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

Faculté des sciences de l'ingénieur
Département d'électronique biomédicale
Laboratoire de génie biomédical

Apprentissage génétique d'un classifieur neuronal Application en cardiologie

Mémoire pour l'obtention du diplôme de
Magister en
Électronique biomédicale

Présenté par :

Mme BENDIMERAD MANSOURIA Née SEKKAL

Soutenu
Devant le jury constitué par :

Président :	**Mr S. DEBBAL**	**Maître de conférences**
Examinateurs :	**Mr S.M. MERIAH**	**Maître de conférences**
	Mme M. Benallel	**Maitre Assistante**
Encadreur	**Mr M.A CHIKH**	**Maître de conférences**

Année universitaire 2008-2009

REMERCIMENTS

Avant tout, je remercie Dieu de m'avoir aidé à réaliser ce présent trava

Je tiens à exprimer ma très profonde gratitude à Monsieur CHIK Mohamed Amine, docteur et maitre de conférences à l'université de Tlem qui n a ménagé aucun effort pour me prendre en charge pour la réalisation de travail.

Sa clairvoyance, sa générosité, sa gentillesse, ses connaissan temps qui m'a dispensé, et sa grande disponibilité dont il a fait preuve; m'ont énormément facilité ma tâche.

J'adresse mes très sincères remerciements à Monsieur BEBBEL Sidi Mohamed, maitre de conférences à l'université de Tlemcen qui a accepté la présidence du jury.

Je tiens aussi à remercier Monsieur MERIAH Sidi Mohamed, maitre de conférences à l'université de Tlemcen d'avoir accepté d'examiner travail.

Je tiens aussi à remercier Monsieur BENALLEL MOUNIRA maitre assistante à l'université de Tlemcen d'avoir accepté d'examiner ce

Dédicace

- *A mes très chères parents, mon père ainsi que ma mère*
- *A mon chère époux, qui ma donné beaucoup de soutien*
- *A mes trois frères que j'estime beaucoup : Mohamed, Hocine et Idriss*
- *A mon beau-père, mes beaux frères ainsi que mes belles sœurs*
- *A toutes les familles : BENDIMERAD ,SEKKAL ET KHEDIM*
- *A toutes mes amies spécialement : Aouicha*

Mansouria

Table des matières

Table des figures

Résumé

Le diagnostic médical est l'activité fondamentale de la médecine, il est donc normal que les premiers investigations dans le domaine du traitement automatique des informations s'intéressent à ce type problème.

Plusieurs systèmes de diagnostic automatique utilisent les réseaux de neurones artificiels, comme technique de reconnaissance. Mais cette technique avec son apprentissage classique (rétro propagation) peut tomber sur des solutions sous optimale (minima locaux).

Dans ce travail, nous avons conçu et implémenté un système neuro-génetque en remplaçant l'apprentissage classique par les algorithmes génétiques en suite nous avons comparé les résultats obtenus avec les systèmes neuronaux classiques.les performances de la classification ont été évaluées par le calcul de erreurs (*e)*, temps *(T)*, sensibilité *(Se)*, la spécificité *(sp)* et le taux de classification correcte*(πclass).*

Aussi dans ce travail, on a utilisé les algorithmes génétiques pour choisir l'architecture des RNA.

Nous avons appliqué cette technique sur les signaux électrocardiogrammes (ECG), et comme base de données la base MIT-BHI.

Mots clés

Modèle hybride, RNA, AGs, neuro-génétique, apprentissage paramétrique, apprentissage structurel.

Abstract

Medical diagnosis is the fundamental activity of the medicine, it is so normal that first investigations in the field of the automatic treatment of information should interested in this type problem.

Several systems of automatic diagnosis use the neural networks, as technique of recognition. But this technique with the classic learning (back-propagation) can fall on solutions under optimal (local minima).

In this work, we conceived and implemented a system neuro-génetque by replacing classic learning by genetic algorithms in continuation we compared results obtained with classic neuronal systems. The performances of the classification were estimated by the calculation of errors, time (T), sensibility (SE), the specificity (sp) and the rate of correct classification (τclass).

So in this work, one used genetic algorithms to choose the architecture of the RNA.

We applied this technique to signals electrocardiograms (ECG), and as data base bases it MIT-BHI.

Keywords

Hybrid model, neural networks, genetic algorithms, neuro-genetic, learning paramétrique, structural learning.

Introduction générale

Les systèmes d'aide au diagnostic médical permettent de chercher une éventuelle solution pour des problèmes médicaux difficiles. L'un des rôles de ces systèmes est de trouver une situation d'un patient à partir d'information sur ses caractéristiques. Cette situation a une classe dont le nom est celui d'une cause possible de la situation (anormale) observée. Cette phase est appelée ***la classification***.

L'approche neuronale est l'une des approches utilisées dans la phase de classification. Plusieurs travaux de recherche utilisent cette méthode avec un apprentissage classique qui utilise la technique de la rétro propagation pour la classification de différentes pathologies représentées sous forme des signaux physiologiques ex : ECG, EEG ou image ex. : IRM.

L'apprentissage des réseaux de neurones multicouches utilise la méthode de la rétropropagation, elle est basée sur l'algorithme de la descende de gradient qu'est une technique d'optimisation locale qui peut conduire a une solution sous optimale.

Ainsi les RNMC (réseaux de neurones multicouches) sont liés par un ensemble de paramètres architecturaux ; on peut citer : le nombre de neurones cachés, les liaisons entre les neurones…. etc.

Nous traitons dans ce mémoire l'hybridation les RNMCs avec une technique d'optimisation globale : (**algorithmes génétiques).**

Les algorithmes génétiques appartiennent à la famille des algorithmes évolutionnistes .Leur but est d'obtenir une solution approchée, à un problème d'optimisation. Les algorithmes génétiques utilisent la notion de sélection naturelle développée au XIX^{e} siècle par le scientifique Darwin et l'appliquent à une population de solutions potentielles au problème donné.

Dans ce travail, nous hybridons les AGs par les RNMCs dans deux situations différentes : la 1ere consiste à remplacé la rétro propagation par les AGs(un apprentissage paramétrique). La 2eme consiste à utilisé les AGs, pour choisir l l'architecture optimale des RNMCs (utilisé pour un apprentissage structurel).

Ce mémoire est composé de quatre chapitres, nous présentons dans le premier chapitre quelques notions sur les systèmes d'aide au diagnostic médical et ses différentes approches, ainsi que des notions sur la classification.

Le deuxième chapitre porte sur les réseaux de neurones artificiels, ses différentes architectures, ses différents types d'apprentissages et ses différents problèmes.

Le chapitre trois décrira le principe des algorithmes génétiques avec les différents techniques de ces opérateurs, ainsi, l'hybridation de ces derniers avec les réseaux de neurones pour assurer un apprentissage paramétrique et un apprentissage structurel.

Nous présentons dans le dernier chapitre une comparaison entre des classifieurs neuronaux classiques et des classifieurs neuro- génétiques et l'interprétation des résultats obtenus.

Le but de ce modeste travail est de montrer l'intérêt des AGs pour régler les problèmes des réseaux de neurones utilisés pour la classification des signaux biomédicaux.

CHAPITRE1

aide au diagnostic médical

I.1 Introduction

Le diagnostic médical est une activité fondamentale de la médecine. Tout professionnel de la santé se doit à un moment ou à un autre d'établir une classification des données médicales de son patient. Il est donc normal que les premières investigations dans le domaine du traitement automatique des informations se soient intéressées à ce problème.

Les pionniers de l'informatique médicale R. Ledeley et L. Lusted , E.Short-liffe, F. De Domal ou F.Gremy en France, ont cherché à automatiser l'élaboration d'un diagnostic médical. Les méthodes développées s'appuyaient sur des modèles statistiques (calcul de probabilités) et /ou des modèles logiques issus des recherches menées en intelligence artificielle.

Ce chapitre présente les systèmes d'aide aux diagnostique avec ces différent approche ainsi une notion sur la classification.

I.2 Les systèmes d'aide au diagnostic

I.2.1 Définition :

Le système ADM (Aide au diagnostic Médical) est Créé il y a une quinzaine d'années**,** Permet de rechercher une éventuelle solution pour des problèmes médicaux difficiles et, en associant un outil informatique performant dont toutes les données sont référencées ce qui aide à analyser la pertinence des hypothèses proposées après saisie de signes et/ou des examens complémentaires. Il avait initialement deux objectifs principaux : aider les médecins à porter des diagnostics et leur offrir un accès rapide à l'information médicale en utilisant les réseaux télématiques

I.2.2 L'intérêt d'automatisation d'un diagnostique :

"Le diagnostic est l'identification d'une maladie par ses symptômes" *(Dictionnaire Larousse)*. Ceux-ci ayant été analysés par l'interrogatoire et l'examen clinique, il ne reste plus au médecin qu'à puiser dans ses connaissances pour trouver la maladie en cause. Mais il existe des difficultés au diagnostic médical qui peuvent être liées aux praticiens ou aux malades.

a) difficultés liées avec les médecins :

- les possibilités de mémorisation sont inégales, entre les médecins
- les connaissances variables en fonction de l'ancienneté des études et d'un éventuel enseignement post-universitaire suivi.

- Un médecin peut avoir ponctuellement fatigue ou souci diminuant ses capacités de concentration.
- Le cerveau humain ne serait pas fait pour garder un trop grand nombre de données, présentes à l'esprit en même temps ; ce qui rend très difficile l'approche tenant compte de tous les signes présents.
- De nombreuses études sur le raisonnement médical montrent que, généralement, le médecin élabore un petit nombre d'hypothèses (quatre à six), très tôt en phase d'examen, sur un faible nombre de signes caractéristiques (un à deux), avec un bon degré de fiabilité.
- Ce qui arrive aussi parfois, c'est que le cerveau engagé dans une direction, bloque, par un phénomène d'induction négative, les autres voies: le diagnostic positif erroné exclut le diagnostic différentiel qui paraît inutile ; cela peut même entraîner une hypothèse diagnostique élaborée avant la fin de l'examen, qui n'est plus systématique, mais orienté par la première "impression" ou "intuition".
- Un signe peut aussi bloquer la bonne voie de diagnostique.

b) <u>difficultés liées avec les malade (ou les maladies)</u> :

- Un signe peut bloquer la bonne voie diagnostique: leur expression peut-être trompeuse.

Intérêt d'une aide diagnostique informatisée :

" Les systèmes experts impliquent pour les médecins deux comportements différents :

-En phase d'exploitation, le médecin peut s'appuyer sur les déductions auxquelles le système expert est arrivé.

-En phase de révision, le médecin peut simuler des multitudes de symptômes et observer les déductions du système expert. Dans ce cas, le système expert permet au médecin pratiquant, non seulement de réviser ses connaissances, mais d'affiner son diagnostique "

Mais if faut mentionner que Le but n'est pas de remplacer le médecin (celui-ci ne pourra jamais être remplacé par un système automatisé), mais de créer un outil *d'aide* au *diagnostic*. Les systèmes experts représentent l'image d'un "conseiller virtuel".

I.2.3 Caractéristiques des systèmes d'aide à la décision : [P.Deg&M.Fie2005]

I.2.3.1 Nature de l'aide à la décision

Les systèmes d'aides à la décision médicales s'inscrivent dans la problématique générale de l'intervention médicale : prédire (médecine prédictive), prévenir (médecine préventive), guérir (médecine curative) et, si cela n'est pas possible, soulager (assistance médicale).pour cela, il faut apprécier la situation individuelle du patient (diagnostique et pronostique) et évaluer les stratégies possible. À partir de cette problématique, on distingue habituellement deux types des systèmes d'aide à la décision :

- *Les systèmes permettant de mieux apprécier l'état d'un patient* (ce qui est vrai).ils concernent essentiellement la décision diagnostique ou pronostique.il visent à diminuer l'incertitude sur la situation actuelle ou future de patient. Ils doivent tenir compte de connaissances médicales multiples (épidémiologique, sémiologisue, physiopathologique, anatomique, etc).
- *Les systemes cherchant à proposer la meilleure stratégie* (ce qu'il faut faire). Quels examens complémentaires faut-il réaliser ?quelles règles de vie, quels médicaments ou quelle intervention proposer ?comment informer au mieux le patient de son état ? ils doivent intégrer des considérations financières et éthiques.

I.2.3.2 Mode d'intervention

a. Système passifs

La plupart des systèmes d'aide à la décision fonctionnent en mode passif. Le médecin doit explicitement faire appel au système.il décrit le cas de patient et attend un conseil du système. Selon les informations fournies et le type de conseil attendu, on peut considérer deux approches :

- Dans un *système consultant*, l'utilisateur fournit des informations sur l'état du patient en retour, le système donne un conseil diagnostique ou thérapeutique. MYCIN, développé par E.Shortliffe et coll. A l'université de stanford, est l'exemple type de système consultant. **[shortliffe 1976].**
- Dans un système critique, l'utilisateur fournit des informations sur l'état du patient et la stratégie (thérapeutique et / ou d'investigation) envisagée par le médecin. Le système critique alors les propositions du praticien. Le

système ATTENDING , développé par P.Miller à l'université de Yale, en est un bon exemple**[Miller 1986a].**l'application en anesthésie permet, par exemple, de critiquer pour un patient donné le plan fourni par le spécialiste(anesthésique choisis, induction, administrations, etc.)

b. Systèmes semi-actifs

Dans un système semi-actif, l'appel au système d'aide à la décision est effectué automatiquement. L'objectif est de fournir des informations, des règles de comportement et de connaissances générales acceptées par tous. Le système joue au minimum le rôle de garde-fou ou de « chien de garde » (watch-dog system). On peut classer dans cette catégorie :

- Les *systèmes de rappel* automatique (reminders) qui supervisent l'attitude du soignant **[McDonald1976].**ils permettent d'éviter des prescriptions d'investigations redondantes, des erreurs de prescription (erreur de posologie, contre-indication médicamenteuse, interaction significative, etc.). ils facilitent le respect, par l'équipe soignante, de protocoles préétablis, etc.
- Les *systèmes d'alarme* qui permettent d'attirer l'attention sur un changement de la situation de patient. il peut s'agir de signaler des valeurs anormales (paramètres biologiques ou physiologiques) ou des modifications anormales (élévation ou abaissement brutal d'un paramètre).

c. Systèmes actifs

Un système actif est un système qui, se déclenchant automatiquement, est capable de fournir un conseil adapté au cas particulier du patient. Il peut prendre automatiquement des décision sans intervention du praticien.il peut s'agir, par exemple, de la prescription d'examens complémentaires (prescription automatique de bilans à partir de protocoles de soins).de thérapeutique (contrôle automatique d'une perfusion par un système en boucle fermée).de surveillance (contrôle intelligent des paramètres d'un ventilateurs, d'un moniteur de dialyse ou d'un pacemaker) ou d'assister un geste chirurgical.

I.2.4 Bases méthodologiques des systèmes d'aide à la décision

L'aide à la décision à bénéficié d'un large éventail de méthodes. Outre les algorithmes simples, on distingue différents types d'approches dans les systèmes d'aide à la décision. Chacune de ces approches peut faire appel à des modalités particulières de raisonnement.

Approche	Raisonnement	Explication	apprentissage
Modèle mathématique	algorithmique	Non	non
Statistique/probabiliste	inductif	limités	Automatique par induction
Système experts	Déductif, abductif	Oui	Difficile, supervisé
Réseaux neuronaux	inductif	non	Supervisé ou automatique

Tableau 1.1 : les approches de l'aide à la décision

I.2.4.1 Différents types de raisonnement

a. ***Raisonnement Inductif*** : le raisonnement inductif permet, par généralisation, de passer de cas particuliers à une loi plus générale. Le raisonnement inductif produit des interférences valides avec un certain degré de crédibilité ou probabilité.si les hommes x, y, z etc. sont mortels, alors on peut poser comme hypothèse d'induction que l'homme est mortel.des expériences répétée permettent de confirmer ou d'infirmer une hypothèse émise.

b. ***Raisonnement Déductif*** : le raisonnement déductif est basé sur les principes de l'implication logique.il permet d'interférer des conclusions dont le degré de vérité est uniquement fonction du degré de vérité des prémisses. La déduction permet de passer d'un cas général à un cas plus particulier. Si la règle « tous les hommes sont mortels » est vraie. Alors Socrate, qui est un homme, est mortel (*modusponens*) . Les résultats d'une interférence logique peuvent servir de prémisse pour une déduction ultérieure. Ainsi, si A implique B et si B implique C alors, par transitivité, A implique C.

c. ***Raisonnement Abductif :*** le raisonnement par abduction est un composant important de la démarche scientifique. On dispose d'observations entre lesquelles on cherche à établir un lien, tel qu'un lien de causalité. Les hypothèses peuvent porter sur la sélection d'une loi qui permettrait d'établir le lien entre les faits considérés comme antécédents et conséquents(par ex. hypothèse diagnostique) ou sur l'établissement d'une nouvelle loi (par ex. découverte scientifique).en supposant que l'hypothèse est vraie, le raisonnement abductif permet alors d'inférer de nouveaux faits conséquents qui devront être confirmés par la réalisation d'examens complémentaires ou de nouvelles expériences scientifiques (principe de la démarche expérimentale).

I.2.4.2 Différentes approches utilisés

a. Utilisation de modèles mathématiques

Des modèles mathématique ont été proposés de puis plusieurs dizaines d'années pour la description de système biologiques ou physiologiques complexe (hémodynamique, effet des radiations ionisantes sur des tissus pharmacocinétiques). Ces modèles peuvent directement servir à la décision médicale, en mode passif (analyse des conséquences de la variation d'un ou plusieurs paramètres) comme en mode actif (contrôle automatiques). Ils sont utilisés pour la description de systèmes biologiques ou physiologiques : pharmacocinétique, surveillance en réanimation...

b. Méthode statistiques

Les méthodes statistiques, reposent essentiellement sur des méthodes de classification multidimensionnelle. Un patient est affecté à une classe (thérapeutique ou diagnostique) en fonction de la valeur de ces paramètresx_i. Un échantillon d'apprentissage, constitué de cas pour lesquels la décision (diagnostique, pronostique ou thérapeutique) a été établie, permet de trouver les variables discriminantes et d'estimer les valeurs de coefficientsa_i. Une fois déterminée l'équation de décision, on peut alors l'appliquer au cas d'un malade particulier.

Des exemples d'outils statistiques sont la régression multiple et l'analyse discriminante. Elles peuvent être utilisées pour la classification diagnostique (par ex. Analyse discriminantes), pronostique ou thérapeutique (par ex. régression multiple).

c. Systèmes probabilistes

Les systèmes probabilistes sont essentiellement basés sur l'utilisation du théorème de Bayes et la théorie de la décision.la probabilité a posteriori d'observer un diagnostic D_i lorsque le signe S est présent en fonction de la probabilité a priori du diagnostic et de probabilité d'observer le signe lorsque le diagnostique est présent (probabilité conditionnelle)

d. systèmes experts

L'intelligence artificielle (IA) est « l'étude des idées et des techniques permettant de rendre les ordinateurs intelligent » .L'objectif est double :

- Pragmatique, rendre les ordinateurs plus utiles,
- Théorique, mieux comprendre les mécanismes de l'intelligence.

L'intelligence artificielle est confluent de plusieurs disciplines : l'informatique, la linguistique, la psychologie cognitive, l'épistémologie.

L'une de ses retombées principales a été le développement des systèmes experts, définis comme des programmes informatiques utilisant des connaissances spécialisées et des mécanismes d'interférences pour obtenir, dans un domaine particulier , des niveaux élevés de performances.les connaissance sont généralement fournis sous forme de règles ou de frames .la base contenant les connaissances ou base de connaissances est séparée de la base contenant les données du problème à analyser ou base de faits et du système de gestion des interférences ou moteur d'interférence .

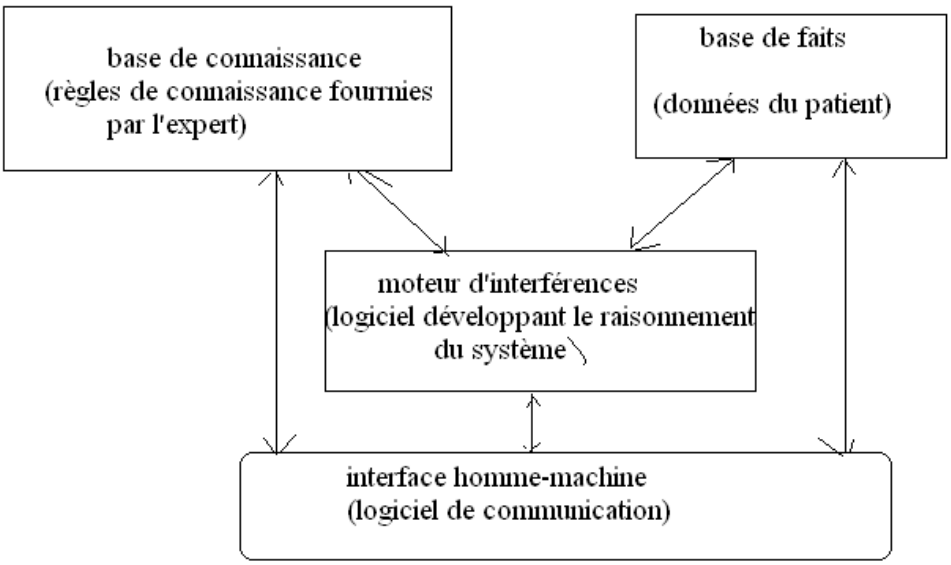


Figure1.1 : Architecture générale d'un système expert

Système expert utilisent pour raisonner la logique dans ses différentes formes : logique des propositions, logique d'ordre 1, logique floue.

