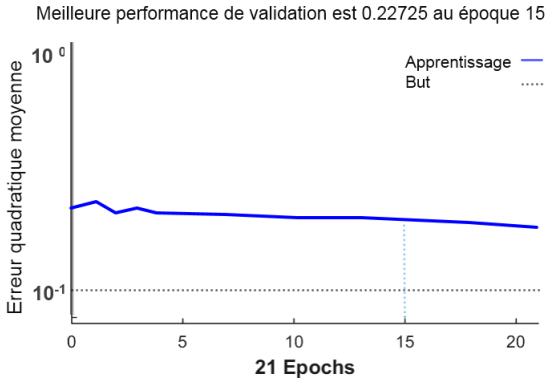

FIGURE 3.12 – La performance d'apprentissage par AG (TH).

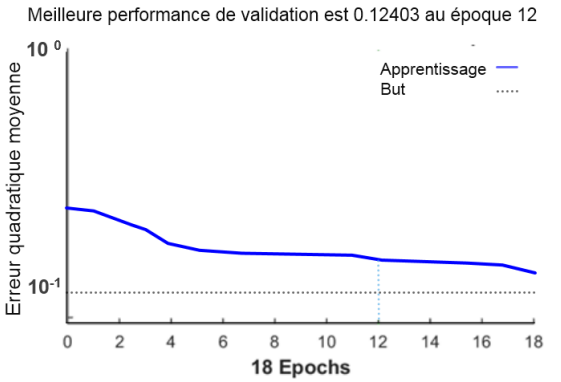

FIGURE 3.13 – La performance d'apprentissage par AG (PIMA).

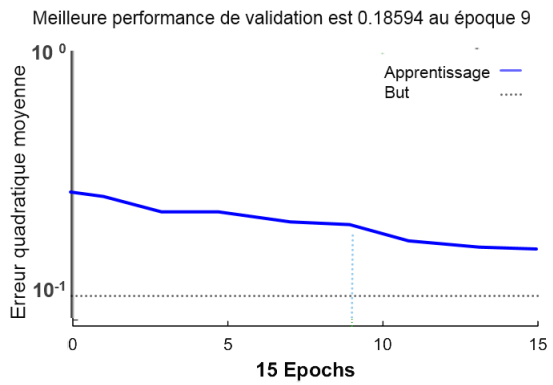

FIGURE 3.14 – La performance d'apprentissage par AG (TH).

A partir de ces résultats nous remarquons que les points de départ des courbes de performance de notre approche (Figure 3.11 & 3.13& 3.12 & 3.14) ont tré proche a l'erreure

minimal(goal) donc le nombre d'époque diminue. Par contre dans l'approche classique le point de départ d'erreur quadratique moyenne est plus loin (Figure 3.9 & 3.10).donc notre approche donne une performance meilleuravec une convergence plus rapide.

7.2 Comparaison avec les résultats de la littérature

Afin de situer la performance de l'approche proposée nous avons réalisé une étude comparative entre les résultat obtenus et celles des travaux déjà réalisés dans ce domaine (étudiés dans l'état de l'art) avec différent base de données les tableaux suivant résume la comparaison avec les autres travaux :

Référence	Méthode	Taux-classification
(Au2001) [45]	Approche Flou	77.6%
(Blachnik 2006) [46]	Net Class	73.83%
	PTDL	70.43%
	C4.5	74.48%
(Pham 2009) [47]	SVM-HBA	94.8%
	ANN-HBA	94.8%
	FCM.HBA	91.7%
(Settouti 2012) [48]	FCM-ANFIS	83.85%
(Bekkadour 2012) [49]	F-HBA	91.7%
Nos –approches	ANN-GA	83.33%
	ANN-SA	83.33%

Table 3.10 – Performance de classification pour la base de données PIMA

Référence	Méthode	Taux-classification
(Pham 2009) [47]	SVM-HBA	100%
	ANN-HBA	98.6%
	DT.HBA	95.7 %
(Rajeswari2010) [50]	Réseau Bayésien Naif	96.52 %
(Ramana2011) [51]	Réseau Bayésien Naif	56.52%
	C4.5	68.69%
	Rétro-propagation	71.59%
	Kplus prouch voisin	62.89%
	SVM	58.26%
(Bekkadour 2012) [49]	F-HBA	100%
Nos –approches	ANN-GA	79%
	ANN-SA	88%

Table 3.11 – Performance de classification pour la base de données TH(Troubles Hépatiques)

Référence	Méthode	Taux-classification
(Weiss 89) [6]	(Cart,K+proche voisin, RNA)	89.6%
(Nakashima 2003) [52]	Classification floue	84.0%
(Blachnik 2006) [46]	NetClass	87.73%
	PTdl	85.77%
	C4.5	85.82 %
(Pham 2009) [47]	SVM-HBA	100%
	ANN-HBA	96.4%
	DT.HBA	89.3%
(Bekkadour 2012) [49]	(F-HBA,ProSadm-HBA)	100%
Nos –approches	ANN-GA	84.61%
	ANN-SA	88.46%

TABLE 3.12 – Performance de classification pour la base de données AP(appendicite)

Pour les mêmes jeux de données, nous remarquons que notre approche obtient des résultats de performance équivalente ou meilleure à celle de la plupart des autres méthodes.