Les systèmes experts présentent quelques limites comme

- *La connaissance est multiple*
- *La connaissance est infinie*
- *La connaissance est souvent contradictoire ou inconstante*
- *Le raisonnement est limité (déduction, abduction)*
- *La validation des connaissances est difficile*

e. Réseaux de neuronaux et systèmes connexionnistes

Les systèmes neuronaux ont trouvé leurs premières applications pratiques dans le développement de systèmes de reconnaissance de forme (reconnaissance de caractères, vocales, de contours dans une image, etc.).Ils se prêtent bien à des problèmes de classification diagnostique lorsqu'une base de cas suffisante est disponible. La couche d'entrée du réseau peut alors être assimilée aux symptômes et la couche de sortie aux diagnostics

Certains hôpitaux américains ont déjà mis en place de logiciels d'aide au diagnostic médical, basés sur des réseaux de neurones. Ceux-ci mettent en correspondance une liste des

paramètres propres au patient (âge, sexe…) et de symptômes décelés, ainsi que les maladies potentielles avec leurs traitements associés.

I.2.5 Les problèmes des systèmes d'aide à la décision

Les systèmes d'aides à la décision présente quelques insuffisances :

- Interface homme machine
  - Problème de la saisie des faits observé chez le patient
  - Langage naturel /Langage artificiel
  - Connexion avec le dossier médical
- Représentation et acquisition des connaissances
  - Difficulté d'obtenir des probabilités fiables
  - Difficulté de mobiliser des experts
  - Faible reproductibilité des experts
- Evaluation et validation des systèmes
  - Problème méthodologiques : validation de la base de connaissance, validation du moteur, validation des résultats
  - Absence de solution de référence
  - Favorise une formation continue de chacun

I.2.6 Exemples de système d'aide à la décision

- Aide au diagnostic des douleurs abdominales de Dombal et coll
  - Le système, de nature probabiliste, a comme objectifs une aide pragmatique dans le cadre de la prise en charge des douleurs abdominales afin d'éviter les gestes inutiles ou les complications. L'évaluation montre que le médecin sans l'aide du système pose moins souvent le bon diagnostic que le médecin assisté du système. Complications et gestes inutiles sont également moins fréquents lorsque le système est utilisé.
- Aide au diagnostic et à la thérapeutique en médecine interne
  - Internist au état unis et ADM en France

- Remarquable quant au domaine couvert et aux résultats proposés. Systèmes utilisant une combinaison de méthodes

- Aide à la chimiothérapie
  - ONCOCIN et PDQ un essai HOLYMPHE
  - Aide au choix du protocole de chimiothérapie, à son respect et à son adaptation temporelle
- HELP
  - Contrôle de la prescription dans le cadre d'un SIH avec système d'alerte.
  - Utilisé en permanence à salt lake city
- Pharmacocinétique et aide au calcul de la posologie

I.3 La classification dans les systèmes ADM

La classification rassemble une famille de méthodes permettant d'automatiser le processus de reconnaissance. Elle peut être considérée comme un domaine per- mettant de définir des algorithmes et techniques susceptibles de classer des objets dont l'aspect est variable par rapport à un objet type. Le but de la classification est donc de classer des observations sur la base d'une série de caractéristiques prédéfinies par apprentissage. Aujourd'hui on ne peut pas parler d'une théorie de la classification d'objets, mais d'approches différentes dont le but commun est de mettre en évidence, de façon automatique, une structure commune à l'ensemble des objets en présence, par recherche de similarités. Le champ d'application de la classification est particulièrement vaste. Il couvre principalement le domaine du traitement des signaux et d'images. On distingue habituellement deux types de classification :

- la classification avec ensemble d'apprentissage connu et parfaitement expertisé, ou classification en mode supervisé;
- la classification en mode non supervisé dans ce dernier cas, on ne dispose pas d'un ensemble d'apprentissage probablement expertisé.

I.3.1 Chaine de classification d'objets

I.3.1.1 Les principaux modules de classification d'objets

Un dispositif de classification ou de reconnaissance automatique de formes est généralement conçu comme une chaine de modules de traitement **[Belaid 19921].**

Ainsi, un système de reconnaissance de formes comporte habituellement les modules suivants.

- **Un module d'acquisition** : des capteurs mesurent des grandeurs caractéristiques de l'objet â classé (signal, image, ...). Cet ensemble de grandeurs constitue la première représentation de l'objet.
- **Un module de prétraitement** : il peut être judicieux de modifier des grandeurs brutes issues des capteurs par un algorithme afin de tenir compte des connaissances qui peuvent être disponible a priori sur le problème. Par exemple à partir de la réponse d'un capteur on peut appliquer un ensemble de filtres destinés à éliminer des bruits indésirables. Ainsi, on obtient une nouvelle représentation de l'objet, plus adéquate pour la classification envisagée. D'autres modules de traitement peuvent élaborer des représentations successives de l'objet; ces différentes représentations ont généralement pour objectif de réduire la dimension de la représentation, c'est à dire de diminuer le

nombre de descripteurs de l'objet et d'élaborer des descripteurs de plus en plus pertinents pour la tâche de discrimination à accomplir.

- **Un module de classification** : l'algorithme de classification considère la dernière représentation de l'objet et décide d'affecter celui-ci à une classe. Cet algorithme peut fournir soit une réponse binaire à valeurs discrètes (appartenance ou non à une classe) soit une réponse à valeurs continues.

La Figure 1.2 illustre une chaine de classification comportant un seul module de prétraitement. On distingue les trois modules et les représentations successives de l'objet. Naturellement, on peut imaginer un dispositif sans module de prétraitement; dans ce cas l'algorithme de classification travaille directement sur des grandeurs relevées par les capteurs.

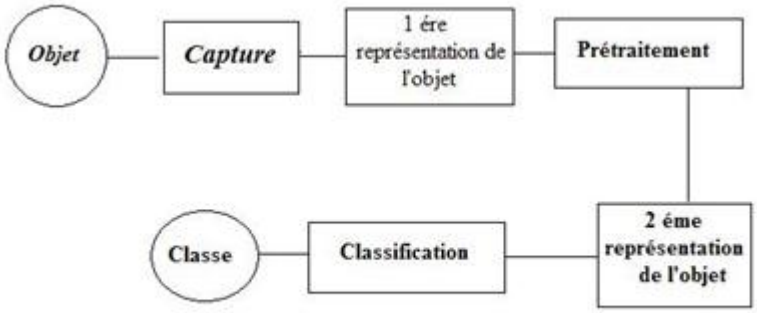


Figure. 1.2 : Chaine de module de classification

La tâche de l'algorithme de classification est d'autant plus aisée que la représentation de l'objet est pertinente. Par exemple pour la classification des battements ventriculaires prématurés (Bvp), chaque cycle cardiaque est entièrement défini par les paramètres de l'ECG. Si les modules d'acquisition (électrodes) ou de prétraitement (filtrage et détection) ne fournissent pas des descripteurs pertinents à l'algorithme de classification, celui-ci ne pourra pas faire de miracle et distinguer les différentes arythmies.

I.3.1.2 l'extraction des descripteurs

Habituellement une classification de formes ne se fait pas directement sur des formes brutes (morphologie de l'ECG par exemple), mais plutôt à partir de descripteurs ou paramètres caractérisant les formes. Chacune des N formes Fi est représentée par un point Xi ($1 \leq i \leq$ N) dans un espace à n dimensions qui est l'espace des paramètres. Il existe deux approches pour caractériser une forme l'approche purement mathématique (analyse en composantes principales. prédiction linéaire, transformée de Fourier, etc...), qui consiste à retenir comme paramètres.il existe deux approches pour caractériser une forme :

- L'approche purement mathématique (analyse en composants principales, prédiction linéaire, transformée de Fourier, etc…), qui consiste à retenir comme paramétras certains coefficients pertinents,
- l'approche intuitive, qui laisse au spécialiste le soin de définir les descripteurs qui lui semblent importants. Cette approche donne souvent de meilleurs résultats, car les paramètres choisis résultent. D'une grande expérience et peuvent être plus discriminants.

I.3.1.3 Procédure de résolution par apprentissage

La classe reçoit une définition qui peut être une définition purement descriptive ou une interprétation, par exemple du type diagnostic médical. Le classifieur qui réalisera le classement des formes doit passer par deux phases **[Leschi 1991],** une phase d'apprentissage et une phase de test.

- **Phase d'apprentissage**

Le but de l'apprentissage est de découvrir les règles (généralement non déterministes) qui gouvernent et régissent des formes. L'apprentissage est un processus calculatoire qui doit être capable d'amener à une certaine prédiction et à une certaine généralisation. Il existe principalement deux types d'apprentissage, supervisé et non supervisé **[Milgram 1993].** Dans le premier cas, on doit apprendre des associations (individus, classes) ; dans le deuxième cas on ne fournit pas d'indications sur les classes.[voir chapitre2]

- **Phase de test**

Cette phase doit permettre l'affectation d'un nouvel objet à l'une des classes, au moyen d'une règle de décision intégrant les résultats de la phase d'apprentissage **[Leschi**

1991]. L'objectif est d'obtenir une estimation la plus fidèle possible du comportement du classifieur dans des conditions réelles d'utilisation. Pour cela, des critères classiques comme les taux de classification et les taux d'erreur sont presque systématiquement utilisés. Mais d'autres critères, comme la spécificité et la sensibilité, apportent aussi des informations utiles.

- *Taux de classification et taux d'erreurs* [**Chikh, 2005]**

Les taux de classification et d'erreurs permettent d'évaluer la qualité du classifieur C par rapport au problème pour lequel il a été conçu. Ces taux sont évalués grâce à une base de test qui contient des formes décrites dans le même espace de représentation E que celles utilisées pour l'apprentissage. Elles sont aussi étiquetées par leur classe réelle d'appartenance afin de pouvoir vérifier les réponses du classifieur. Pour que l'estimation du taux de reconnaissance soit la plus fiable possible, il est important que le classifieur n'ait jamais utilisé les échantillons de cette base pour faire son apprentissage (la base de test ne doit avoir aucun objet en commun avec la base d'apprentissage et les éventuelles bases de validation). De plus, cette base de test doit être suffisamment représentative du problème de classification [Guyon 19981. En général, quand les échantillons étiquetés à disposition sont suffisamment nombreux, ils sont séparés en deux parties disjointes et en respectant les proportions par classes de la base initiale. Une partie sert pour former la base d'apprentissage et l'autre pour former la base de test. Le découpage le plus courant est de 2/3 pour l'apprentissage et le 1/3 restant pour la base de test. Les performances en terme de taux de classification sont alors déterminées en présentant au classifieur chacun des exemples e_j de la base de test et en comparant la classe donnée en résultat C (e_j) = s à la vraie classe de e_j. En considérant que la base de test contient N objets et que sur ceux-ci

$N_{corrects}$ sont biens classés par le système, le taux de classification τ_{class} est simplement défini par *:*

$$\mathcal{T}class = \frac{\mathcal{N}\text{crrects} .100}{N}$$

Le taux d'erreur τ_{err} est défini à partir du nombre d'objets N_{err} mal classés *:*

$$\mathcal{T}err = \frac{\mathcal{N}\text{crrects} .100}{N}$$

- *Sensibilité et spécificité*

L'évaluation des performances d'un classifieur peut être réalisée par l'appréciation de deux lois statistiques, qui sont la sensibilité et la spécificité. Pour rappel, ces deux quantités sont définies par :

$$Se(i) = \frac{\mathrm{VP(i)}}{VP\ (i) + \ FN(i)}$$

$$Sp(i) = \frac{\mathrm{VN(i)}}{VN\ (i) + \ FP(i)}$$

Où les grandeurs VP(i), FN(i), VN(i), FP(i) sont définies dans le tableau1.2

	Présence d'événement de classe i	**Absence d'événement de classe i**
Classification Positive	Vrai Positif VP (i)	Faux Positif FP (i)
Classification Négative	Faux Négatif FN (i)	Vrai Négatif VN (i)

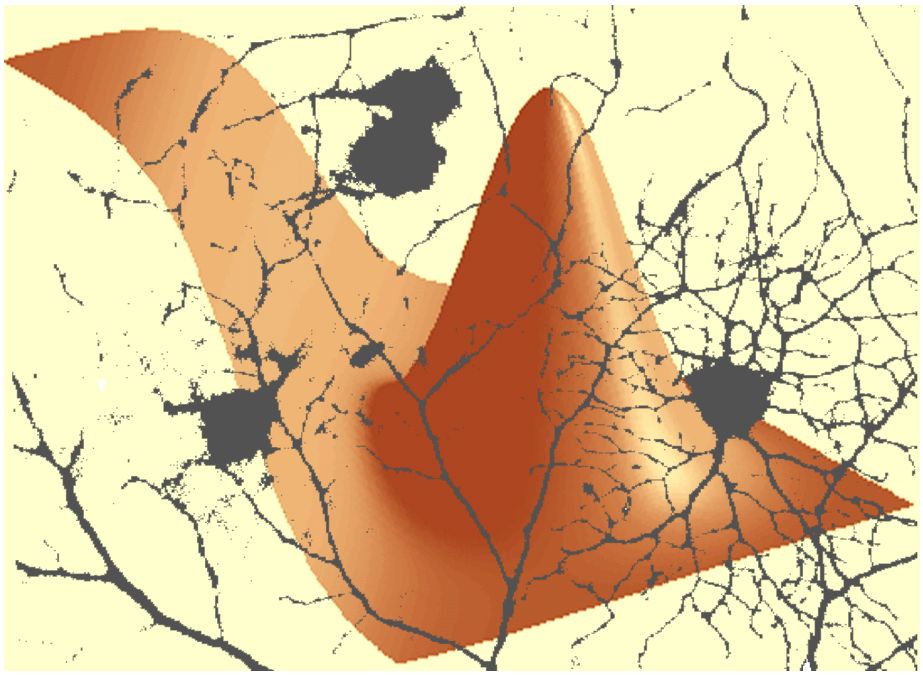
La sensibilité $S_e\ (i)$ représente la probabilité de bonne classification de la classe i et spécifité $S_p\ (i)$ est une mesure indirecte de la probabilité de fausse alarme.

I.4 conclusion

Ce chapitre a permis de définir le système ADM, ainsi de présenter son intérêt dans la médecine et ses différents approches.

Ce chapitre a aussi introduit des notions sur la classification des formes dans les systèmes ADM, surtout la procédure de résolution par apprentissage que nous allons utiliser pour réaliser des classifieurs des signaux bio médicaux.

CHAPITRE 2



réseaux de neurones & apprentissage

II.1 Introduction

Un RNA est un ensemble de neurones formels (voir annexe c) (d'unités de calcul simples, de nœuds processeurs) associés en couches (ou sous-groupes) et fonctionnant en parallèle.

Les réseaux de neurone sont des modèles mathématiques et informatiques, des assemblages d'unités de calculs appelés neurone formels (voir annexe c), et dont l'inspiration originale était un modèle de la cellule nerveuse humain (voir annexe a)

Dans un réseau, chaque sous-groupe fait un traitement indépendant des autres et transmet le résultat de son analyse au sous-groupe suivant. L'information donnée au réseau va donc se propager couche par couche, de la couche d'entrée à la couche de sortie, en passant soit par aucune, une ou plusieurs couches intermédiaires (dites couches cachées). Il est à noter qu'en fonction de l'algorithme d'apprentissage, il est aussi possible d'avoir une propagation de l'information en sens inverse ("back propagation"). Habituellement (excepté pour les couches d'entrée et de sortie), chaque neurone dans une couche est connecté à tous les neurones de la couche précédente et de la couche suivante.

Les RNA ont la capacité de stocker de la connaissance empirique et de la rendre disponible à l'usage. La connaissance du réseau va être stockée dans les poids synaptiques, obtenus par des processus d'adaptation ou d'apprentissage. En ce sens, les RNA ressemblent donc au cerveau car non seulement, la connaissance est acquise au travers d'un apprentissage mais de plus, cette connaissance est stockée dans les connexions entre les entités soit, dans les poids synaptiques.

Ce chapitre présente les différentes structures des RNA, ainsi ses différents types d'apprentissage et ses problèmes

II.2 Structure d'interconnexion

Les connexions entre les neurones qui composent le réseau décrivent la topologie du modèle.

On appelle topologie des réseaux la façon dont les neurones sont connectés entre eux à travers les différentes couches. En général on peut distinguer deux grands groupes des réseaux de neurones selon leur topologie : **les réseaux à couches** et **les réseaux récurrents** (cf. figure suivante).

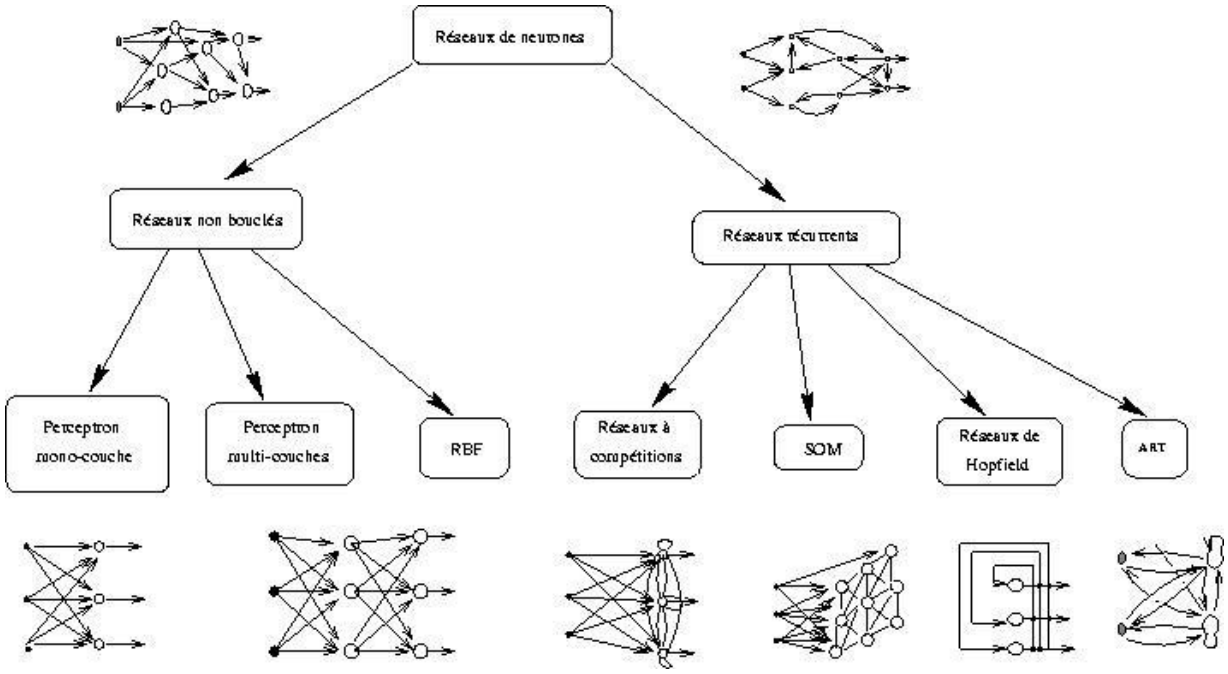


Figure 2.1 : représentation des différentes topologies de R.N.A

II.2.1 les réseaux bouclés (ou réseaux non récurrents) :

les réseaux bouclés des réseaux de neurones dans lesquels l'information se propage couche par couche sans retour en arrière possible ; Les réseaux à couches peuvent se diviser en réseaux sans couches cachées et en réseaux avec une ou plusieurs couches cachées.

Nous allons dans les paragraphes suivant voir plus en détails les réseaux à couches.

II.2.1.1 Le perceptron monocouche

Historiquement le premier RNA, c'est le Perceptron de Rosenblatt. C'est un réseau simple, puisque il ne se compose que d'une couche d'entrée et d'une couche de sortie.

Les réseaux sans couches cachées, sont les réseaux les plus simples. Ils sont utilisables pour des problèmes de classification et d'approximation. Leur avantage est que l'apprentissage du réseau converge vers une solution optimale. Cela est dû au fait que c'est un système linéaire. Leur inconvénient est qu'ils peuvent seulement classifier ou approximer les

problèmes linéaires et ne peuvent résoudre un problème non linéaire. La figure suivante décrit un exemple d'une classification en utilisant une fonction linéaire. On voit que dans le cas non linéairement séparable (figure de droite), on doit utiliser une fonction non linéaire, qui est dessinée sous la forme d'une courbe

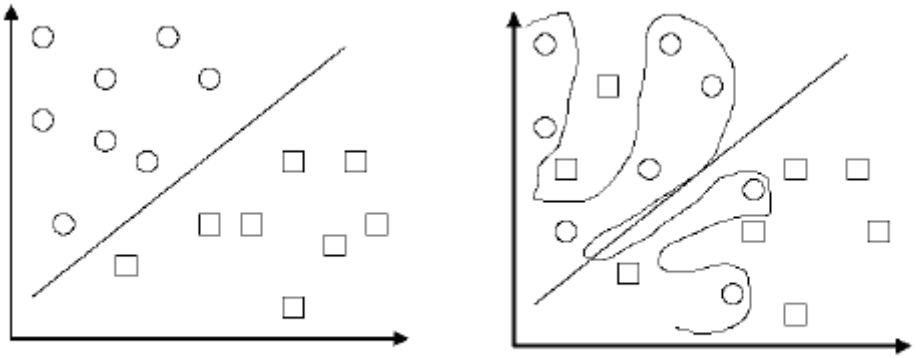


Figure2.2 : **classification par une fonction linéaire**

L'exemple classique pour un système de neurones monocouches est le Perceptron monocouche, inventé par Rosenblatt .C'est un modèle très simple, basé sur l'orientation physico physiologique. Il ne dispose que deux couches :

- Une couche d'entrée qui s'appelle la rétine et qui est une aire sensorielle ;
- une couche de sortie qui donne la réponse correspondante à la simulation présentée à l'entrée.

Le fonctionnement est le suivant : une donnée est présentée au réseau en activant la rétine. L'activation se propage vers la couche de sortie où on peut noter la réponse du système. Cette réponse suit la formule suivante :

$$y = \varphi(\sum_{j=1}^{2} w_j x_j + \Theta) \text{ où } \varphi \text{ est la fonction d'activation utilisée,}$$

La figure suivante montre la structure de Perceptron monocouche, avec une sortie *y* et deux entrées 1 *x* et 2 *x* qui forme la rétine du réseau.

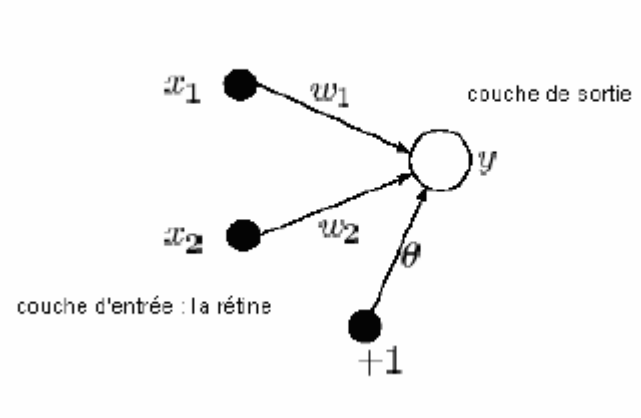


Figure2.3 : structure d'un Perceptron monocouche

II.2.12 Les perceptrons multi couche

Un neurone de perceptron réalise un produit scalaire entre son vecteur d'entrées $\mathbf{x}$ et un vecteur de paramètres $\mathbf{w}$appelé poids, y ajoute un biais b, et utilise une fonction d'activation $f$pour déterminer sa sortie :

$$y = f(\mathbf{x} \cdot \mathbf{w} + b)$$

Equation 2.1 : Expression de la sortie d'un neurone de perceptron

Les fonctions d'activation doivent être de préférence strictement croissantes et bornées. Les fonctions classiquement utilisées sont la fonction linéaire, la tangente hyperbolique (f_1) et la fonction sigmoïde standard (f_2) :

$$f_1(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$
$$f_2(x) = \frac{tanh(x) + 1}{2}$$

Equation 2.2 : Expression du sigmoïde standard

La différence entre ces deux dernières fonctions est le domaine des valeurs prises, qui est de $]-1;1[$ pour la tangente hyperbolique et de $]0;1[$ pour la sigmoïde standard. La figure 2.4 montre la courbe de la tangente hyperbolique

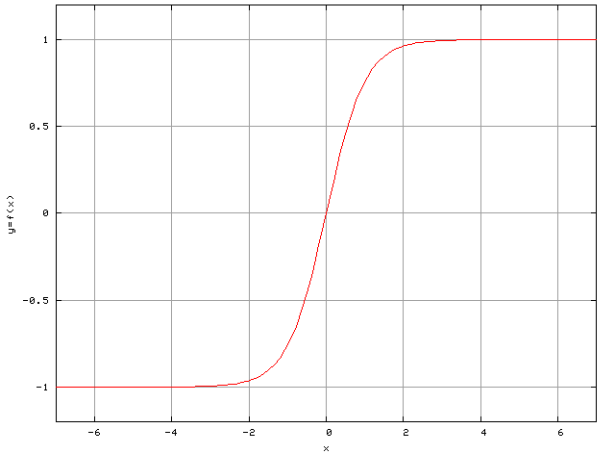


Figure2.4 : Tangente hyperbolique

Le perceptron est organisé en plusieurs couches. La première couche est reliée aux entrées, puis ensuite chaque couche est reliée à la couche précédente. C'est la dernière couche qui produit les sorties du PMC. Les sorties des autres couches ne sont pas visibles à l'extérieur du réseau, et elles sont appelées pour cette raison couches cachées.

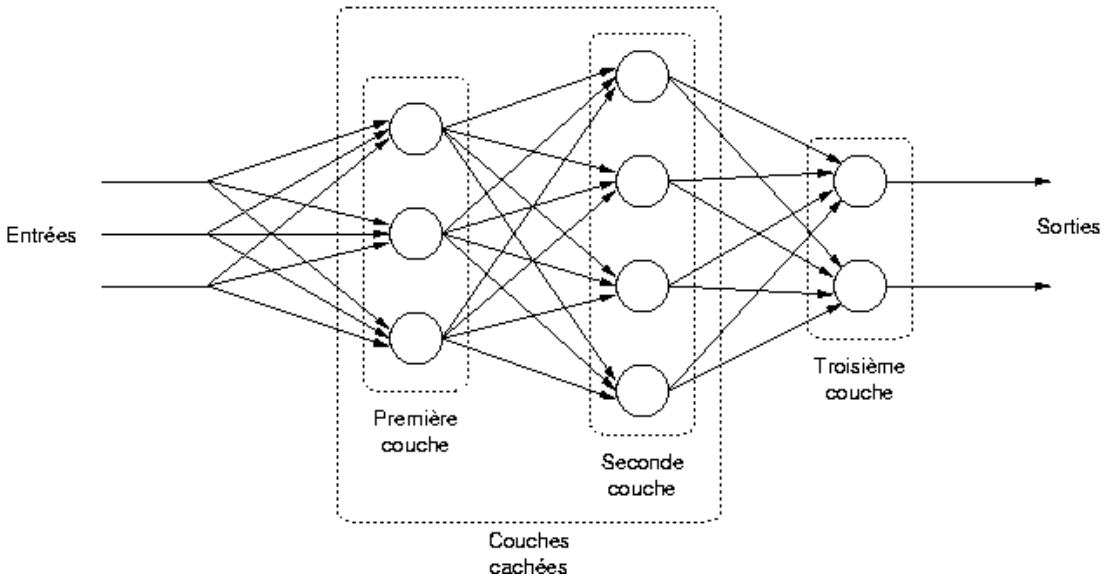


Figure2.5 : perceptron multicouche

Notons N_c le nombre de couches. L'indice l servira à désigner une couche, et notons n_l le nombre de neurones dans la couche l. L'indice i désigne un neurone. Le vecteur de poids du neurone i de la couche l est noté $\mathbf{w_{l,i}}$, et tous les vecteurs de poids d'une couche sont regroupés dans une matrice $\mathbf{W_l}$. En ce qui concerne les biais, on les considère souvent comme un poids supplémentaire associé à une entrée qui est toujours à 1. Ceci revient exactement au même, mais permet de simplifier les notations, et c'est donc ce que nous ferons dans le reste de ce mémoire. Notons $\mathbf{a_l}$ le vecteur regroupant les sorties des neurones de la couche l. Comme chaque couche est reliée à la précédente, $\mathbf{a_l}$ est également l'entrée de la couche $l+1$, en ajoutant l'entrée supplémentaire à 1 pour le biais. Par extension notons $\mathbf{a_0}$ l'entrée du réseau.

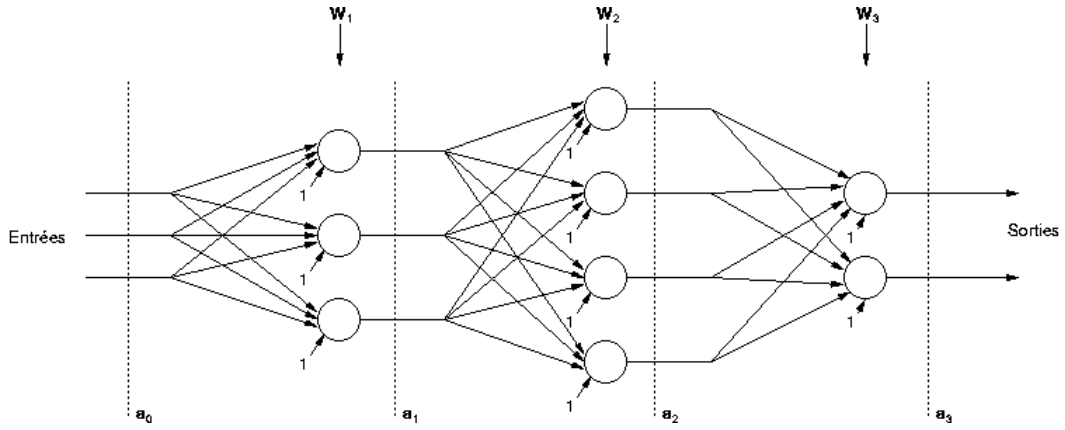


Figure2.6 : Notation des poids et des sorties des couches

Nous allons considérer de plus que tous les neurones d'une couche ont la même fonction d'activation, mais qu'elle peut différer d'une couche à l'autre. La fonction d'activation des neurones de la couche l est notée f_l et on définit une extension vectorielle de f_l par :

$$\mathbf{f_l}(\mathbf{a_l}) = (f_l(a_{l1}), f_l(a_{l2}), \ldots, f_l(a_{ln_l}))$$

Equation 2.3 : Définition vectorielle de la fonction d'activation d'une couche

On peut alors écrire la relation suivante pour exprimer la sortie d'une couche en fonction de son entrée :

$$\mathbf{a_l} = \mathbf{f_l}(\mathbf{a_{l-1}} \cdot \mathbf{W_l})$$

Equation 2.4 : Relation entrée/sortie d'une couche de perceptron

Le calcul de la sortie du perceptron multicouche se fait de manière itérative. Il faut tout d'abord placer les entrées du réseau dans le vecteur $\mathbf{a_0}$, puis appliquer l'équation (1.4) avec $l = 1 \ldots N_c$ afin de calculer successivement $\mathbf{a_1}, \mathbf{a_2} \cdots, \mathbf{a_{N_c}}$. La sortie du réseau est alors $\mathbf{a_{N_c}}$.

Les fonctions qu'il est possible de réaliser avec un PMC sont diverses. Dans un perceptron à une couche, il n'y a pas de couche cachée, et l'unique couche relie les entrées du réseau aux sorties. Si la fonction d'activation utilisée est une sigmoïde, chaque sortie est une sigmoïde de produit scalaire. L'espace d'entrée est donc coupé en deux par un hyperplan. La sortie est égale à 1 d'un côté de l'hyperplan et à -1 de l'autre côté, lorsque l'on est situé à une certaine distance de celui-ci. Pour des points situés près de l'hyperplan, la transition est progressive. Dans l'exemple simple d'un réseau à deux entrées et une sortie composé d'un unique neurone, l'hyperplan est une droite. La figure suivante montre la sortie de ce neurone (axe vertical) en fonction de ses deux entrées (axes horizontaux) :

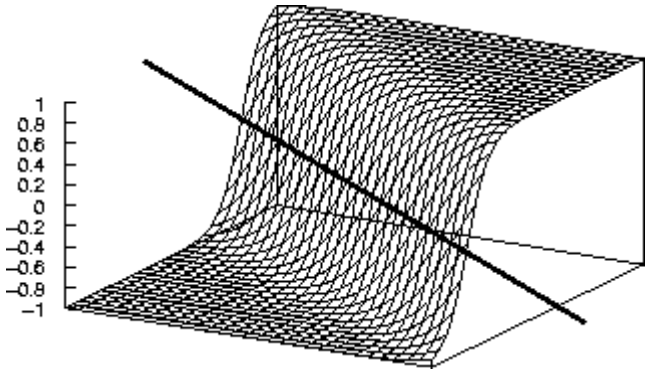


Figure2.7 : Sortie d'un perceptron à une couche en deux dimensions

Dans ce neurone, le biais fixe la distance entre l'origine et la droite (montrée figure précédente), tandis que le vecteur poids (à deux dimensions) est orthogonal à la droite, et donc fixe sa direction, ainsi que la "largeur" de la zone de transition : plus le module du vecteur poids est élevé, plus la sortie évoluera rapidement de -1 à 1 en traversant la droite.

Dans un perceptron à deux couches, les sorties du réseau seront des combinaisons des sorties de la première couche, et on voit apparaître des intersections entre les zones définies par les neurones de la première couche.

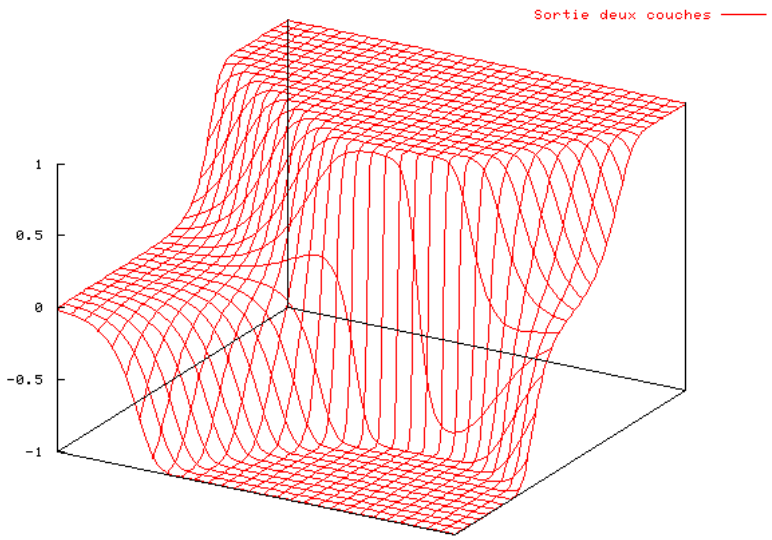


Figure2.8 : Sortie d'un perceptron à deux couches en deux dimensions

De manière plus analytique on peut considérer la sortie d'une couche (équation (2.4) comme une fonction vectorielle, dont les éléments constituent une base de fonctions. En ajoutant une seconde couche, les sorties sont des combinaisons des différents éléments de cette base, qui peuvent éventuellement servir à nouveau de base pour une autre couche. Il y a une ressemblance entre une couche de PMC, qui permet une décomposition dans une base de fonctions tangentes hyperboliques et une couche de GRBF, qui elle permet une décomposition dans une base de fonctions gaussiennes.

Il a été montré qu'un perceptron à deux couches avec des fonctions d'activation intégrables au sens de Riemann non polynomiales sur la première couche et une fonction d'activation linéaire sur la seconde est un approximateur universel .Comme pour le GRBF, ceci veut dire que le réseau est capable d'approximer n'importe quelle fonction douce avec une précision donnée, pourvu que l'on fournisse un nombre suffisant de neurones dans la couche cachée. Cependant en pratique il n'est pas forcément possible d'approximer toute fonction, car dans certains cas le nombre de neurones nécessaire peut être très important, et le théorème dans l'article cité ne garantit pas que l'algorithme d'apprentissage pourra converger vers le résultat souhaité.

II.2.1.3 Le réseau à fonction radiale (RBF) :

Le réseau RBF est un réseau de neurones supervisé. Il s'agit d'une ' **spécialisation** ' d'un PMC. Il a la même structure que le Perceptron multicouches, mais la fonction d'activation est une fonction de type Gaussienne. Ce réseau, à cause de son architecture, utilise le plus souvent la règle d'apprentissage de correction d'erreur et la règle par apprentissage compétitif. Il peut avoir un apprentissage qui combine en même temps l'apprentissage supervisé et l'apprentissage non supervisé Ce réseau obtient des performances comparables ou supérieures à ceux du Perceptron multicouches. De plus leur apprentissage plus rapide et plus simple en font des outils de choix pour plusieurs types d'applications, dont la classification et l'approximation des fonctions. Cependant, ce réseau n'a pas si grandes recherches que le Perceptron multicouches.

- ***Architecture***

Un RBF est constitué uniquement de **3 couches** :

- La couche d'entrée : elle retransmet les inputs sans distorsion.
- La couche RBF : couche cachée qui contient les neurones RBF.
- La couche de sortie : simple couche qui contient une fonction linéaire.

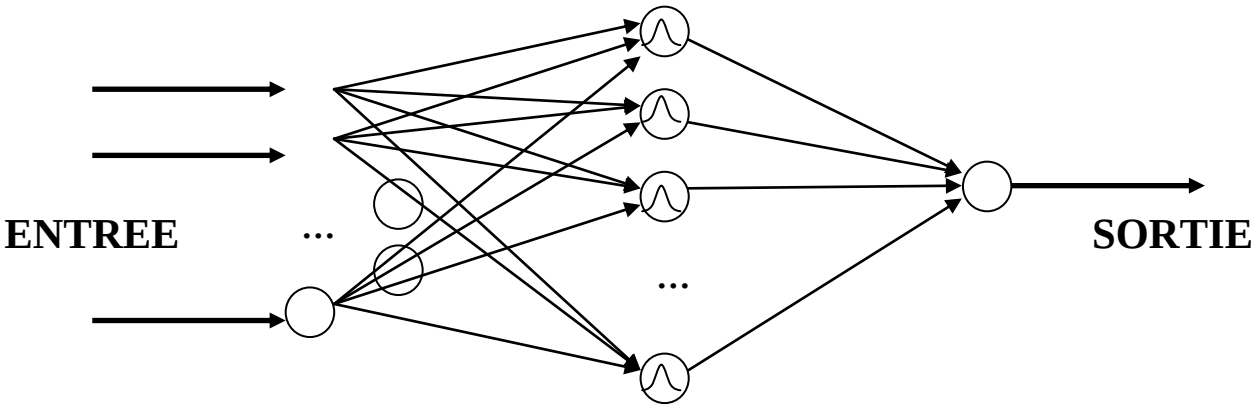


Figure2.9: architecture d'un réseau à fonction radiale (RBF)

- Chaque neurone RBF contient une **gaussienne** qui est centrée sur un point de l'espace d'entrée.
- Pour une entrée donnée, la sortie du neurone RBF est la **hauteur** de la gaussienne en ce point.

- La fonction gaussienne permet aux neurones de ne répondre qu'à une petite **région** de l'espace d'entrée, région sur laquelle la gaussienne est centrée.

$$F(x) = EXP(x^2 / (2* Beta^2))$$

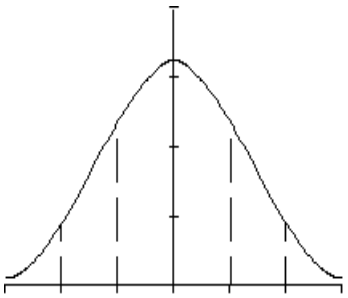


Figure2.10: fonction gaussienne

II.2.2 les réseaux non bouclés (ou les réseaux récurrents)

Sont des réseaux de neurones dans lesquels il y a une liaison vers l'arrière. Les connexions de ces réseaux forment des boucles. Ainsi la fonction d'activation peut circuler le long de ces boucles et affecter le réseau pendant une période arbitrairement longue. Pour cette raison les comportements des réseaux récurrents sont potentiellement plus complexes que ceux des réseaux à couches.

En général, les réseaux bouclés sont plus difficiles à entraîner que les réseaux non bouclés. L'apprentissage est cependant généralement assez complexe dans ces réseaux, et leurs propriétés sont souvent moins bien connues que celles des réseaux non bouclés.

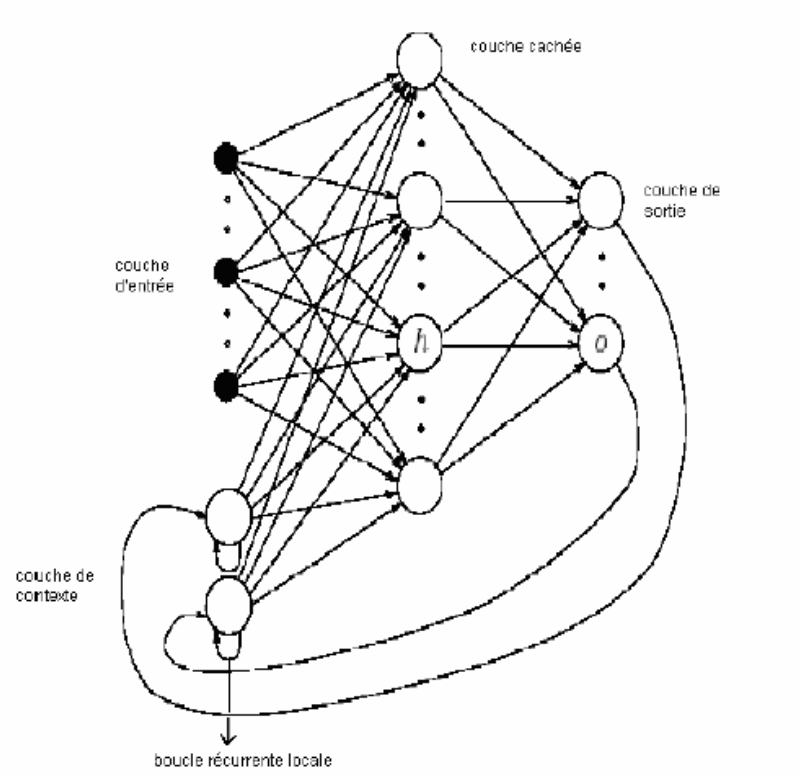
Pendant les dernières années, depuis le premier réseau récurrent de Hopfield, beaucoup d'intérêt a été accordé aux réseaux récurrents, en raison de leur aptitude à gérer des taches complexes et de la richesse de leur comportement dynamique**.**

Trois classes importantes de réseaux récurrents sont présentées dans ce paragraphe :

- **les réseaux récurrents à couches** (le réseau de Elman et le réseau de Jordan).
- **les réseaux à compétition** (réseaux ART et réseaux de Kohonen, « Self-organisation mapping »).
- **les réseaux à connexions symétriques** (réseau de Hopfield).

II.2.2.2 Les réseaux récurrents à couches

- **Le réseau de Jordan** [est le réseau récurrent le plus ancien. On présente la structure de ce réseau sur la figure suivante



:

Figure 2.11 : **architecture de réseau de Jordan**

Ce réseau a pour but d'effectuer une séquence d'actions par rapport à une tâche donnée par l'utilisateur. La tâche est constante durant l'exécution de la séquence ; le réseau doit toutefois retenir sa position dans la séquence. Pour mener sa tâche à bien, il lui faut donc une mémoire du contexte représentée par une couche nommée « couche de contexte ». Les neurones de sorties sont reliés vers cette couche. Le nombre de neurone de cette couche dépend du nombre des couches de sortie. Elle retient l'état du réseau au temps précédent et sa propre activation par une boucle récurrente locale. Il en résulte que la couche de contexte retient une trace des événements passés. L'apprentissage a lieu aussi bien au niveau des connexions entre les couches d'entrée et les couches cachées qu'entre les couches cachées et les couches de sortie. Ainsi on peut utiliser toutes les règles d'apprentissage décrit par un Perceptron multicouche pour entraîner ce réseau.