8 Conclusion

La recherche d'un RNA personnalisé et adapté à un problème spécifique est considérée comme une tâche complexe et a principalement utilisé des méthodes manuelles de recherche. Dans ce chapitre, nous avons présenté une méthode hybride de recherche automatique des paramètres de RNA quasi-optimales. Cette méthode est composée d'une combinaison entre le PMC et deux métaheuristiques (PMC+AG et PMC+RS) qui recherchent les différents paramètres du réseau. Des expériences ont été effectuées et les résultats démontrent que ce procédé est capable de réaliser des réseaux de neurones avec des performances satisfaisantes en comparaison avec le perceptron sans métaheuristique et avec d'autres procédés récents décrits dans la littérature.

Conclusion générale

Les techniques d'intelligence Artificiels telles que les réseaux neurologiques artificiels, la logique floue, les algorithmes génétiques (AGs),et le recuit simulé sont des sujets populaires de recherches, puisqu'elles peuvent traiter des problèmes complexes difficilement résolu par des méthodes classiques, Les métaheuristiques ont attiré une attention considérable à la communauté du recherche en informatique biomédicale. Une des approches les plus populaires est l'optimisation des réseaux de neurones artificiel par métaheuristiques, telques : les algorithmes génétiques ou le recuit simulé.

Afin de créer un système perfomant et parcimonieux , utilisé pour la classification des données médicale, nous avons implémenté deux métaheuristiques qui a comme but d'ajuster les paramètre d'un perceptron multicouche pour faire une bonne qualité de reconnaissance.
Le modèle développé (nommé Op-RNA) permet d'optimiser les paramètre du PMC (poids de connexion, fonction d'activation, fonction d'apprentissage, pas d'apprentissage, et le moment)et donnée la meilleure configuration pour effectue une bon classification.
Les résultat obtenue par Op-RNA sont très prometteurs et sont bien situé parmi les travaux déjà réalisés dans ce domaine ce qui confirme la rigueur de la contribution proposée pour la résolution de notre problématique.

Perspectives :
Dans les perceptive d'avenir, nous prévoyons d'assurer l'interprétabilité des résultats du modèle en intégrant la notion du flous avec le classifieur. Nous voulons aussi améliorer et automatiser (optimisation de l'architecture, et le choix des paramètre des métaheuristique) notre application afin de l'intégrer dans l'avenir dans un système d'aide au diagnostic applicable dans un hôpital ou dans un cabinet médical avec une interface simple et compréhensible peut utiliser par un expert du domaine.

Bibliographie

[1] V.Gardeux., *Conception d'heuristiques d'optimisation pour les problèmes de grande dimension.Application à l'analyse de données de puces à ADN*, Ph.D. thesis, Université de Paris-Est Créteil, 2011.

[2] J.Dréo, *Adaptation de la méthode des colonies de fourmis pour l'optimisation en variables continues. Application en génie biomédical*, Ph.D. thesis, université Paris12, 2004.

[3] El-Ghazali Talbi, *Metaheuristics : From Design to Implementation . :*, ISBN, July 2009.

[4] ""techniques avancées pour le traitement de l'information", réseaux de neurones, logique floue, algorithmes génétiques, cépaduès-Éditions,198 p., 1996.," .

[5] J. Dréo, A. Pétrowski, P. Siarry, and E. Thaillard, *Métaheuristiques pour l'optimisation difficile*, Eyrolles, 2003.

[6] I. . InProceedings of the 1989. Weiss, S. M.and Kapouleas, "An empirical comparison of pattern recognition,neural nets, and machine learning classification methods," in *11th International joint conference on artificial intelligence pp. 781–787. Detroit, MI, USA.*, 1989.

[7] S.Haykin, *Neural Networks : A Comprehensive Foundation*, Englewood Cliffs, 1999.

[8] A..Abraham, "Meta learning evolutionary artificial neural networks," *Neurocomputing*, vol. 56, pp. 1–38, 2004.

[9] L. Jourdan, "Métaheuristiques coopératives : du déterministe au stochastique. modeling and simulation," M.S. thesis, Université des Sciences et Technologie de Lille - Lille I, 2010.