Le réseau d'Elman

Le réseau d'Elman a été introduit par Elman à 1990 et est présenté sur la figure ci-après. Ce réseau est très semblable au réseau de Jordan, mais son architecture est plus adaptée au traitement de séquence structurée. Là où le réseau de Jordan bouclait ses sorties sur ses entrées, le réseau d'Elman fait boucler sa couche cachée sur elle-même. L'effet de cette boucle est de recopier sans autre traitement la valeur de la couche cachée dans la couche de contexte, de telle sorte que l'état de la couche de contexte à un temps *t* est égal à l'état de la couche cachée au temps (*t-1*). Dans le réseau d'Elman, comme dans le réseau de Jordan, cette boucle récurrente permet de retenir une trace d'événements passés. Pour l'apprentissage du réseau on applique une rétro propagation classique pour corriger les poids, sans tenir compte de la recopie de l'activation de la couche cachée vers la couche de contexte.

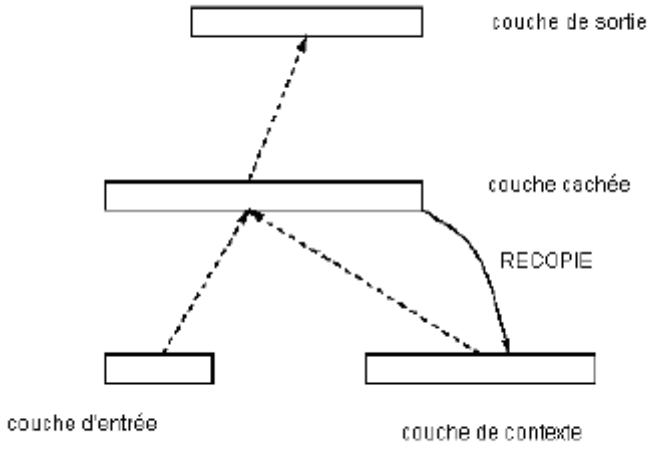


Figure 2.12 : réseaux d'Elman

III.2.2.2 Les réseaux à compétition

Les réseaux à compétition sont composés d'une couche de neurones qui reflète passivement les données d'entrée présentée au réseau, et une couche de neurones de sortie en compétition. On utilise le réseau de la façon suivante : une donnée d'entrée est présentée au réseau, provoquant des réponses variées de la part des neurones de sortie. La compétition s'installe alors entre ces derniers, et prend la forme d'un combat d'influence qui doit éventuellement se stabiliser, grâce à la force des liens inhibiteurs. A la fin de la compétition, le ou les neurones de sortie les plus activés sont déclarés "vainqueurs". Des

exemples de réseaux ART (Adaptive Resonance Theory) et de réseaux de Kohonen (Self-organisation mapping) sont présentés ci-après.

Les réseaux ART

Les réseaux ART sont des réseaux à compétition. Le problème majeur qui se pose dans ce type de réseaux est le dilemme stabilité/plasticité. En effet dans un apprentissage par compétition, rien ne garantit que les catégories formées vont rester stable. La seule possibilité pour assurer une stabilité, serait que le taux d'apprentissage tende vers zéro, mais alors le réseau perd sa plasticité. C'est pour résoudre ce problème, qu'on utilise les réseaux ART. Les valeurs de poids ne sont adaptées que s'il y a une approximation suffisante entre l'entrée du réseau et le prototype déjà connu par le réseau. Alors il existe une résonance. Sinon, l'entrée du système est très éloignée du prototype, on fonde alors une nouvelle classe. Le mode d'apprentissage peut être supervisé ou non. On va présenter deux types de ces réseaux :

- ✓ **ART-1 :** est un système binaire dont la fonction est d'associer à une donnée d'entrée binaire un seul neurone de sortie. Les autres membres de cette famille ne sont pas si limités. Le but le plus connu de ce réseau est de modéliser la reconnaissance d'un objet perçu, ainsi que de stocker une information. Mais ces réseaux sont très sensibles aux bruits qui apparaissent sur les données. Pour y échapper, on doit mettre en place une couche de filtrage.

- ✓ **ART-2** : traite des valeurs continues, mais il a le même but que le réseau précédent. Cependant les calculs utilisés sont beaucoup plus difficiles à comprendre.

Les réseaux de Kohonen (Self organisation mapping) :

Utilisent la carte de Kohonen. Cette carte est composée de deux couches dont la première constitue l'entrée du réseau et la seconde la sortie. On peut voir la structure de cette carte sur la figure ci-après. Les neurones de cette carte sont disposés en une topologie déterminée. Plusieurs topologies sont possibles, par exemple une carte carré ou un carte rectangulaire.

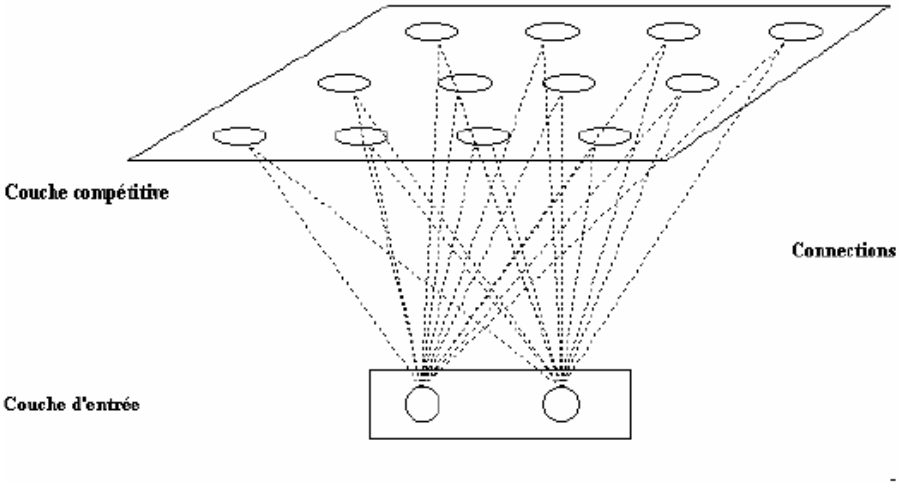


Figure 2.13 : réseau de Kohonen

II.2.2.3 Réseau de Hopfield :

Les réseaux de Hopfield sont des réseaux récurrents et entièrement connectés. Dans ce type de réseau, chaque neurone est connecté à chaque autre neurone et il n'y a aucune différenciation entre les neurones d'entrée et de sortie. Ils fonctionnent comme une mémoire associative non linéaire et sont capables de trouver un objet stocké en fonction de représentations partielles ou bruitées. L'application principale des réseaux de Hopfield est l'entrepôt de connaissances mais aussi la résolution de problèmes d'optimisation. Le mode d'apprentissage utilisé ici est le mode non supervisé.

II.2.3 Choix de l'architecture

Il existe un grand nombre de type de réseaux de neurones, avec pour chacun des avantages et des inconvénients. Le choix d'un réseau peut dépendre :

- De la tache à effectuer (classification, association, contrôle de processus, séparation aveugle de sources…..).
- De la nature des données (variante dans le temps, nom variante...)
- D'éventuelles contraintes d'utilisation temps réel (certains types de réseaux de neurones, tels que la « machine de bolzmann », nécessitant des tirages aléatoires et un nombre de cycles de calculs indéfini avant stabilisation du résultat en sortie, présent plus de contraintes que d'autre réseaux pour une utilisation en temps réel)
- Des différents types de réseaux de neurones disponibles dans le logiciel de simulation que l'on compte utiliser (à moins de le programmer)

Ce choix est aussi fonction de la maitrise ou de la connaissance que l'on a de certains réseaux, ou encore du temps dont on dispose pour tester une architecture prétendue plus performante.

II.3 Apprentissage des réseaux de neurones

L'apprentissage est une phase du développement d'un réseau de neurones durant laquelle le comportement du réseau est modifié jusqu'à l'obtention du comportement désiré.

L'apprentissage s'effectue grâce à des algorithmes itératifs, appelé « algorithme d'apprentissage », qui modifient les poids des connexions pour les adapter aux données présentées au réseau et sont caractéristiques du système à modéliser.

Au niveau des algorithmes d'apprentissage, il a été défini deux grandes classes selon que l'apprentissage est dit supervisé ou non supervisé

II.3.1 Apprentissage supervisé

L'apprentissage est dit supervisé si les différentes familles de formes, ou classe, sont connues a priori ainsi que l'affectation de chaque forme à telle ou telle famille.

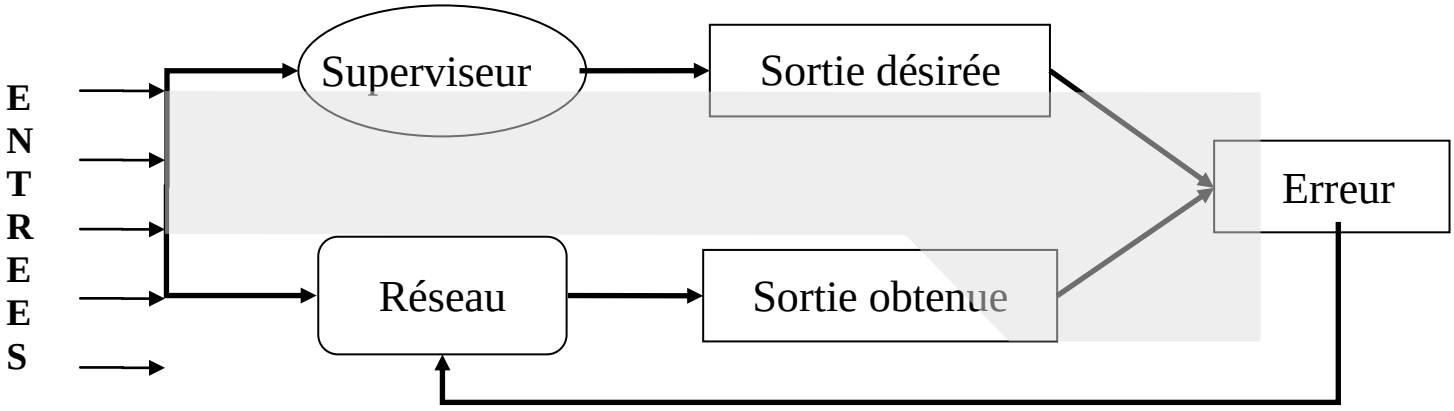


figure2.14 : présentation d'un apprentissage supervisé

Le réseau compare sortie obtenue qu'il a calculé, en fonction des entrées *u* fournies, et la réponse *désirée* attendue en sortie. Ainsi le réseau va se modifier jusqu'à ce qu'il trouve la bonne sortie *désirés*, c'est-à-dire celle attendue, correspondante à une entrée *u* donnée. Les différentes réponses sont connues à priori. On dispose d'une base d'apprentissage qui contient un ensemble d'observation sous forme des couples entrées/sorties associées. Les poids sont modifiés en fonction des sorties désirées. Cet apprentissage est appliqué généralement pour les réseaux non bouclé

Cet apprentissage exploite en plus plusieurs idées simples :

- *La minimisation itérative d'un critère de l'erreur en sortie du réseau* : on initialise la matrice des poids au hasard, puis l'on fait évaluer ces matrices de manière à ce qu'elles autorisent l'association souhaite, c'est à dire jusqu'à ce qu'un critère de l'erreur (entre les sortie réellement obtenues et les sorties souhaitées) soit quasi nul.
- *La quantification vectorielle* : on cherche à regrouper de vecteurs qui se ressemblent sur des sous-réseaux identifiés.
- *L'approximation vectorielle* : on ne cherche pas à faire réaliser l'approximation d'une fonction par un seul réseau, mais par plusieurs modules reliés entre eux.
- *La construction ad hoc du réseau, généralement inspirée par la règle de Hebb : on applique une formule de construction de la ou des matrices de connexions utilisant les couples d'apprentissage. La solution obtenue est alors directement implantée dans le réseau neuronal.*

II.3.1.1 Minimisation itérative d'un critère de l'erreur en sortie

Il s'agit de minimiser une « fonction de cout » de l'erreur calculée entre la sortie réelle du réseau et sa sortie désirée, pour une entrée donnée. Or, les problèmes de minimisation d'une fonction de cout relèvent des méthodes de recherche opérationnelle : il s'agit tout simplement d'un problème d'optimisation.

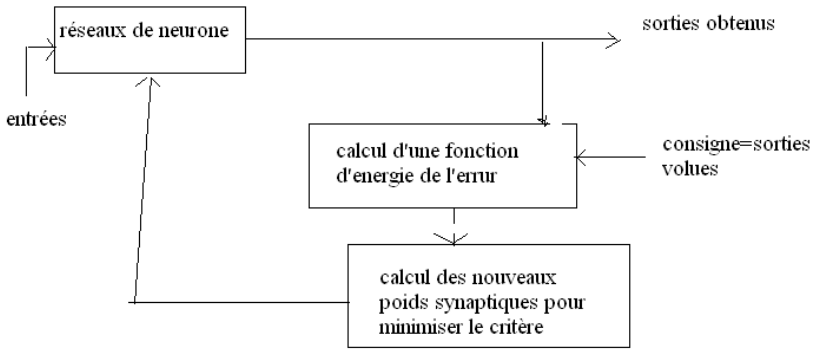


Figure2.15 : minimisation de l'erreur

Fonction de cout

La définition d'une fonction de coût est primordiale, car celle-ci sert à mesurer l'écart entre la sortie désirée du modèle et les mesures faites sur les exemples d'apprentissage.

La fonction la plus couramment utilisée, et dont nous nous sommes servi lors de nos travaux, est la fonction dite « Erreur quadratique ».

L'erreur quadratique sur la base d'apprentissage consiste à minimiser la somme des carrés des erreurs entre la sortie du réseau et la valeur réelle de la sortie.

Les algorithmes utilisés nécessitent que la fonction de coût soit dérivable par rapport aux poids.

L'apprentissage supervisé est le type d'apprentissage le plus utilisé. Pour ce type la règle la plus utilisée est celle de Widrow-Hoff (la règle Delta). D'autres règles d'apprentissage sont par exemple la règle de Hebb, la règle du perceptron (procédure de correction d'erreur), et la rétro propagation de gradient qui est l'algorithme le plus répandu pour les réseaux multi couches.

a. ***Règle de Hebb :*** Cett***e*** règle permet de modifier la valeur des poids synaptiques en fonction de l'activité des unités qui les relient. Le but principal est le suivant : Si deux unités connectées sont actives simultanément, le poids de leur connexion est augmenté ou diminué. R est une constante positive qui représente la force d'apprentissage (learning rate).

b. **La règle delta :** qui calcule la différence entre la valeur de la sortie et la valeur désirée pour ajuster les poids synaptiques. Elle emploie une fonction d'erreur, nommée aussi le moindre carré moyen, basée sur les différences utilisées pour l'ajustement des poids.

c. **Algorithme d'apprentissage par correction d'erreur ou règle du perceptron**

- ***Présentation de l'algorithme***

L'algorithme d'apprentissage peut être décrit de la manière suivante. On initialise les poids du perceptron à des valeurs quelconques. A chaque fois que l'on présente un nouvel exemple, on ajuste les poids selon que le perceptron l'a correctement classé ou non. L'algorithme s'arrête lorsque tous les exemples ont été présentés sans modification d'aucun poids.

La procédure d'apprentissage du perceptron linéaire à seuil est une procédure de correction d'erreur puisque les poids ne sont pas modifiés lorsque la sortie attendue est égale à la sortie calculée par le perceptron courant. Étudions les modifications sur les poids lorsque diffère de :

si la sortie calculée=0 et sortie attendue=1, cela signifie que le perceptron n'a pas assez pris en compte les neurones actifs de l'entrée (c'est-à-dire les neurones ayant une entrée à 1)dans ce cas ;l'algorithme ajoute la valeur de la rétine aux poids synaptiques (*renforcement*).

si la sortie calculée=1 et la sortie attendue=0; l'algorithme retranche la valeur de la rétine aux poids synaptiques (*inhibition*).

Lors de la phase d'apprentissage, sachant que l'échantillon d'apprentissage est un ensemble linéairement séparable, tous les exemples sont présentés jusqu'à la convergence, c'est-à-dire jusqu'à ce qu'une présentation complète des exemples n'entraîne aucune modification des poids.

- ***Critiques sur la méthode par correction d'erreur***

Si l'échantillon est linéairement séparable, si tous les exemples sont présentés équitablement (c'est-à-dire que la procédure de choix des exemples n'en exclut aucun), et que le critère d'arrêt est la stabilité après une présentation complète de l'échantillon alors la procédure d'apprentissage par correction d'erreur converge et l'algorithme s'arrête avec un perceptron qui classifie correctement l'échantillon d'apprentissage.

L'inconvénient majeur de cet algorithme est que si l'échantillon présenté n'est pas linéairement séparable, l'algorithme ne convergera pas et l'on n'aura aucun moyen de le savoir.

On pourrait penser qu'il suffit d'observer l'évolution des poids synaptiques pour en déduire si l'on doit arrêter ou non l'algorithme. En effet, si les poids et le seuil prennent deux fois les mêmes valeurs sans que le perceptron ait appris et alors que tous les exemples ont été présentés, cela signifie que l'échantillon n'est pas séparable.

d. La règle de rétro propagation : (*Backpropagation* en anglais) inventée par Rumelhart, Hinton et Williams en1986 .Elle s'utilise pour ajuster les poids de la couche d'entrée à la couche cachée. Cette règle peut aussi être considérée comme une généralisation de la règle delta pour des fonctions d'activation non linéaire et pour des réseaux multicouches.

La technique de **rétropropagation du gradient** est une méthode qui permet de calculer le gradient de l'erreur pour chaque neurone du réseau, de la dernière couche vers la première.

- **Algorithmes de minimisation**
  - **Principe des algorithmes**

Soit *J(w)* la fonction de coût. Les algorithmes utilisés nécessitent que *J(w)* soit dérivable par rapport aux poids. Le principe de ces méthodes est de se placer en un point initial, de trouver une direction de descente du coût dans l'espace des paramètres *w*, puis de se déplacer d'un pas dans cette direction. On atteint un nouveau point et l'on itère la procédure jusqu'à satisfaction d'un critère d'arrêt. Ainsi, à l'itération *k*, on calcule :

$$w_k = w_{k-1} + a_{k-1}.d_{k-1}$$

a_k est le pas de la descente et d_k est la direction de descente : les différents algorithmes se distinguent par le choix de ces deux quantités.

- ✓ **Descente du gradient**

Le principe est de partir d'un point aléatoire puis de se déplacer dans la direction de la plus forte pente. En appliquant un certain nombre d'itérations, l'algorithme converge vers une solution qui est un minimum local de f.

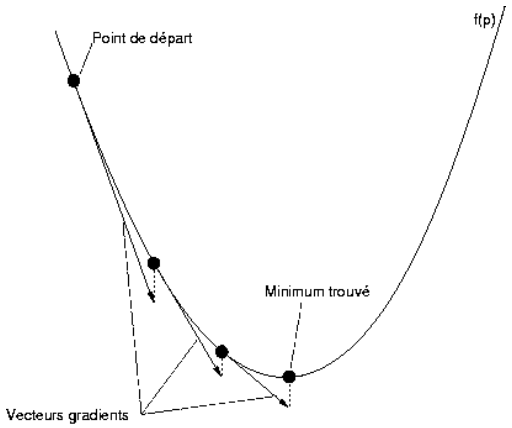


Figure 2.16. Illustration du principe de la descente de gradient, dans le cas d'une fonction à une variable

On commence donc par choisir un vecteur $\mathbf{p}_0$ de manière aléatoire. Puis pour l'itération numéro i on calcule le gradient de f au point $\mathbf{p}_{i-1}$:

$$\mathbf{g}_i = \nabla f(\mathbf{p}_{i-1})$$

Équation 2.5. Calcul du gradient

Le nouveau vecteur de paramètres calculé est :

$$\mathbf{p}_i = \mathbf{p}_{i-1} - \eta \mathbf{g}_i$$

Équation 2.6. Nouveau vecteur paramètre

Où η est une constante qui ajuste la vitesse de convergence de l'algorithme, déterminée empiriquement. Une fois le nouveau vecteur P_i calculé on passe à l'itération suivante. Si η est trop grande, l'algorithme n'est pas stable et oscille autour d'une solution, et si η est trop petite, un très grand nombre d'itérations sera nécessaire pour converger vers la solution, et la probabilité de convergence vers une solution locale est plus grande. Plusieurs critères peuvent être définis pour arrêter l'algorithme : on peut limiter à un certain nombre d'itérations, ou arrêter lorsque $f(P_i)$ atteint un certain seuil minimal ou encore lorsque le vecteur évolue peu, c'est à dire quand la valeur suivante atteint un seuil minimal :

$$\frac{||g_i||}{||p_i||}$$

Équation 2.7. **Critère d'arrêt de la descente de gradient**

Ce dernier critère peut présenter le défaut d'arrêter l'algorithme trop tôt si la fonction présente des plateaux. Le choix du meilleur critère ainsi que le seuil à fixer est généralement trouvé de manière empirique. Il est également possible de prendre une combinaison de ces différents critères.

Le choix du coefficient η peut être délicat dans certains cas. Par exemple si f possède par endroits de grands plateaux, il faudrait avoir un coefficient η grand pour pouvoir s'en affranchir avec peu d'itérations. Si en d'autres endroits f évolue au contraire très rapidement, il faut qu'il soit faible pour que l'algorithme soit stable. Une variante peut être utile dans ce cas, la descente de gradient adaptative.

Dans une descente de gradient adaptative, le coefficient η est également ajusté à chaque itération, suivant l'évolution de la valeur de $f(P_i)$. Si $f(P_i)$ diminue, il est probable que l'on pourrait aller plus vite en augmentant légèrement η, et au contraire si $f(P_i)$ augmente, cela veut dire que le coefficient η est trop grand et qu'il faut le diminuer. Donc on décide d'augmenter η (de 10% par exemple) si $f(P_i)$ diminue, et de le réduire (en le divisant par 2 par exemple) si $f(P_i)$ augmente. Cette approche permet généralement de réduire le nombre d'itérations requis, et s'est révélée efficace avec tous les réseaux de neurones que nous avons testés.

La descente de gradient peut être appliquée de deux manières lorsque l'on évalue la fonction à l'aide d'une base d'exemples. La méthode que nous avons employé, et décrite ci-dessus, est celle du gradient total. Le vecteur g_i est calculé avec tous les exemples de la base

d'apprentissage à chaque itération, et le nouveau vecteur de paramètres est déterminé après avoir parcouru toute la base. Dans une autre méthode, dite du gradient stochastique, le vecteur g_i est calculé avec chaque exemple, et le vecteur de paramètres est recalculé entre chaque exemple. Cette dernière méthode est particulièrement adaptée aux systèmes dits online, pour lesquels les exemples sont communiqués l'un après l'autre pendant l'optimisation, alors que pour la méthode du gradient total il est nécessaire d'avoir la base complète avant de commencer la première itération.

- ✓ **La méthode de Newton**

La méthode de Newton utilise la courbure (dérivée seconde) de la fonction de coût pour atteindre le minimum. La modification des paramètres s'écrit ainsi :

$wk = w_{k-1} - H^{-1}{}_{k-1} . \nabla J(w_{k-1})$

La direction de descente est $H^{-1}{}_{k-1} . \nabla J(w_{k-1})$ où H_{k-1} est l'inverse du hessien de la fonction de coût, la matrice hessienne est la matrice des dérivées secondes de l'indice de performances de poids et de biais.

Dans la pratique, le calcul du hessien et surtout de son inverse est à la fois complexe et source d'instabilités numériques ; on utilise de préférence une méthode de "quasi-Newton".

- ✓ . **La méthode de quasi-Newton**

Les méthodes de quasi-Newton consistent à approcher l'inverse du hessien plutôt que de calculer sa valeur exacte.

La modification des paramètres s'écrit : $wk = w_{k-1} - \alpha_{k-1} M_{k-1} . \nabla J(w_{k-1})$

La suite M_k est construite de façon à converger vers l'inverse du hessien avec M_0 égale à la matrice identité. Cette suite est construite grâce à la méthode dite BFGS (Broyden, Flechter, Goldfarb, Shanno 1970) dont la vitesse de convergence est beaucoup plus grande que celle de la méthode du gradient.

Dans Matlab on retrouve ces algorithmes dans la fonction « **trainbfg** ».

- ✓ **La méthode Levenberg-Marquardt**

Si la descente de gradient est trop lente pour réaliser l'apprentissage, il est également possible d'utiliser un algorithme du second ordre, tel que celui de Levenberg Marquardt. Dans ce cas chaque itération demande plus de calculs, mais dans la plupart des cas le nombre d'itérations nécessaires pour converger est bien moindre.

Une autre manière de diminuer le nombre d'itérations d'un algorithme d'optimisation est d'utiliser les dérivées secondes.

Mais le calcul des dérivées secondes peut être très long, tout d'abord parce que le nombre de dérivées secondes est le carré de celui des dérivées premières, et également parce que la dérivée seconde peut être assez complexe. De nombreux algorithmes, peut-être abusivement appelés algorithmes d'ordre 2, utilisent en fait une approximation des dérivées secondes calculées à partir de dérivées premières. Cependant ils gardent l'avantage de nécessiter beaucoup moins d'itérations qu'une descente de gradient.

L'algorithme de Levenberg Marquardt fait partie de ces algorithmes, et s'applique au cas où la fonction est une erreur quadratique moyenne.

En pratique cet algorithme, en particulier dans le cas des réseaux de neurones, permet de converger avec beaucoup moins d'itérations. Mais chaque itération demande plus de calculs, en particulier pour l'inversion de la matrice $\mathbf{H}$, et son utilisation se limite donc aux cas où le nombre de paramètres à optimiser n'est pas très élevé. En effet le nombre d'opérations nécessaires à l'inversion d'une matrice est proportionnel à N^3, Nétant la taille de la matrice.

Dans Matlab on retrouve ces algorithmes dans la fonction « **trainlm** »

- En général, les réseaux bouclés sont plus difficiles à entraîner que les réseaux non bouclés. L'apprentissage est cependant généralement assez complexe dans ces réseaux, et leurs propriétés sont souvent moins bien connues que celles des réseaux non bouclés. Pour cette raison on va s'intéresse dans ce travail par les réseaux non bouclé c.à.d. **les perceptrons multicouche**

➢ Application de la rétro propagation pour un PMC

L'apprentissage d'un perceptron se fait avec une rétro propagation, Dans le cas d'un perceptron à une couche l'expression de l'évolution des poids est assez simple. En effet l'erreur du réseau est de la forme :

$$e_{i,k} = f_1(\mathbf{w_{1,i}} \cdot \mathbf{x_k}) - y_{i,k}$$

Équation 2.8. Erreur d'un perceptron à une couche

où $i$est le numéro de la sortie et $k$celui de l'exemple de la base d'apprentissage. La performance du réseau, une erreur quadratique moyenne, est :

$$\epsilon = \frac{1}{N_a} \sum_{k=1}^{N_a} \sum_{i=1}^{N_s} e_{i,k}^2$$

Équation 2.9. Performance du perceptron à une couche

En appliquant à ces équations l'algorithme de descente du gradient ((2.5) et (2.6))

$$\delta \mathbf{w_{1,i}} = -2\eta \sum_{k=1}^{N_a} e_{k,i} f_1^{'}(\mathbf{w_{1,i}} \cdot \mathbf{x_k}) \mathbf{x_k}$$

Équation 2.10. Évolution des poids durant une étape de l'apprentissage d'un perceptron à une couche

Dans un perceptron multicouche il faut tenir compte de l'influence de plusieurs couches dans le calcul du gradient. Reprenons les notations utilisées dans l'architecture de PMC, en notant $\mathbf{a_{l,k}}$ la sortie de la couche l lorsque l'on applique en entrée l'exemple $\mathbf{x_k}$. En notant de plus $\mathbf{a_{0,k}} = \mathbf{x_k}$, on peut appliquer (2.4) pour calculer tous les $\mathbf{a_{l,k}}$ avec $l = 1 \ldots N_c$ et $k = 1 \ldots N_a$. La performance du perceptron est :

$$\epsilon = \frac{1}{N_a} \sum_{k=1}^{N_a} ||\mathbf{a_{N_c,k}} - \mathbf{y_k}||^2$$

Équation 2.11. Performance du perceptron multicouche

En appliquant l'algorithme de descente du gradient ((2.5) et (2.6)) à cette performance

$$\delta \mathbf{w_{l,i}} = -2\eta \sum_{k=1}^{N_a} e_{l,k,i} f_l^{'}(\mathbf{w_{l,i}} \cdot \mathbf{a_{l-1,k}}) \mathbf{a_{l-1,k}}$$

Équation 2.12. Évolution des poids durant une étape de l'apprentissage d'un perceptron multicouches

Où le terme d'erreur $\mathbf{e_{l,k}}$ de composantes $e_{l,k,i}$ est de la forme :

$$\begin{cases} \mathbf{e_{N_c,k}} = \mathbf{a_{N_c,k}} - \mathbf{y_k} \\ e_{l,k,i} = \sum_{j=1}^{n_{l+1}} e_{l+1,k,j} w_{l+1,j,i} f_{l+1}^{'}(\mathbf{w_{l+1,j}} \cdot \mathbf{a_{l,k}}) \quad l = 1 \ldots N_c - 1, \quad i = 1 \ldots n_l \end{cases}$$

Équation 2.13. Erreur par couche dans un perceptron multicouche

On constate que l'évolution des poids est similaire à celle vue pour le perceptron à une couche, en définissant une erreur sur chaque couche du perceptron. L'erreur de la dernière couche est effectivement l'erreur du réseau, et pour chaque couche cachée les erreurs sont calculées à partir des erreurs de la couche suivante. Pour chaque neurone l'erreur est la somme des erreurs de chaque neurone de la couche suivante, pondérée par le poids qui le lie au

neurone dont on calcule l'erreur et par la dérivée de la fonction d'activation. Pour cette raison cet algorithme est appelé rétro propagation de l'erreur.

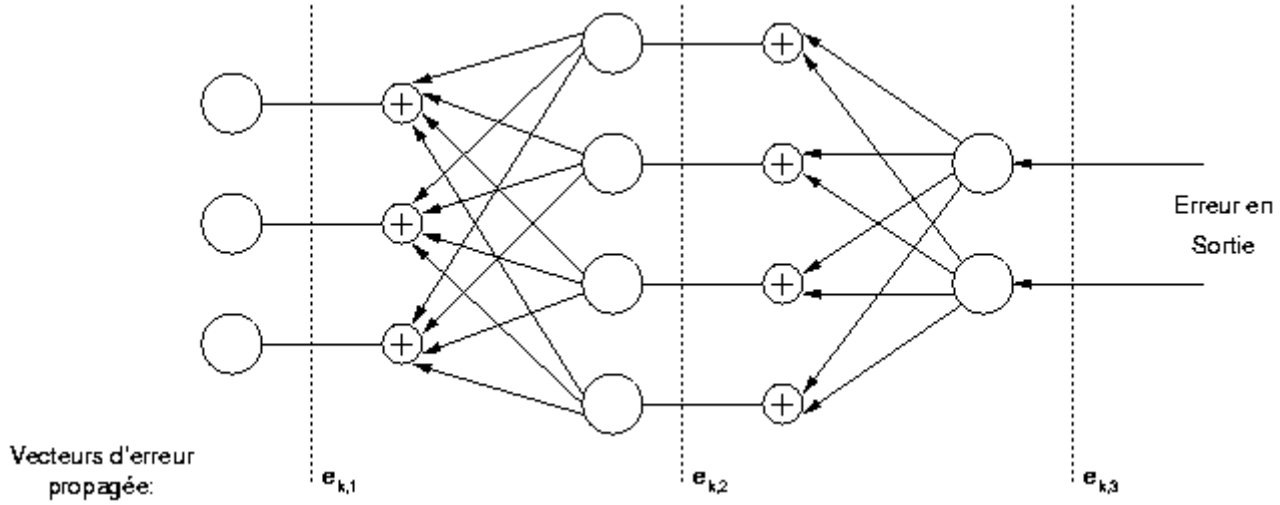


Figure 2.17. Principe de la rétro propagation

L'expression de la descente de gradient sous cette forme permet une réalisation simple de l'algorithme. A chaque itération, les différents vecteurs $a_{k,l}$puis la sortie du perceptron sont calculés en utilisant l'équation **(2.4)** en allant de la couche 1à la couche N_c, puis les erreurs sont calculées en utilisant l'équation **(2.12)** en allant de la couche N_cà la couche 1. Enfin les poids de chaque neurone sont mis à jour en utilisant l'équation **(2.13)**.

Lorsque l'on augmente le nombre de couches, on constate que l'algorithme d'apprentissage nécessite de plus en plus d'itérations pour converger vers un résultat. C'est pour cette raison que l'on dépasse rarement deux couches cachées dans un PMC. De plus une ou deux couches cachées suffisent généralement pour approximer ce que l'on veut. Si la descente de gradient est trop lente pour réaliser l'apprentissage, il est également possible d'utiliser un algorithme du second ordre, tel que celui de Levenberg Marquardt, . Dans ce cas chaque itération demande plus de calculs, mais dans la plupart des cas le nombre d'itérations nécessaires pour converger est bien moindre.

II.3.2 Apprentissage non supervisé

L'apprentissage non supervisé est une technique différente où on ne détermine pas de variables de sortie. Le réseau va de lui même catégoriser les variables d'entrée.

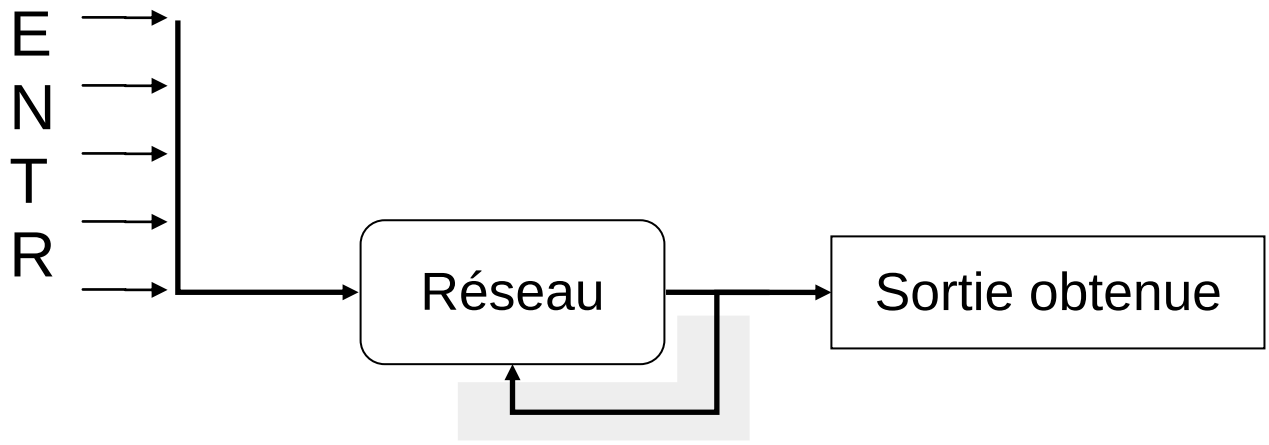


Figure2.18 : Présentation d'un apprentissage non supervisé

L'apprentissage est non supervisé généralement est appliqué pour les réseaux bouclé, basé sur des probabilités. Le réseau va se modifier en fonction des régularités statistiques de l'entrée *u* et établir des catégories, en attribuant et en optimisant une valeur de qualité, aux catégories reconnues. On ne sait pas à priori si la sortie obtenue est valable ou non. Les entrées sont projetées sur l'espace de réseau.

II.3.3 Apprentissage par renforcement

L'apprentissage par renforcement consiste à apprendre quoi faire, comment associer des actions à des situations, afin de maximiser quantitativement une récompense. On ne dit pas à l'apprenant quelle action faire, mais au lieu de cela, il doit découvrir quelles actions donnent le plus de récompenses en les essayant. Dans le cas le plus intéressant, des actions peuvent affecter non seulement les récompenses immédiates mais aussi la situation suivante, et par là, les récompenses à plus long terme .ces deux propriétés « recherche par essai-erreur et récompense à long terme » sont les deux caractéristiques les plus importantes de l'apprentissage par renforcement.

L'apprentissage par renforcement est différent de l'apprentissage supervisé.
Ce dernier nécessite un superviseur qui dicte au réseau quelle action est correcte dans telle situation. Dans l'apprentissage par renforcement, le réseau n'a pas de superviseur à sa disposition, il interagit avec l'environnement qui lui donne un retour quantitatif sur les valeurs des ses actions.

II.3.4 Apprentissage « en ligne » et apprentissage « hors-ligne »

Il existe deux modes principaux d'apprentissage, selon la façon dont les vecteurs de poids synaptiques sont adaptés. Le premier, dit apprentissage « en-ligne », consiste à modifier les valeurs des poids synaptiques immédiatement après la présentation d'un objet ou exemple.

Dans ce cas, seul le gradient instantané de la fonction de coût est utilisé pour l'adaptation des paramètres du système.

C'est généralement le cas pour l'apprentissage par renforcement qui nécessite une réponse de l'environnement.

Dans Le second mode principal d'apprentissage : « hors-ligne » ou « off-line », Le réseau est entraîné à partir d'une base d'apprentissage (généralement une base d'exemples).

Ce type d'apprentissage consiste à accumuler les gradients instantanés consécutifs, et à n'effectuer l'adaptation des poids synaptiques que lorsque l'ensemble des objets d'apprentissage ont été présentés au réseau de neurones. On parle alors d'apprentissage « hors-ligne ». Cette dernière méthode permet de mieux estimer le gradient de la fonction de coût, puisqu'il est à présent calculé à partir d'un ensemble d'objets, plutôt qu'à partir d'un seul.

Dans l'apprentissage enligne chacune des formes est représenté une seule fois ce qui peut sembler donner un apprentissage plus rapide mais nécessite plus de données d'apprentissage **.**

II.3.5 Problèmes d'apprentissage

L'apprentissage d'un réseau de neurones multicouches est un problème d'optimisation puisque cela consiste à minimiser une fonction coût. On y retrouve donc toutes les difficultés liées à ces problèmes, minima locaux, choix d'architecture…. Nous allons introduire dans ce qui suit plusieurs techniques permettant de pallier ces difficultés.

II.3.5.1 -minima locaux

Dans le cas où la fonction à minimiser n'admet qu'un seul minimum, ce dernier sera atteint très rapidement quel que soit l'algorithme de minimisation utilisé. Ces situations sont malheureusement très rares, et correspondent souvent à des problèmes très simples pour lesquelles l'utilisation des techniques neuronales peut paraître exagérée. On rencontrera plus souvent dans la nature, des fonctions coût dont la forme est plus proche de celle présentée dans la Figure suivante et comportant de multiples minima locaux.

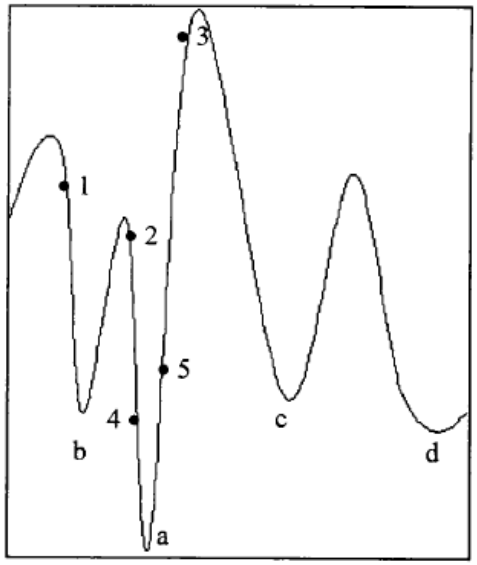


Figure2.19 : Forme générale d'une fonction coût possédant plusieurs minima

L'algorithme de descente du gradient détermine une direction suivant laquelle la fonction coût sera décroissante. Supposons que nous soyons au point 1 de la Figure 2 .9, l'algorithme va nous faire descendre vers le point b qui est un minimum local. Une fois au point b, il ne sera plus possible d'atteindre le minimum global (point a), puisque cela ferait augmenter la valeur de la fonction coût. Une manière de s'en sortir, consiste à partir d'un point initial compris entre les points 2 et 3. Dans ce cas, toute descente de la fonction coût convergera vers le minimum global a

Pour un réseau multicouche, on peut observer le même phénomène. Lorsqu'on initialise aléatoirement les paramètres à optimiser, ici les poids synaptiques, on démarre l'apprentissage en un point quelconque de la fonction coût et on ne peut jamais être sûr d'atteindre le minimum global. Aussi, une solution consiste à réaliser plusieurs apprentissages à partir de différentes initialisations des poids. En effet on augmente ainsi les chances de débuter un apprentissage dans une zone favorable. Il faut donc systématiquement réaliser plusieurs apprentissages à partir de configurations initiales différentes, et choisir celui qui converge vers l'erreur la plus faible. Ce procédé assure de trouver le minimum global sur un nombre infini de configurations. Dans la pratique on réalise une dizaine d'essais permettant d'obtenir un minimum local acceptable, à défaut d'avoir le meilleur

Plusieurs recherches travaillent sur cette problématique …. Il ya des travaux sur l'utilisation des algorithmes de minimisation plus complexes, permettant d'obtenir une

meilleure convergence vers le minimum global, comme le recuit simulé. algorithme constructif **[Juan-Manuel TORRES-MORENO1997],[D. Martinez & D. Estève.1992],[M. Karouia, R. Lengellé, & Denoeux T.1995], [Y. Shang&B.W. Wha.1996]** ,**],**algorithme de décalage **[Y. LeCun, J. Denker1991],** algorithme Rprop **[C. Igel and M. H□usken.2003],** algorithme évolutionniste**[d.meurier2007].**

D'autres travaux ils ont utilisé des techniques de prétraitement des données EX. Réduction ave ACP **[Hedilli2004],** LPC **[Belgacem2002],** utilisation des ondelettes **[Y.OUSSAR 2002]**

II.3.5.2 choix d'architecture

On se pose beaucoup de questions sur la manière d'organiser un réseau multicouches. Doit-on utiliser une ou deux couches cachées ? Combien faut-il de neurones en couche cachée ? Quel doit être la taille de la base d'apprentissage ?

A toutes ces questions, il n'existe pas de réponse absolue, ni de formules de calcul toutes prêtes. On dispose par contre de plusieurs règles pseudo-empiriques permettant d'orienter l'utilisateur sur ces choix.

Ces règles n'étant pas à toute épreuve, il convient d'essayer plusieurs configurations et de choisir celle qui donne les meilleurs résultats en généralisation.

On peut toutefois être tenté d'utiliser plusieurs couches cachées afin de réduire le nombre de neurones par couche, mais cela augmente fortement le nombre de minima locaux et le temps d'apprentissage. Aussi, il est préférable tant que cela est possible, d'utiliser une seule couche cachée.