[10] I.H. Osman and G. Laporte, "Metaheuristics : A bibliography. annals of operations research 63, 513–623 (1996)," .

[11] Blum C and Roli A ., "Metaheuristics in combinatorial optimization," *Overview and Conceptual Comparison. ACM Computing Surveys*, vol. 35(3), pp. 268–308, 2003.

[12] H.H. Hoos and T. Stützle., *Stochastic local search : Foundations and applications.* ,, Morgan Kaufmann, 2004.

[13] F. Glover, "Heuristics for integer programming using surrogate constraints.," *Decision Sciences*, vol. 8, pp. 156–166, 1977.

[14] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Computers and Operations Research*, vol. 13(5), pp. 533–549, 1986.

[15] P. Hansen, "The steepest ascent mildest descent heuristic for combinatorial programming. in numerical methods in combinatorial optimization, 1986," .

[16] F. Glover, "Tabu search - part i," *ORSA Journal on Computing*, vol. 1(3), pp. 190–206, 1989.

[17] F. Glover, "Tabu search - part ii," *ORSA Journal on Computing*, vol. 2(1), pp. 4–32, 1990.

[18] F. Glover and M. Laguna, "Tabu search," *Kluwer Academic Publishers*, vol. 5, pp. 18, 1998.

[19] E. Taillard, "Robust taboo search for the quadratic assignment problem," *Parallel computingelsevier*, vol. 17(4-5), pp. 443–455, 1991.

[20] R. Batitti and G. Techioiii, "The reactive tabu search," *ORSA Journal on Computing*, vol. 6(2), pp. 126–140, 1994.

[21] Gelatt C. Kirkpatrick, S. and M. V ecchi, *Optimization by simulated annealing*, Science, 1983.

[22] Rosenbluth A. W. Rosenbluth M. N. T eller A. H. Metropolis, N. and E T eller, "Equation of state calculations by fast computing machines," *Journal of Chemical Physics*, p. 21, 1953.

[23] J.H.Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

[24] E.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1989.

[25] C. Darwin, "On the origin of species by means of natural selection or the preservation of favoredraces in the struggle for life, murray, london.(in dréo, j., pétrowski, a., siarry, p. et thaillard, e. (2006). metaheuristics for hard optimization, springer 2006).," 1859.

[26] Davalo E. and Naim P, *Des Réseaux de Neurones*, EYROLLES, 1993.

[27] F.Rosenblatt, "The perceptron : a probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, pp. 386–408, 1958.

[28] D.Rumelhart, G.Hinton, and R.Williams, *Parallel Distributed Processing*, MIT Press, 1986.

[29] "Neural networks for pattern recognition, oxford university press, 1995.," .

[30] G.Dreyfus, J-M.Martinez, M.Samuelides, M.BGordon, F. Badran, and SS.Thiria L.Hérault, *Réseaux de neurones : Méthodologie et Applications*, Eyrolles, 2002.

[31] O.Nerrand, P.Roussel-Ragot, L.Personnaz, and G. Dreyfus, "Neural networks and nonlinear adaptive filtering : unifying concepts and new algorithms," *Neural Computation*, vol. 5, pp. 165–199, 1993.

[32] C.Touzet, "Les réseaux de neurones artificiels : Introduction au connexionnisme : Cours, exercices et travaux pratiques," 1992.

[33] K.P.Ferentinos, "Biological engineering applications of feed-forward neural networks designed and parameterized by genetic algorithms," . *Neural Networks*, vol. 18(7), pp. 934–950, 2005.

[34] L.M.Almeida and T.B.Ludermir, "Automatically searching near-optimal artificial neural networks," *European Symposium on Artificial Neural Networks*, vol. 6, pp. 549–554, 2007.

[35] L.M.Almeida and T.B.Ludermir, "An improved method for automatically searching nearoptimal artificial neural networks," in *International Joint Conference on Neural Networks*, 2008.

[36] A.E.Eiben and J.E.Smith, *Introduction to Evolutionary Computing.*, Springer, Heidelberg, 2003.

[37] F. P.Eduardo, da Luz, F.Ariane, and dos Santos., "Weights estimation by firefly with predation optimization for ensemble precipitation prediction using brain," *Edição Esp. Dez.*, vol. 2, pp. 091 – 093, 2013.

[38] R.Adenilson, M.Carvalho Fernando, Ramos Antonio, and A. Chaves, "Metaheuristics for the feedforward artificial neural network (ann) architecture optimization problem," 2010.