Il reste encore à choisir le nombre de neurones de la couche cachée. Pour cela on utilise quelques règles pseudo-empiriques permettant d'avoir une bonne estimation. La première méthode, appelée la règle de la pyramide **[Masters, 1993],** repose sur la conjecture que le nombre de neurones de chaque couche suit une progression géométrique de la couche de sortie vers la couche d'entrée. Pour cela, supposons que l'on ait m neurones dans la couche de sortie n dans la couche d'entrée et k couches cachées, on définit la raison r par :

$$r = \sqrt[k+1]{\frac{n}{m}}$$

On numérote ensuite les couches du réseau, la sortie portant l'indice 0 et l'entrée l'indice (k+l). Le nombre **Ni** de neurones de la couche i est alors égal à :

$$N_i = m \cdot r^i$$

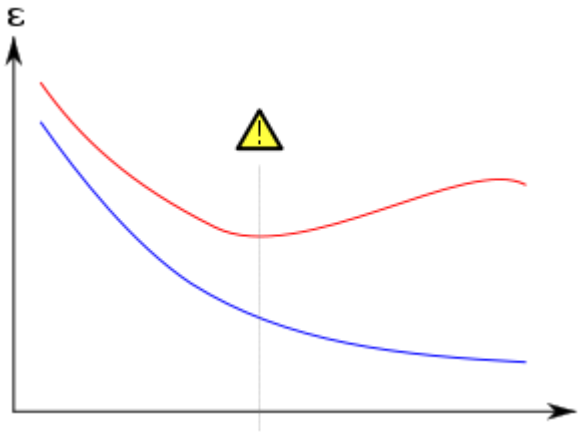
Pour un réseau à une couche cachée, le nombre Ni de neurones sera :

$$N = m \cdot r^1 = m \cdot \sqrt{\frac{n}{m}} = \sqrt{m \cdot n}$$

Il y a plusieurs chercheurs qui travaillent sur la problématique de choix d'architecture, surtout sur le nombre de neurone dans la couche cachée et les liaison entre les différent couche .la plupart des recherches travail sur un apprentissage structurelle des réseaux de neurone on utilisant différent algorithme ex. Algorithme constructif **[N. Dunkin, J. Shawe-Taylor, and P. Koiran.1997],** algorithme de construction en cascade **[S. E. Fahlman and C. Lebiere.1990],** algorithme de construction incremental **[N. Dunkin, J. Shawe-Taylor, and P. Koiran.1997].**

II.3.5.3 Sur apprentissage :

Sur apprentissage (en anglais « overfitting ») est un problème pouvant survenir dans l'apprentissage des réseaux de neurones. Il est en général provoqué par un mauvais dimensionnement de la structure utilisée pour classifier. De part sa trop grande capacité à stocker des informations, une structure dans une situation de sur apprentissage aura de la peine à généraliser les caractéristiques des données. Elle se comporte alors comme une table contenant tous les échantillons utilisés lors de l'apprentissage et perd ses pouvoirs de prédiction sur de nouveaux échantillons.



Surapprentissage dans un apprentissage supervisé. En rouge, l'erreur sur l'ensemble de validation. En bleu, l'erreur d'apprentissage. Si l'erreur de validation augmente alors que l'erreur d'apprentissage continue à diminuer alors il y a un risque de surapprentissage.

Figure2.20 : sur apprentissage

- ***Éviter le sur apprentissage***

Pour limiter ce genre de problèmes dans le cas des réseaux de neurones, on doit veiller à utiliser un nombre adéquat de neurones et de couches cachées. Cependant, ces paramètres sont difficiles à déterminer à l'avance. Pour détecter un surapprentissage, on sépare les données en deux sous-ensembles : l'ensemble d'apprentissage et l'ensemble de validation. L'ensemble d'apprentissage comme son nom l'indique permet de faire évoluer les poids du réseau de neurones avec par exemple une rétro propagation. L'ensemble de validation n'est pas utilisé pour l'apprentissage mais permet de vérifier la pertinence du réseau avec des échantillons qu'il ne connait pas.

On peut vraisemblablement parler de surapprentissage si l'erreur de prédiction du réseau sur l'ensemble d'apprentissage diminue alors que l'erreur sur la validation augmente de manière significative. Cela signifie que le réseau continue à améliorer ses performances sur les échantillons d'apprentissage mais perd son pouvoir de prédiction sur ceux provenant de la validation.

Pour avoir un réseau qui généralise bien, on arrête l'apprentissage dès que l'on observe cette divergence entre les deux courbes. On peut aussi diminuer la taille du réseau et recommencer l'apprentissage.

II.4 Les étapes de conception d'un réseau

Le novice est souvent surpris d'apprendre que pour construire un réseau de neurones, la première chose à faire n'est pas de choisir le type de réseau mais de bien choisir ses échantillons de données d'apprentissage, de tests et validation. Ce n'est qu'ensuite que le choix du type de réseau interviendra. Afin de clarifier un peu les idées, voici chronologiquement les quatre grandes étapes qui doivent guider la création d'un réseau de neurones.

II.4.1 Chois des échantillons

Le processus d'élaboration d'un réseau de neurones commence toujours par le choix et la préparation des échantillons de données. Comme dans les cas d'analyse de données, cette étape est cruciale et va aider le concepteur à déterminer le type de réseau le plus approprié pour résoudre sont problème. La façon dont se présente l'échantillon conditionne : le type de réseau, le nombre de cellules d'entrée, le nombre de cellules de sortie et la façon dont il faudra mener l'apprentissage, les tests et la validation.

II.4.2 Élaboration de la structure du réseau

La structure du réseau dépend étroitement du type des échantillons. Il faut d'abord choisir le type de réseau : un perceptron standard, un réseau de Hopfield, un réseau à décalage temporel (TDNN), un réseau de Kohonen, un ARTMAP etc... Dans le cas du perceptron par exemple, il faudra aussi choisir le nombre de neurones dans la couche cachée. Plusieurs méthodes existent et on peut par exemple prendre une moyenne du nombre de neurones d'entrée et de sortie, mais rien de vaut de tester toutes les possibilités et de choisir celle qui offre les meilleurs résultats.

II.4.3 Apprentissage

L'apprentissage consiste tout d'abord à calculer les pondérations optimales des différentes liaisons, en utilisant un échantillon. La méthode la plus utilisée est la *rétro propagation* : on entre des valeurs du les cellules d'entrée et en fonction de l'erreur obtenue en sortie (le *delta*), on corrige les poids accordés aux pondérations. C'est un cycle qui est répété jusqu'à ce que la courbe d'erreurs du réseau ne soit croissante (il faut bien prendre garde ne pas sur-entrainer un réseau de neurones qui deviendra alors moins performant). Il existe d'autres méthodes d'apprentissage telles que le *quickprop* par exemple. Mais la plus utilisée reste encore la rétropropagation.

II.4.4 Validation et test

Une fois le réseau calculé, il faut toujours procéder à des tests afin de vérifier que notre réseau réagit correctement. Il y a plusieurs méthodes pour effectuer une validation : la cross validation, le bootstrapping... mais pour les tests, dans le cas général, une partie de l'échantillon est simplement écarté de l'échantillon d'apprentissage et conservé pour les tests hors échantillon.

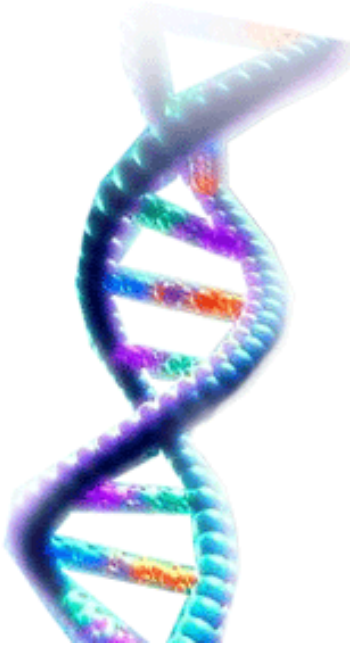
II.5 conclusion

L'approche neuronale est une technique utilisé dans les systèmes d'aide aux diagnostique médical, qui prend une grande place dans le recherche biomédicale.

Cette approche rencontre deux grands problèmes dans leur apprentissage ces problèmes sont les « minima locaux » et « choix d'architecture

.dans ce travail on s'intéresse à ces deux problèmes. En hybridant cette approche avec une technique intelligente issue de l'intelligence artificielle, une technique d'optimisation globale présentée dans le chapitre3.

CHAPITRE 3



les algorithmes génétiques

III.1 Introduction

La résolution du problème de minimisation de la fonction de cout dans les réseaux de neurones, n'est pas aisée. Nous avons vu précédemment que les algorithmes utilisés sont plus ou moins robuste face `à la convergence vers une solution globale.

Dans la famille des algorithmes stochastiques, beaucoup plus robustes que les algorithmes déterministes, les algorithmes génétiques sont de plus en plus utilises. Ils sont bases sur un phénomène naturel qui a fait ses preuves : ***l'evolution***. Plus précisément, ils s'inspirent de l'évolution d'une population d'individus dans un milieu donne.

Les AG tirent leur nom de l'évolution biologique des êtres vivants dans le monde réel. Ces algorithmes cherchent à simuler le processus de la sélection naturelle dans un environnement défavorable en s'inspirant de la théorie de l'évolution proposée par C. Darwin. Dans un environnement, « *les individus* » les mieux adaptés tendent à vivre assez longtemps pour se reproduire alors que les plus faibles ont tendance à disparaître

Par analogie avec l'évolution naturelle, les AG font évoluer un ensemble de solutions candidates, appelé une « population d'individus ». Un « individu » n'est autre qu'une solution possible du problème à résoudre. Chaque individu de cette population se voit attribuer une fonction appelée fonction d'adaptation (*fitness*) qui permet de mesurer sa qualité ou son poids; cette fonction d'adaptation peut représenter la fonction objectif à optimiser. Ensuite, les meilleurs individus de cette population sont sélectionnés, subissent des croisements et des mutations et une nouvelle population de solutions est produite pour la génération suivante.

Ce processus se poursuit, génération après génération, jusqu'à ce que le critère d'arrêt soit atteint, comme par exemple le nombre maximal de générations.

Ce chapitre est organisé en trois sections. Dans la section 1, le principe du fonctionnement d'un AG, ainsi que les principaux éléments qui caractérisent un AG standard, dans section2 on va présenter les systèmes hybrides et comment hybridé les AG avec les réseaux de neurones.

III.2 Les algorithmes génétiques

III.2.1 De la génétique à l'algorithmique

Les théories de l'évolution montrent que les êtres vivants évoluent sous l'effet du milieu, les milieux adaptés ont plus de chance de survivre donc de se reproduire. De génération en génération, les caractéristiques des individus les mieux adaptés on par conséquent plus de chance de se reproduire dans la population.la génétique, dont l'objet est d'étudier les mécanismes de l'hérédité, propose quant à elle un modèle permettant d'expliquer la transmission de ces caractéristiques d'une génération à l'autre.

Il existe en effet des entités responsables de la production des caractères héréditaires : ces entités sont appelées les gènes et l'ensemble des gènes d'un individu définit son génotype. Par opposition le phénotype d'un individu correspond à son apparence physique, ou plus précisément à un ensemble de caractéristiques que l'on peut observer, mesurer ou qualifier chez lui (certaines pouvant nécessiter des investigations complexes, comme la mesure de taux de globules rouges dans un système sanguin ; le phénotype est susceptible de varie au cours du temps par l'effet du milieu dans lequel il évolue , mais ces variations ne peut être transmises. Un gène est en effet un segment de chromosomes, long filament d'ADN (acide désoxyribonucléique), qui est le matériel génétique de toutes les cellules. Les chromosomes sont situés dans les noyaux des cellules. Toutes les cellules sauf les cellules sexuelles, possède le même nombre de chromosome, présents sous forme de paires, dans chaque paire, un chromosome vient du paire et l'autre de la mère ; les chromosomes d'une même paire appelées homologues. Dans les cellules sexuelles on observe que la moitie des chromosomes, , chaque paire ne fournissant qu'un seul de ces chromosomes homologues.

Deux mécanismes permettent de fabriquer une nouvelle cellule. Le premier procède par division cellulaire : une cellule duplique sans matériel génétique avant de se couper en deux copies conformes. Ou presque conformes : une erreur de reproduction peut se produire, affectant un gène ; il ya alors mutation de ce gène. Ce mécanisme est mise en ouvre dans la reproduction asexué. Le seconde fait intervenir deux parents pour fabriquer un enfant : c'est celui qui intervient dans la reproduction sexuée, qu'il convient plutôt d'appeler procréation, puisque l'individu crée n'est pas une reproduction de ces parents. Dans ce mécanisme, les ce mécanisme les cellules

sexuelles transmises par les parents apportent chacun la moitié des chromosomes de l'enfant ce qui lui redonne bien le nombre voulu de chromosomes.

A défaut de pouvoir de créer de nouveau gènes, la procréation favorise un brassage génétique, puisque le patrimoine génétique de l'enfant est issu à l'égalité des deux parents (alors que la reproduction conforme n'évolue que du fait des mutations qui se produise aléatoirement et rarement, du moins dans des conditions normales).deux opérations viennent s'ajouté à ce brassage de la procréation pour accroitre encore la diversité qui en résulte : les mutations, déjà évoqué, et le crossing-over(ou enjambement) .le crossing-over a été induit pour expliquer comment des gènes situés sur un même chromosome (gène liés), peuvent ne pas subir le même sort : l'un est transmis et l'autre non. Il arrive en effet, lors de la production des chromosomes qu'un parent va transmettre à l'enfant par l'intermédiaire des cellules sexuelle qu'il fabrique, qu'un échange s'opère, chez ces parents, entre chromosomes homologues : une séquence de l'un se substitue a une séquence de l'autre : cet échange s'appelle crossing-over. Le chromosome qui reçoit l'enfant est alors constitué de morceaux des deux chromosomes homologues détenus par la cellule parentale et provenant donc des grands-parents de cet enfant. On remarquera que deux gènes proches sur un chromosome ont une probabilité plus faible d'être séparés par un crossing- over que deux gènes éloignés **[saaidi.L2007]**

Les algorithmes génétiques vont s'inspirer, de façon plus ou moins fidèle, de ces mécanismes, sans pour autant en épuiser la complexité. Leurs prémices remontent au début des années 60, John Holland étudie les systèmes évolutifs et, en 1975, il introduit le premier modèle formel des algorithmes génétiques (*the canonical genetic algorithm AGC*) dans son livre *Adaptation in Natural and Artificial Systems*. Il expliqua comment ajouter de l'intelligence dans un programme informatique avec les croisements (échangeant le matériel génétique) et la mutation (source de la diversité génétique). Ce modèle servira de base aux recherches ultérieures et sera plus particulièrement repris par Goldberg qui publiera en 1989, un ouvrage de vulgarisation des algorithmes génétiques, et ajouta à la théorie des algorithmes génétiques les idées suivantes :

- un individu est lié à un environnement par son code d'ADN.
- Une solution est liée à un problème par son indice de qualité.

Le domaine de l'évolution artificielle n'a connu une réelle expansion qu'à partir de ces 15 dernières années. Pourtant, l'idée de simuler sur ordinateurs des

phénomènes évolutionnaires remonte aux années 50. Des concepts tels que la représentation des chromosomes par des chaînes binaires étaient déjà présents.

L'essor de l'évolution artificielle, depuis les années 80, peut s'expliquer par deux phénomènes concurrents. Premièrement, cet essor est principalement dû à l'accroissement exponentiel des moyens de calculs mis à la disposition des chercheurs, ce qui leur permet d'afficher des résultats expérimentaux pertinents et prometteurs. Le deuxième point est l'abandon du biologiquement plausible.

Nature	**Algorithme génétique**
individu	Solution à un problème
population	Ensemble de solution
But : survive	But : optimiser une fonction
Adéquation au milieu	Qualité de la solution (**fitness**)
Brin ADN	Représentation codée d'une solution
mutation	Modification d'une solution
Croisement	Décomposition et recombinaison
Sélection naturelle	Roulette biaisée

Tableau3.1 : **de la génétique à l'algorithmique**

III.2.2 définition

Les algorithmes génétiques sont des algorithmes d'exploration fondés sur les mécanismes de la sélection naturelle et de la génétique. Ils sont basés sur les principes de survie de structures les mieux adaptées et les échanges d'information.

À chaque génération, un nouvel ensemble de créatures artificielles (codées sous forme de chaînes de caractères) est construit à partir des meilleurs éléments de la génération précédente. Bien que reposant fortement sur le hasard (et donc sur un générateur de nombres aléatoires) ces algorithmes ne sont pas purement aléatoires. **[C. Bontemps1995]**

III.2.3 Les éléments d'un algorithme génétique

Les algorithmes génétiques sont inspirés de la génétique classique: on considère une « population » le point répartis dans l'espace. Avant d'expliquer en détail le fonctionnement d'un algorithme génétique, nous allons présentés quelques mots de vocabulaire relatifs à la génétique. Ces mots sont souvent utilisés pour décrire un algorithme génétique

- *Individu/chromosome/séquence* : une solution potentielle du problème ;
- *Population* : un ensemble de chromosomes ou de points de l'espace de recherche ;
- *Environnement* : l'espace de recherche ;
- *Fonction de fitness* : la fonction - positive - que nous cherchons à optimiser.

Avant d'aller plus loin il nous faut définir quelques termes importants généralement définis sous l'hypothèse de codage binaire.

- **Définition 1 (Séquence/Chromosome/Individu).** .

Nous appelons une séquence (chromosome, individu) A de longueur l(A) une suite $A = \{a_1, a_2, ..., a_l\}$ avec $\forall i \in [1, l], a_i \in V = \{0, 1\}.$ dans le codage binaire et $a \in \Re.$ Dans le codage réel

- **Définition 2 (Fitness d'une séquence).** Nous appelons fitness d'une séquence toute valeur positive que nous noterons f (A), où f est typiquement appelée fonction de fitness.

La fitness (efficacité) est donc donnée par une fonction à valeurs positives réelles. Dans le cas d'un codage binaire, nous utiliserons souvent une fonction de décodage *d* qui permettra de passer d'une chaîne binaire à un chiffre à valeur réelle. La fonction de fitness est alors choisie telle qu'elle transforme cette valeur en valeur positive

Le but d'un algorithme génétique est alors simplement de trouver la chaîne qui optimise cette fonction *f*. Bien évidemment, chaque problème particulier nécessitera ses propres fonctions *d* et *f*.**[T.Vallée&M.Yudizoglu2001]**

III.2.4 **fonctionnement des AGs [N. Benhamed2002]**

Les AG fonctionnent avec une population regroupant un ensemble d'individus (*chromosomes)*. Pour chaque individu on attribue une valeur calculée par la fonction d'adaptation ou *fitness*

. En pratique, à partir d'une population, des chromosomes sont générés d'une façon aléatoire lors de l'initialisation. Pour définir la taille de la population. Ont mentionné que cette taille varie d'un problème à un autre.

Dans chaque cycle d'opérations génétiques, une nouvelle population appelée *génération* est créée à partir des chromosomes de la population courante. Pour cela certains chromosomes appelés '*parents*' sont sélectionnés afin d'élaborer les opérations génétiques. Les gènes de ces parents sont mixés et recombinés pour la production d'autres chromosomes appelés '*enfants*' constituant la nouvelle génération.

Les étapes de l'AG sont répétées durant *t* cycles, l'arrêt de l'algorithme est fixé d'après un *critère d'arrêt*. On peut avoir plusieurs critères d'arrêt :

- Le nombre de génération fixé initialement a été atteint.
- La valeur de la fonction d'adaptation a atteint une valeur fixée *a priori*.
- L'absence d'évolution de la valeur de la fonction d'adaptation des individus d'une population à une autre.
- Les chromosomes ont atteint un certain degré d'homogénéité.

La figure 2.1 illustre les différentes étapes d'un AG:

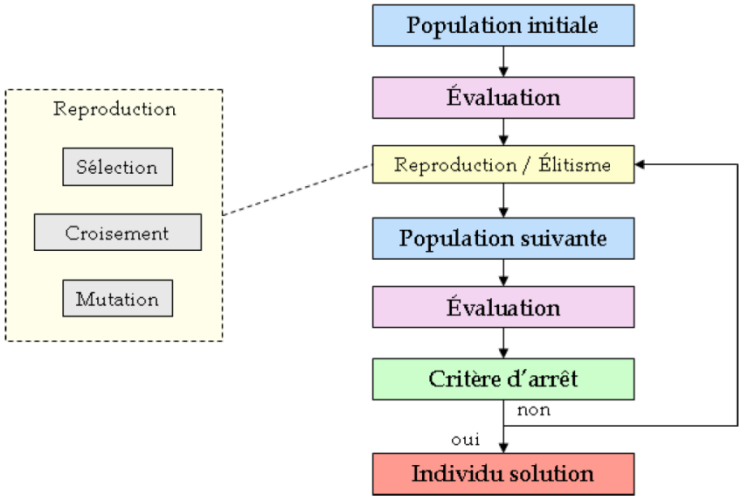


Figure3.1 : schéma de principe d'un algorithme génétique

Globalement l'algorithme est basé sur :

- Une représentation chromosomique des solutions du problème.
- Une méthode pour créer une population initiale de solutions.
- Une fonction d'évaluation (fitness) pour classer les solutions en fonction de leurs aptitudes.
- Des opérateurs génétiques qui définissent la manière dont les caractéristiques génétiques des parents sont transmis aux descendants (enfants).
- Les valeurs des paramètres utilisés par l'AG.

III.2.5 Codage et opérateurs d'un algorithme génétique

Trois opérateurs caractérisent les algorithmes génétiques et rappellent l'origine de ces méthodes, ils vont permettre à la population d'évoluer, par la création d'individus nouveaux construits à l'aide des individus anciens. Plus précisément, on prélève, dans certains individus de la population courante, une partie de leurs caractéristiques en choisissant certaines parties des chromosomes qui les représentent ; puis on recombine ces différentes parties pour constituer les individus de la nouvelle population. La phase de sélection indique dans quelles configurations de la population courante on va prélever des morceaux de chromosomes ; la phase de croisement prélève ces morceaux de chromosomes et les recombine pour former les configurations de la population suivante ; la phase de mutation s'applique à la nouvelle population en changeant éventuellement certains gènes de certains chromosomes obtenus à la fin de la phase de croisement. Une succession des trois

opérations de sélection, de croisement et de mutation constitue une génération, et les algorithmes génétiques consistent donc à faire évoluer une population initiale pendant un certain nombre de générations, nombre déterminé par l'utilisateur. Par ailleurs, comme au cours de cette évolution le meilleur individu calculé depuis le début risque de disparaître, il faut le conserver en mémoire afin de pouvoir le restituer à l'utilisateur à la fin du processus. Dans la suite, on remarquera que, sauf dans la phase de sélection, il n'est pas besoin de connaître la fonction à optimiser pour mettre en œuvre un algorithme génétique. **[I. Charon, A. Germa, O. Hudry,1996]**

III.2.5.1 Le codage

Le ***codage*** est une partie très importante des algorithmes génétiques. Il permet de représenter l'individu sous la forme d'un chromosome. Ce chromosome est constitué de gènes qui prennent des valeurs dans un alphabet binaire ou non.

Le choix du codage est délicat. Il doit permettre de coder toutes les solutions et permettre la mise en œuvre des operateurs de reproduction. C'est ainsi que le bon déroulement des algorithmes génétiques sera assuré.

Plusieurs codages sont employés. Voici quelques exemples.

- **Le codage binaire**

Le gène est code par un caractère binaire, 0 ou 1. C'est le plus courant et celui qui a ´été employé lors de la première application des algorithmes génétiques.

Exemples :

Chromosome A

0	1	0	0	1	1	0	1

Chromosome B

1	1	1	0	0	1	0	1

- **Le codage par permutations de valeurs entières**

Le gène est codé par une valeur entière dans un ensemble de cardinalité ´égale au nombre de gènes. Ce codage est bien adapte aux problèmes d'ordonnancement.

Exemples :

Chromosome A

1	3	6	2	5	8	4	7

Chromosome B

6	1	3	2	7	4	5	8

- **Le codage par valeurs**

Le gène est code par une valeur prise dans un ensemble fini ou infini. Ce codage est généralement utilise pour des valeurs qu'on ne peut pas mettre sous la forme d'un des deux codages précédents. Ces valeurs sont bien entendu liées au problème à résoudre.

Exemples :

Chromosome A

A	A	C	G	T	T	C	G

Chromosome B

2,34	5,67	0,50	0,34	1,87	5,78	3,67	9,12

Chromosome C

BLUE	RED	GREEN	BLUE	WHITE	RED	RED	BLACK

III.2.5.2 Initialisation de la population [N. Benhamed2002]

La population initiale est constituée d'un ensemble d'individus (chromosomes) générés aléatoirement. Cependant rien n'empêche d'utiliser des résultats et des solutions existantes pour former la population initiale. En d'autres termes, un ensemble de solutions connues préalablement peut être injecté dans la population initiale.

Le nombre d'individus d'une population ou la taille de la population constitue un paramètre important pour l'AG qu'il faudra déterminer. La représentation de la population *P* est :

P = (C1, C2, ..., Ci, ..., Ctaille_pop)

Où *Ci* représente la *i*ème chromosome dans la population et *taille_pop* représente le nombre de chromosomes dans la population.

III.2.5.3 Principes de sélection

A l'inverse d'autres techniques d'optimisation, les algorithmes génétiques ne requièrent pas d'hypothèse particulière sur la régularité de la fonction objective. L'algorithme génétique n'utilise notamment pas ses dérivées successives, ce qui rend très vaste son domaine d'application. Aucune hypothèse sur la continuité n'est non plus requise. Néanmoins, dans la pratique, les algorithmes génétiques sont sensibles à la régularité des fonctions qu'ils optimisent.

Le peu d'hypothèses requises permet de traiter des problèmes très complexes. La fonction à optimiser peut ainsi être le résultat d'une simulation.

La sélection permet d'identifier statistiquement les meilleurs individus d'une population et d'éliminer les mauvais. On trouve dans la littérature un nombre important de principes de sélection plus ou moins adaptés aux problèmes qu'ils traitent.

Plusieurs approches sont possibles :

a. La roulette

Cette méthode exploite la métaphore d'une roulette de casino. La roue est divisée en autant de secteurs que d'individus dans la population. La taille de ces secteurs est proportionnelle à l'adaptation de chaque individu. Ces secteurs sont ensuite concaténés sur un axe que l'on normalise entre 0 et 1. On tire alors un nombre aléatoire de distribution uniforme entre 0 et 1, puis on "regarde" quel est le secteur sélectionné. Avec ce système, les grands segments, c'est-à-dire les bons individus, seront plus souvent choisis que les petits.

Lorsque la dimension de la population est réduite, il est difficile d'obtenir en pratique l'espérance mathématique de sélection en raison du peu de tirages effectués.

Un biais de sélection plus ou moins fort existe suivant la dimension de la population.

Individus	*fitness*	*fitness relative*
1	15	0.1
2	30	0.2
3	45	0.3
4	22.5	0.15
5	7.5	0.05
6	30	0.2

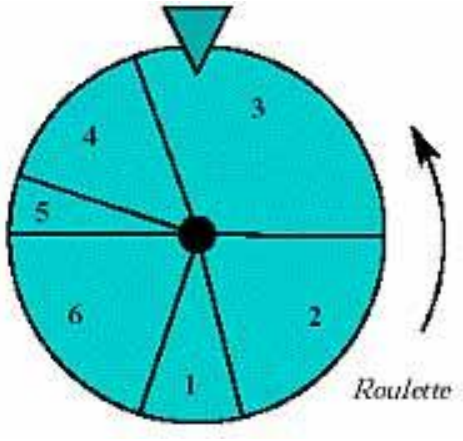


Figure3.2 : exemple d'application de la roulette

b. Stochastique remainder

Elle évite ce genre de problème, car une partie de la population et sélectionné d'une manière purement déterministe. On associe a chaque individu le rapport ri de sa fitness sur la moyenne des fitness puis on dépond de sa partie entière E (ri) qui indique le nombre de fois à reproduire l'individu i. on assure ainsi un nombre exact de représentants pour la génération suivantes, ce qui élimine le bais. Cependant, les individus faibles (fitness inferieure à la fitness moyenne) sont invariablement éliminés avec cette méthode, ce qui est mauvais, ceux-ci occupent des positions dans l'espace d'état qui associées avec d'autre peuvent nous rapprocher du sous-domaine contenant l'optimum on associe donc à la sélection déterministe un principe de sélection aléatoire basée sur une roulette Whell selection exécutée sur tout les individus affectées de nouvelle fitness ri- E(ri)

Individus	*fitness*	r_i	$E[r_i]$	$r_i - E(r_i)$	p_i
1	15	0.6	0	0.6	1/5
2	30	1.2	1	0.2	1/15
3	45	1.8	1	0.8	4/15
4	22.5	0.9	0	0.9	3/10
5	7.5	0.3	0	0.3	1/10
6	30	1.2	1	0.2	1/15

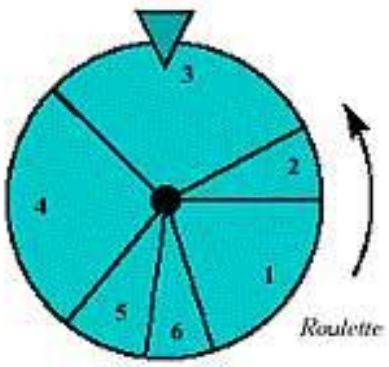


Figure3.3 : exemple d'application de la sélection *remainder*

c. Tournament selection (sélection par tournoi) [A.Berro2001]

A chaque fois qu'il faut sélectionner un individu, la « sélection par tournoi » consiste à tirer aléatoirement (*k*) individus de la population, sans tenir compte de la valeur de leur fonction d'adaptation, et de choisir le meilleur individu parmi les *k* individus. Le nombre d'individus sélectionnés a une influence sur la pression de sélection, lorsque *k* = 2, la sélection est dite par « tournoi binaire ».

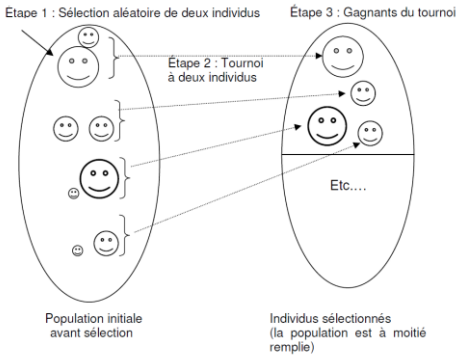


Figure3.4 : Représentation d'une sélection par tournoi d'individus pour un critère de maximisation. Chaque individu représente une solution possible

d. Uniform sélection (sélection uniforme)

C'est une technique très simple qui consiste à sélectionner un individu *Ci* aléatoirement de la population *P*. La probabilité *pi* pour qu'un individu soit sélectionné est définie par :

$$p_i = \frac{1}{taille\_pop}$$

e. stochastique uniforme

Dans cette méthode on dispose une ligne dont lequel chaque parents correspond à une longueur proportionnel à sa valeur. L'algorithme se déplace la long de la ligne dans des pas de taille égaux a chaque pas l'algorithme prend un parent.

III.2.5.4 Opérateur de croisement [N. Benhamed2002]

Le croisement a pour but d'enrichir la diversité de la population en manipulant la structure des chromosomes. Classiquement, les croisements sont envisagés avec deux parents et génèrent deux enfants.ils consiste à échanger les gènes des parents afin de donner des enfants qui porte des propriétés combinées. Bien qu'il soit aléatoire, cet échange d'informatique offre aux algorithmes génétiques une part de leurs puissances : quelque fois de bons gène d'un parent viennent les mauvais gènes d'un autre et créent des fils mieux adaptés aux parents

Il existe différentes techniques de croisement. Chacune des techniques s'applique sur des chromosomes dont la représentation est soit binaire ou réelle. Nous citerons quelques techniques. Une notation des chromosomes qui est adoptée dans les prochaines sections est la suivante :

Le chromosome parent 1 $C^1 = (c^1_1 , c^1_2 , \dots , c^1_l , \dots , c^1_L)$

Le chromosome parent 2 $C^2 = (c^2_1 , c^2_2 , \dots , c^2_l , \dots , c^2_L).$

où $c^k_l \in [a_l , b_l]$ avec k=1,2

a. Croisement à 1 point (croisement simple) [N. Benhamed2002]

On choisit aléatoirement un point de croisement pour chaque couple d'individus sélectionné. Notons que le croisement s'effectue directement au niveau des gènes représentés soit en binaires ou en réels. Un chromosome ne peut pas être coupé au milieu d'un gène. La figure 2.5 illustre ce croisement d'un seul point de coupure dans le cadre d'une représentation binaire ou réelle des gènes des chromosomes.

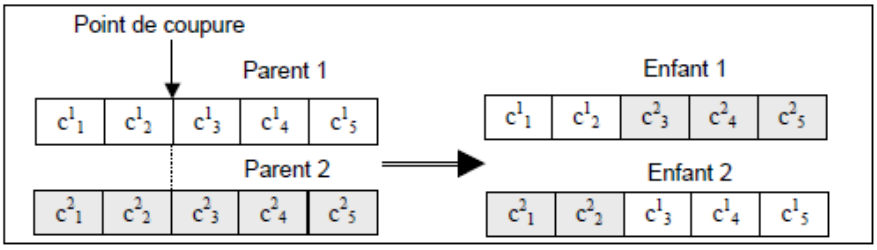


Figure 3.5 : croisement à 1 point

b. Croisement multiple (multipoint)

Plusieurs auteurs se sont penchés sur l'utilisation de plusieurs points de coupure concernant l'opérateur de croisement. Le nombre de points de coupure généré est en moyenne $L/2$ où L est la taille du chromosome. L'individu est représenté sous la forme d'un anneau et l'échange de gènes entre les deux parents s'effectue de la façon suivante (voir figure 2.6) :

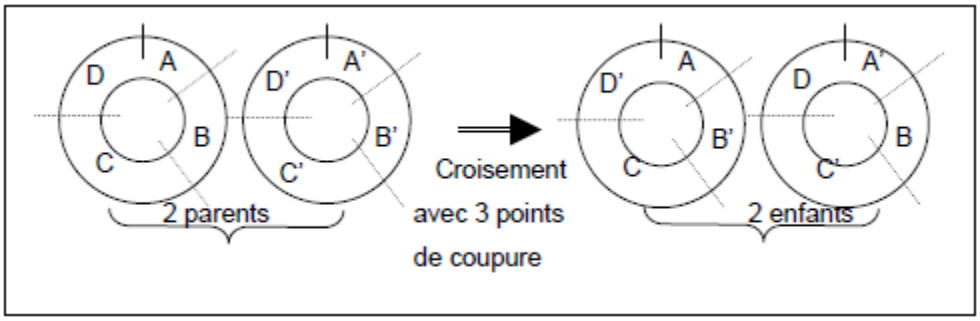


Figure3.6 : croisement avec 3 points de coupure

Cette technique s'applique autant pour une codification binaire que réelle des chromosomes. C'est une technique très utilisée dans différentes applications du fait que les résultats obtenus sont satisfaisants. **[A.E Eiben1999].**

c. Croisement uniforme

Cette technique est complètement différente des deux techniques précédentes. Un masque de croisement est généré aléatoirement pour chaque couple d'individus ou pour chaque génération. Les valeurs de ce masque sont binaires. Sa taille est identique à celle du chromosome. Son fonctionnement est illustré par la figure 2.7 :

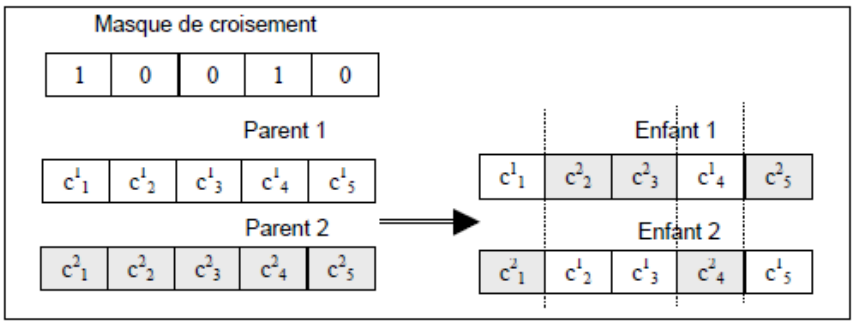


Figure3.7 : Croisement uniforme

Le fonctionnement du croisement uniforme est le suivant : si la valeur du bit du masque est égale à 1 alors la valeur du gène du parent1 est copiée chez l'enfant1 et si la valeur du bit du masque est égale à 0 alors la valeur du gène du parent2 est transmise à l'enfant1. Les valeurs des gènes de l'enfant2 sont les suivantes : les valeurs des gènes du parents1 lorsque la valeur du bit du masque est égale à 0 et les valeurs des gènes du parents2 lorsque la valeur du bit du masque est égale à 1.

Malgré que cette technique soit différente des deux autres au niveau conceptuel, on remarque qu'il existe une ressemblance entre ces techniques. Le croisement uniforme peut être un cas général des deux techniques. [**N. Benhamed2002]**

d. Croisement arithmétique

Cette technique a été développée par Michalewicz **[Z.Michalewicz 1992].**Lorsque cette opération est opérée sur les parents C^1 et C^2, deux enfants (progénitures) sont générés $H^k = (h^k_1 , k^k_2 , \dots , h^k_l , \dots , h^k_L)$ avec k=1 ,2 tels que

$$h^1_l = \lambda_l c^1_l + (1-\lambda_l)c^2_l \qquad \text{et} \qquad h^2_l=\lambda_l c^2_l + (1-\lambda_l)c^1_l$$

Dans le cas d'un croisement arithmétique uniforme, la valeur de λ_l i est une constante choisie par l'utilisateur, par contre si la valeur de λ_l est générée aléatoirement alors nous sommes dans le cas d'un croisement arithmétique non uniforme. λ appartient au domaine [0,*1*].

e. Croisement linéaire

Le croisement linéaire génère trois progénitures $H^k = (h^k_1 , k^k_2 , \dots , h^k_l , \dots , h^k_L)$ avec k=1, 2,3 tels que

$$h^1_l = 1/2 (c^1_l + c^2_l) \qquad h^2_l = 3/2\, c^1_l - 1/2\, c^2_l \qquad h^3_l = -1/2\, c^1_l + 3/2\, c^2_l$$

Lors de la sélection, la nouvelle génération sera formée des deux meilleurs de ces enfants. Nous remarquons que le domaine d'exploration est borné. Cet opérateur a été développé par Wright **[A.Wright 1991].**

f. Croisement discret

Les valeurs des gènes *hi* des enfants générés *H* sont égales aux valeurs des gènes c^1_i du parent C^1 ou aux valeurs des gènes c^2_i du parent C^2. Le choix du parent est effectué d'une façon aléatoire ayant une distribution uniforme. L'intervalle d'exploration est borné.

g. Croisement étendu (inter médiate)

Le gène *hi* du chromosome *H* généré par cet opérateur est défini comme suit :

$$h_i = c^1_i + \alpha\,(c^2_i - c^1_i)$$

Tels que la valeur de α est choisie aléatoirement ayant une distribution uniforme de l'intervalle *[-d, 1+d*]. Dans ce type de croisement, Muhlenbein **[Muhlenbein .H1993]** a déterminé la valeur de *d* à 0. Dans le cas contraire ce type de croisement est appelé le *croisement intermédiaire étendu*. Le bon choix de la valeur de *d* est 0.25 et la valeur de α varie pour chaque gène. L'opérateur de Muhlenbein ne s'applique que sur des gènes de type réel. Autrement dit la formule précédente s'écrit

$$h_i = c^1_i + \alpha_i\,(c^2_i - c^1_i)$$

Nous remarquons que les valeurs des gènes des chromosomes générés n'appartiennent plus à l'intervalle [*ai* , *bi*]. Ce qui fait que le domaine d'exploration est assez vaste.

h. Croisement heuristique

La même technique que le croisement étendu seulement

$$h_i = \tilde{c}^1_i + \alpha\,(c^1_i - \tilde{c}^2_i)$$

i. Choix de techenique approprié

Les avis des auteurs divergent en ce qui concerne le choix d'une technique de croisement .Plusieurs articles ont comparé un ensemble de techniques de croisement. Les conclusions sont assez diversifiées dû à la nature des différents problèmes

étudiés. Par exemple, Eshelman et al **[Eshelman L.J1989]** ont observé que le croisement multipoint (multiple) et uniforme donnent de meilleurs résultats. Il s'avère que le croisement avec 8 points de coupure est la meilleure technique pour obtenir une solution adéquate en fonction du problème à optimiser. De Jong et Spears **[Man K.F, & Tang K.S. 1997]** ont précisé que le croisement à 2 points de coupure ne fonctionne pas à la perfection lorsque la taille de la population est grande. Ils ont ajouté que les performances du système à optimiser sont robustes lors de l'utilisation du croisement uniforme sur une population de petite taille.

A partir de l'ensemble des techniques de croisement citées précédemment, nous constatons que peu importe la technique utilisée, il faut respecter le concept de base du croisement. Ce concept est de permuter les valeurs des gènes de deux parents pour former des progénitures et d'explorer le domaine de recherche dans l'espace des solutions. Pour faire un bon choix d'une technique de croisement, il suffit d'effectuer plusieurs expériences avec différentes techniques. Pour raffiner cette recherche, les techniques de sélection standards (par exemple la roulette biaisée) et l'opérateur de mutation vont nous permettre de converger plus rapidement vers des solutions.

Certaines remarques sur le croisement sont illustrées dans les points suivants :

- Le croisement est la clef de la puissance des AG. Il est directement lié à l'aptitude qu'a une population d'individus d'explorer son espace de recherche et de combiner entre eux les meilleurs résultats. Grâce au croisement, les AG se concentrent sur les parties les plus prometteuses de l'espace des solutions du fait que cet opérateur de croisement combine des chaînes contenant des solutions partielles.

- Le croisement n'est habituellement pas appliqué à toutes les paires d'individus choisis aléatoirement lors de la reproduction. La probabilité du croisement appliquée est comprise entre 0.6 et 1.0 **[A.E Eiben1999].**Dans le cas où le croisement ne s'appliquerait pas, alors les enfants sont semblables aux parents.

III.2.5.5 Opérateur de mutation

Cet opérateur est appliqué sur chaque chromosome issu de l'opération de croisement ou appartenant à une population. L'action de l'opérateur de mutation consiste à changer ou à permuter des valeurs des gènes du chromosome *C* avec une probabilité *pmut*. Cette probabilité de mutation est assez faible en pratique. On peut observer à un moment du cycle de l'AG que des individus générés n'évoluent plus. Pour les faire évoluer, l'opération de mutation peut modifier les valeurs des gènes

pour constituer des individus non similaires. Cet opérateur permet une recherche de la solution au problème à optimiser dans un domaine très restreint. L'utilité de cet opérateur est donc l'exploitation de l'espace de recherche des solutions. Ces mutations ne créent généralement pas de meilleures solutions au problème mais elles évitent l'établissement de populations uniformes incapables d'évoluer. Ceci permet à l'AG de converger vers des solutions globales. A partir d'une exploration de l'espace de recherche, la mutation permet de passer de l'exploration vers l'exploitation et de trouver un ensemble de solutions. Cependant, plusieurs techniques de mutation ont été développées dans la littérature. Certaines d'entre elles s'appliquent sur des gènes dont la représentation est binaire et d'autres sur des gènes de type réel. Pour déterminer le nombre de positions dont les gènes doivent subir un changement, il suffit de connaître

La taille du chromosome *L* et la probabilité de mutation *pmut*. Ce nombre est défini par le produit de *L* x *pmut*.

Posons la notation suivante du chromosome qui doit subir la mutation

$C = (c_1 , c_2 , \dots , c_i , \dots , c_L)$ tels que $ci \in [ai , bi]$. Le résultat obtenu par l'opérateur de mutation est un chromosome *C'* tels que $C' = (c'_1 , c'_2 , \dots , c'_i , \dots , c'_L)$

a. Mutation uniforme

Dans le cas binaire, Un ou plusieurs gènes, selon la taille du chromosome, et un taux de mutation $\frac{\text{gènes mutés}}{\text{gènes totals}}$ très faible sont choisis aléatoirement. Ils sont alors inverses (si la valeur du gène à muter est égale à 1 alors elle est inversée à 0 et si la valeur du gène est égale à 0 alors elle est inversée à 1).

La figure 2.8 illustre l'effet de la mutation sur une chaîne binaire d'un chromosome. La position du gène qui doit subir la mutation est déterminée aléatoirement.