[39] Juliana A. Anochi, Haroldo F., Campos Velho, Helaine C.M. Furtado, and Eduardo F.P Luz., "Self-configuring two types neural networks by mpca," in *2nd International Symposium on Uncertainty Quantification and Stochastic Modeling*, June 23th to June 27th, 2014 Rouen, France.

[40] Xin Yao, "Evolving artificial neural networks,prceding," *IEEE*, vol. 87(9), pp. 1423–1447, septembre1991.

[41] M. Leandro Almeida and Teresa B. Ludermir., "An evolutionary approach for tuning artificial neural network parameters," *Springer-Verlag Berlin Heidelberg*, vol. 7, pp. 156–163, 2008.

[42] Meng Joo Er and Fan Liu, *Parameter Tuning of MLP Neural Network Using Genetic Algorithms*, Springer-Verlag Berlin Heidelberg, 2009.

[43] A. Frank and A. Asuncion, "Uci machine learning repository, [http ://archive.ics.uci.edu/ml]. university of california, irvine,ca : School of information and computer science. (access 2010).," .

[44] Settouti Nesma, "Renforcement de l'apprentissage structurel pour la reconnaissance du diabète," M.S. thesis, Université Abou bekr belkaid Tlemcen, 28 Juin 2011.

[45] W. H. Au and K. C. C. . Chan, "Classification with degree of membership : a fuzzy approach," in *1st IEEE international conference on data mining pp.35–42. San Jose, CA, USA.*, 2001.

[46] W. Blachnik, M.and Duch, "Prototype-based threshold rules. lncs of neural information processing," *Berlin/Heidelberg : Springer*, vol. 4234, pp. 1028–1037., 2006.

[47] Pham HNA and Triantaphyllou E., "An application of a new meta-heuristic for optimizing the classification accuracy when analyzing some medical datasets," *Expert Systems with Applications*, vol. 36(5), pp. 9240–9, 2009.

[48] Nesma Settouti, Meryem Saidi, and Mohamed Amine Chikh., "Interpretable classifier of diabetes disease," *International Journal of Computer Theory and Engineering*, vol. 4, pp. 438–442, 2012.

[49] Bekaddour Fatima and M. Amine Chikh., "An interpretable fuzzy classifier for breast cancer diagnosis," in *Biomedical Engineering International Conference (BIOMEIC' 12)*, October 10-11, 2012.

[50] P.Rajeswari and G.Sophia Reena., "Analysis of liver disorder using datamining algorithm.," *Global journal of computer science and technology*, vol. 10, pp. 48–52, November 2010.

[51] Bendi Venkata Ramana, M.Surendra Prasad Babu, and N.B.Venkateswarlu, "A critical study of selected classification algorithms for liver disease diagnosis," *International Journal of Database Management Systems (IJDMS*, vol. 3, pp. 18, May 2011.

[52] G.and Ishibuchi Nakashima, T.and Nakai and H. Constructing ., "fuzzy ensembles for pattern classification problems," in *international conference on systems, man and cybernetics (Vol. 4, pp. 3200–3205). Washington, DC, USA (October).*, 2003.

Résumé

La spécification correcte des paramètres d'un RNA quasi optimale d'une manière automatique pour un problème spécifique avec une performance satisfaisante est le principale aspect qui est motivé notre approche présent dans ce mémoire de master.

Cette approche emploie une recherche évolutive (AG ou RS) pour effectuer le réglage simultané des poids initiaux, des fonctions d'activation, fonctions d'apprentissage, pas d'apprentissage, et le moment de la PMC. Des expériences ont été effectuées et les résultats obtenue après l'utilisation de cette méthode montrent qu'elle est capable de trouver une meilleure configuration pour effectue une bon classification. Nous avons validé nos résultats expérimentaux sur des bases de données médicales connues : *Pima (Diabètes), TH (Troubles Hépatiques), AP (Appendicite)*.

Mots clés : Algorithmes évolutionnaires, Algorithmes génétiques, Recuit simulé, Optimisation, Métaheuristique, Paramétrage de réseau de neurone artificielle, Les poids initiaux.

Abstract

The correct specification of parameters of a quasi-optimal RNA in an automatic way to a specific problem with a satisfactory performance is the main aspect that motivated our present approach in this master thesis.

This approach employs an evolutionary search (GA or AS) to perform simultaneous tuning of the initial weights, activation functions, learning functions, learning rat, and momentum of MLPNN. Experiments were performed and results obtained after using this method show that it is able to find a better configuration for conducting a proper classification. We validated our experimental results on known medical databases : *Pima (Diabetes), TH (Liver Disorders), AP (appendicitis)*.

Keywords : Evolutionary algorithms, Genetic algorithms, Simulated annealing, Optimization, Metaheuristics, Artificial neural network setting, The initial weights.