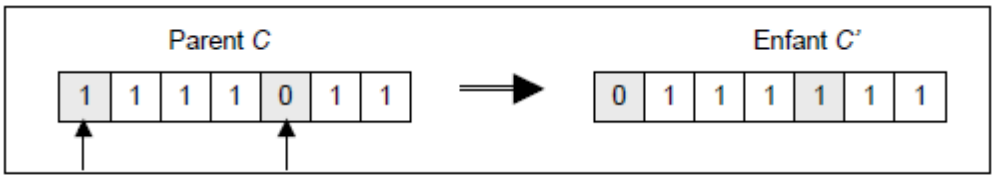


Figure3.8 : mutation uniforme binaire

Dans le cas réel, le gène *ci* est modifié par le gène *c'i*. Cette valeur du gène c'_i est choisie aléatoirement de l'intervalle $[a_i , b_i]$.

Le taux de mutation ´étant plus important (car un nombre plus grand de gènes sont mutés).

b. Mutation non uniforme

Cette technique est appliquée en fonction de la génération *t* courante et le nombre maximum de générations *gen_max*. Le gène *c'i* est défini comme suit

$$c'_i = \begin{cases} c_i + \Delta(t, b_i - c_i) & \text{si } \tau = 0 \\ c_i - \Delta(t, c_i - a_i) & \text{si } \tau = 1 \end{cases}$$

Où τ est généré aléatoirement tels que $\tau \in \{0, 1\}$ et

$$\Delta(t, y) = y\left(1 - r^{\left(1 - \frac{t}{gen\_\max}\right)^b}\right)$$

Où *r* est un nombre aléatoire de l'intervalle [0,1], et *b* est un paramètre choisi par l'utilisateur. Ce paramètre permet de déterminer le degré de dépendance sur le nombre d'itérations.

Cette fonction $\Delta(t, y)$ retourne une valeur appartenant à l'intervalle $[0 , y]$ et permet de définir les intervalles de types exploration et exploitation. Cette technique s'applique sur des gènes dont la représentation est réelle.

Pour visualiser le comportement de la c'_i (soit la notation *f(t)*), nous avons effectué quelques simulations en faisant varier le nombre maximum de générations de 0 à *gen_max* et la valeur du paramètre *b* sur la plage des valeurs comprise entre 0 et 100. Les résultats obtenus sont illustrés par les graphiques 1 et 2.

Nous avons choisi deux valeurs du nombre maximum de génération : 10 000 et 1000. Le graphique 1 représente le cas où le nombre de génération est égal à 10

000 et le graphique 2 représente le second cas ou le nombre de génération est égale à 1000. Le choix de ces deux valeurs permet d'analyser l'influence du nombre maximal de générations sur la forme de la fonction c'_i.

Posons $r = 0.0037$, $c_i = 0.0897$, $a_i = 0$ et $b_i = 1$. L'équation (c'_i) s'écrit

$$c'_i = 1 + (c_i - 1) r^{(1-\frac{t}{Nbre\_gen})^b}$$

- Si $\tau = 0$ alors
  - Si la valeur de *t* est très petite (tend vers 0)

    Alors $r^{(1-\frac{t}{Nbre\_gen})^b}$ tend vers *r*. et c_i' tend vers $1+r(c_i-1)$.
  - Si la valeur de *t* est très grande (tend vers *Nbre_gen*)

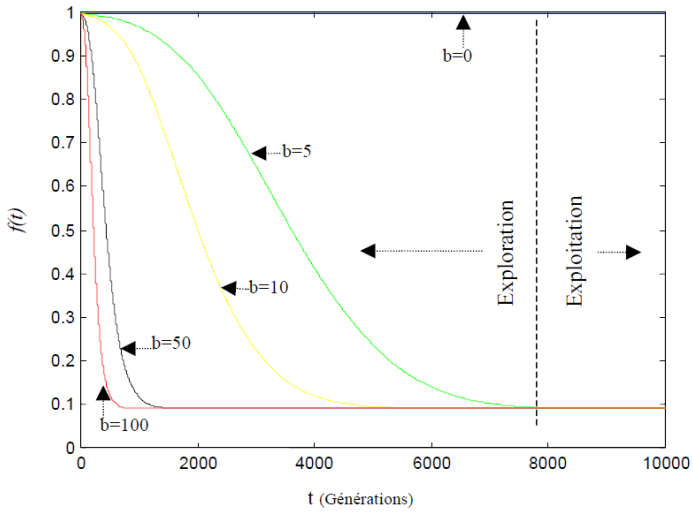
    Alors $r^{(1-\frac{t}{Nbre\_gen})^b}$ tend vers 1. Et c_i' tend vers c_i.

$$c'_i = c_i r^{(1-\frac{t}{Nbre\_gen})^b}$$

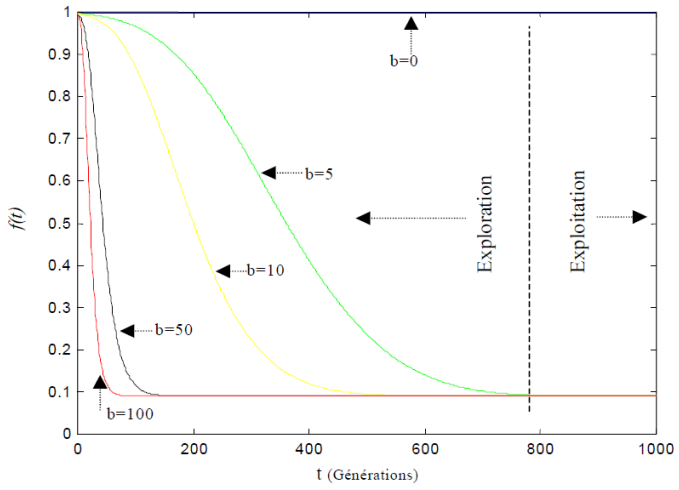
- Si τ = 1 alors
  - si la valeur de *t* est très petite (tend vers 0)

    Alors $r^{(1-\frac{t}{Nbre\_gen})^b}$ tend vers *r*. et c_i' tend vers $r\ c_i$.
  - si la valeur de *t* est très grande (tend vers *Nbre_gen*)

    Alors $r^{(1-\frac{t}{Nbre\_gen})^b}$ tend vers 1. Et c_i' tend vers c_i.



Graphique 1 - Opérateur de mutation non uniforme avec *gen_max*=10 000



Graphique 2 - Opérateur de mutation non uniforme avec *gen_max*=1000

Lorsque la valeur du paramètre *b* est égale à 0, nous observons dans les deux graphiques 1 et 2 que les valeurs des gènes des enfants sont toutes égales aux valeurs des gènes des parents et la fonction *f(t)* n'évolue pas. Ce cas de figure est bien entendu à éviter. Par contre lorsque la valeur de *b* augmente, le domaine d'exploration diminue au fur et à mesure et le domaine d'exploitation s'agrandit. Par exemple lorsque la valeur de *b* = 5, nous observons que jusqu'à la génération 8000 (graphique 1) et la génération 800 (graphique 2), l'AG ne fait qu'explorer l'espace des solutions. Pour le reste des générations, la phase d'exploitation est entamée pour trouver les solutions globales. Dans le cas ou le nombre de générations est de 10000, le temps de calcul est plus grand que celui de 1000 générations et la solution obtenue

est identique dans les deux cas. Nous constatons que le choix du nombre de génération n'influe pas sur les résultats. Pour le choix de la valeur du paramètre *b,* il est préférable qu'elle soit égale à 5 car l'espace de recherche est très bien exploré et le passage vers l'exploitation donne une solution assez rapide. Par contre lorsque la valeur de *b* est assez grande (supérieure à 10), l'espace de recherche n'est pas exploré de façon satisfaisante contrairement à l'exploitation.

c. Mutation gaussien

La *mutation Gaussienne* consiste à ajouter un bruit Gaussien aux composantes du vecteur individu concerné, ce qui implique l'ajustement d'un paramètre supplémentaire σ la déviation standard de ce bruit :

$$\forall i \in 1, ..., n, \quad x_i' = x_i + N(0, \sigma)$$

L'ajustement de Q est relativement complexe (trop petit, il ralentit l'évolution, trop grand, il perturbe la convergence de l'AG), de nombreuses stratégies ont été proposées, consistant à rendre ce paramètre variable au cours de l'évolution, soit en fonction du temps, de la valeur de fitness, dépendant des axes de l'espace de recherche (mutations non isotropes), ou encore auto-adaptatif, comme ci-après. Des études ont aussi été conduites sur l'emploi de bruits non Gaussiens.

Dans MATLAB deux paramètres utilisés pour ajuster cette grandeur **scale** et **shrink**

Scale : détermine L'ajustement de Q dans la1 ère génération qui dépond de « initial range » c.à.d. l'intervalle de variation de la population initial

Exp : si population initiale varie de [2,1], donc l'ajustement deQ est le même à toutes les coordonnées de vecteur parentale tel que :

Var1=scale *(v(2)-v(1))

Shrink : détermine l'ajustement aux les autres générations, de la k_{eme} génération

Var $_k$=var $_{k-1}$(1-shrink*(k/gen-max))

d. La *mutation auto-adaptative*

La *mutation auto-adaptative* est une des grandes innovations des stratégies évolutionnaires, et consiste à faire gérer les paramètres directement par l'AG, en l'intégrant au code génétique. Les individus sont donc des vecteurs (x,σ)., et la mutation se traduit simplement comme suit :

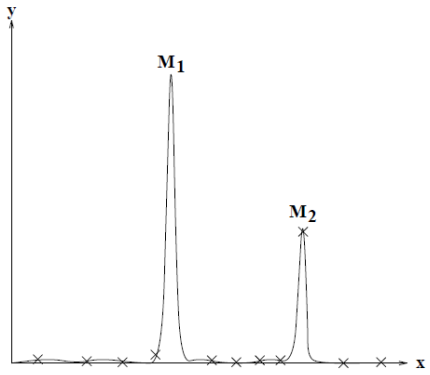
$$\forall i \in 1, ..., n, \quad \sigma_i' = \sigma_i exp(N(0,\tau) \quad et \quad x_i' = x_i + N(0,\sigma_i')$$

De nombreux travaux théoriques et expérimentaux ont montré l'intérêt et la puissance de ces méthodes auto-adaptatives.

III.2.6 Amélioration classique des AGs

III.2.6.1 Introduction

Les processus de sélection présentés sont très sensibles aux ´écarts de fitness et dans certains cas, un très bon individu risque d'être reproduit trop souvent et peut même provoquer l'´elimination complète de ses congénères; on obtient alors une population homogène contenant un seul type d'individu. Ainsi, dans l'exemple de la figure 6 le second mode M2 risque d'ˆêtre le seul représentant pour la génération suivante et seule la mutation pourra aider à atteindre l'objectif globalM1 au prix de nombreux essais successifs.



x : individus

Figure 3.9: Exemple o`u les sélections classiques risquent de ne reproduire qu'un individu

Pour ´éviter ce comportement, il existe d'autres modes de sélection (*ranking*) ainsi que des principes (*scaling, sharing*) qui empêchent les individus "forts" d'´eliminer complètement les plus "faibles".

III.2.6.2 Scaling

Le *scaling* ou mise `a l’´echelle, modifie les fitness afin de réduire ou d’amplifier artificiellement les écarts entre les individus. Le processus de sélection n’opère plus sur la fitness réelle mais sur son image après scaling. Parmi les fonctions de scaling, on peut envisager le, , scaling linéaire ,scaling top et le scaling exponentiel. Soit fr la fitness avant scaling et fs la fitness modifiée par le scaling.

a. scaling rang (Rankscaling)

Dans rankscaling, Chaque chromosome se voit associé un rang en fonction de sa position. Le plus mauvais chromosome aura le rang 1, le suivant 2, et ainsi de suite jusqu'au meilleur chromosome qui aura le rang N (pour une population de N chromosomes). Avec cette méthode, tous les chromosomes ont une chance d'être sélectionnés. Cependant, elle conduit à une convergence plus lente vers la bonne solution. Ceci est dû au fait que les meilleurs chromosomes ne diffèrent pas énormément des plus mauvais **[C. Bontemps 1995]**

b. Scaling linéaire

Dans ce cas la fonction de scaling est définie de la façon suivante

$$f_s = a f_r + b$$

$$a = \frac{max' - min'}{max - min};\ b = \frac{min'.max - min.max'}{max - min}.$$

En règle générale, le coefficient a est inferieur `a un, ce qui permet de réduire les écarts de fitness et donc de favoriser l’exploration de l’espace. Ce scaling est statique par rapport au numéro de génération et pénalise la fin de convergence lorsque l’on désire favoriser les modes dominants.

c. Scaling exponentiel

Il est défini de la façon suivante

$$f_s = (f_r)^{k(n)}$$

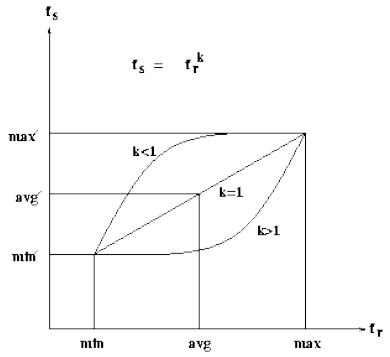


Figure 3.10: Fonction de scaling exponentielle

Où *n* est la génération courante.

- Pour *k* proche de zéro, on réduit fortement les écarts de fitness ; aucun individu n'est vraiment favorisé et l'algorithme génétique se comporte comme un algorithme de recherche aléatoire et permet d'explorer l'espace.
- Pour *k* proche de 1 : le scaling est inopérant.
- Pour *k* > 1 les écarts sont exagérés et seuls les bons individus sont sélectionnés ce qui produit l'émergence des modes.

Dans la pratique, on fait généralement varier *k* des faibles valeurs vers les fortes valeurs au cours des générations. Pour cela on peut utiliser la formule suivante :

$$k = \left(\tan \left[\left(\frac{n}{N+1} \right) \frac{\pi}{2} \right] \right)^p$$

n étant la génération courante, *N* le nombre total de générations prévues, *p* un paramètre à choisir. Le choix de *p*=0.1 s'est avéré pertinent dans les applications. L'évolution de *k* en fonction de la génération *n* est donnée par la figure 3.11.

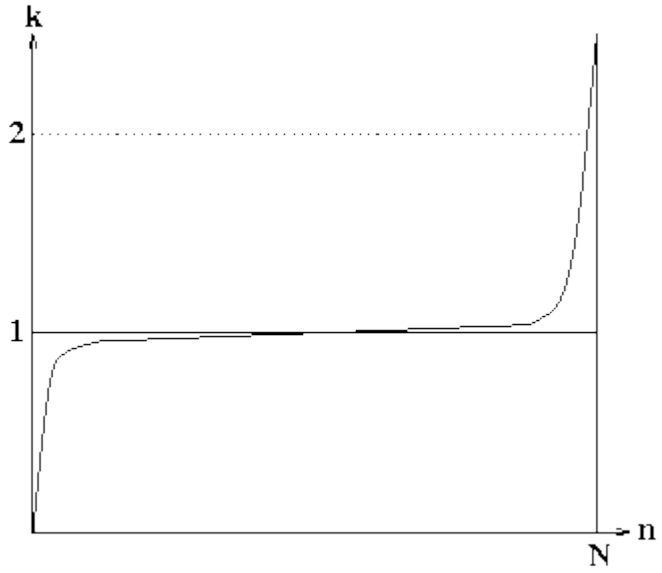


Figure 3.11: Allure de l'évolution de *k* en fonction des générations

Ce dernier principe de scaling donne effectivement de meilleurs résultats sur nos problèmes que le scaling linéaire et sera donc systématiquement utilisé. Dans le cas des fonctions objectifs multi-modes présentant des optimaux quasi-équivalents, cette technique de scaling, en amplifiant les écarts de fitness en fin de convergence, va effectivement favoriser le mode dominant mais aussi masquer les modes sous-optimaux qui peuvent tout de même présenter un intérêt. Le scaling permet donc une bonne exploration de l'espace d'état mais ne favorise pas la répartition des individus sur les différents modes de la fonction objective.

d. Scaling proportionnel

Cette technique fait tourne les résultats autour de sa moyenne

$$\boldsymbol{fs = 2 * moyenne(fr) - fr}$$

Scaling proportionnel à des inconvenant quand le bon individu n'est dans une bonne gamme

e. Scaling top

La valeur de fitness scaling basé sur les résultats et la valeur de top

Exp : si top=0.4, 40%des individus meilleurs prennent la valeur de1 Le reste prend la valeur0

III.2.6.3 Sharing

I.2.7 Introduction

L'objectif du sharing est de répartir sur chaque sommet de la fonction à optimiser un nombre d'individus proportionnel à la fitness associée à ce sommet. La figure 3.12 présente deux exemples de répartitions de populations dans le cas d'une fonction à cinq sommets : le premier sans sharing, le second avec sharing.

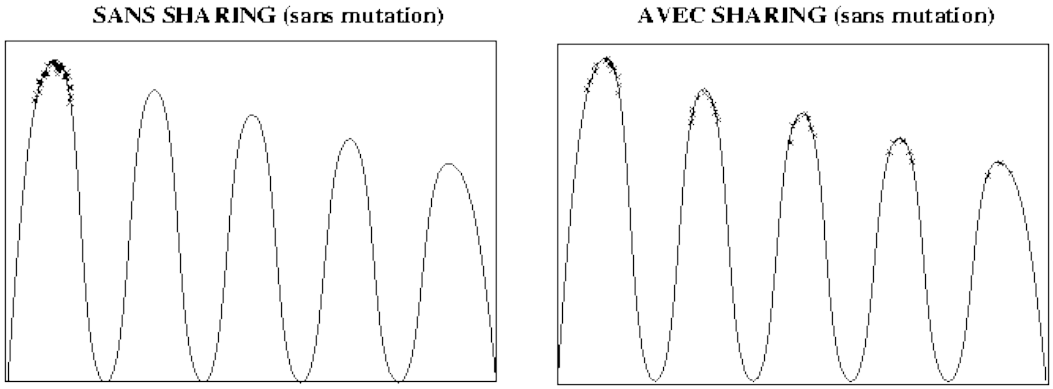


Figure 3.12: Objectif du sharing

II.2.7 Principe

De la même façon que le scaling, le sharing consiste à modifier la fitness utilisée par le processus de sélection. Pour éviter le rassemblement des individus autour d'un sommet dominant, le sharing pénalise les fitness en fonction du taux d'agrégation de la population dans le voisinage d'un individu. Il requiert l'introduction d'une notion de distance. Dans la pratique, il faut définir une distance indiquant la dissimilarité entre deux individus. Cette distance est alors utilisée pour calculer la nouvelle fitness de la façon suivante :

$$f'_i = \frac{f_i}{m'_i};\ m'_i = \sum_{j=1}^{N} S(d(x_i, x_j))$$

Avec

$$S(d) = 1 - \left(\frac{d}{\sigma_{share}}\right)^{\alpha} \text{ si } d < \sigma_{share}$$

$$S(d) = 0 \text{ si } d > \sigma_{share}$$

Le paramètre σ_{share} permet de délimiter le voisinage d'un point et dépend du problème traité. La figure 3.13 donne l'allure de *S*(*d*) pour différentes valeurs de a. Suivant la valeur donnée à a le sharing sera plus ou moins efficace. Ainsi pour a < 1, on pénalise les groupes très agglomérés.

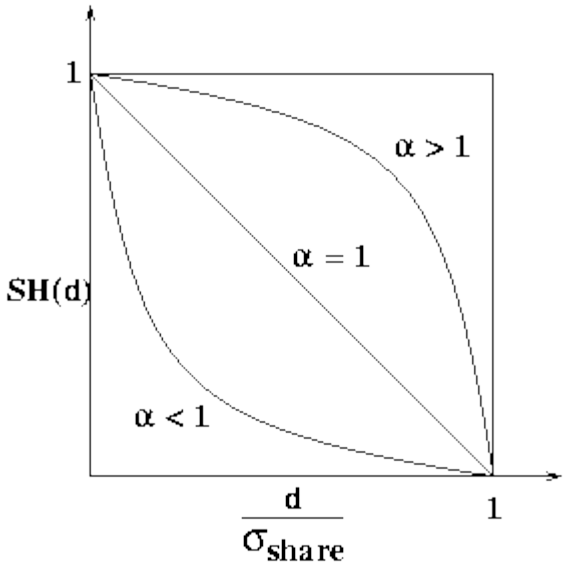


Figure 3.13: Allure de $S(\frac{d}{\sigma_{share}})$

Dans la pratique ce type de sharing donne effectivement de bons résultats mais au prix de N^2 calculs de distances entre chromosomes à chaque génération pour une population de taille *N*. Or les algorithmes génétiques induisent une complexité en *N* sans sharing et le fait de passer en N^2 peut être pénalisant, notamment pour *N* grand.

III.2.7 Le choix des paramètres d'un algorithme génétique

Pour lancer l'AG, il faut définir certains paramètres tels que : la taille de la population, les probabilités de mutation et de croisement et le nombre de générations. Il est difficile de les fixer ou de trouver les meilleurs avant l'exécution de

l'algorithme **[A.E Eiben1999]** .C'est un problème de réglage qui doit être optimisé pour chaque type de problème traité.

Cela constitue une part importante du travail de l'expérimentateur. Dans la littérature, la définition de ces paramètres diffère d'une application à une autre. Les paramètres que nous abordons sont : la taille de la population, le nombre de générations et les probabilités de reproduction.

III.2.7.1 Taille de la population

L'AG nécessite la détermination du nombre d'individus qui constituent la population *P*.

Le problème qui se pose est comment fixer la taille de cette population. Une population trop petite évolue probablement vers un optimum local peu intéressant. Une population trop grande met plus de temps pour converger vers des solutions envisageables. La taille de la population doit être choisie de façon à réaliser un bon compromis entre le temps de calcul et la qualité du résultat. Eiben et al **[A.E Eiben1999]** mentionnent que plusieurs chercheurs se sont penchés sur ce problème qui consiste à déterminer la taille optimale de la population pour atteindre la meilleure solution. Certains d'entre eux l'ont fixé entre 20 et 100 de manière empirique. D'autres essayent de l'ajuster en respectant le taux d'erreur de sélection ou de la faire varier d'une population à une autre. Par contre, Bautista et al **[Bautista et al1999]** ont fixé le nombre d'individus à 50 et ont mentionné qu'il peut exister une relation entre la taille de la population et le nombre de gènes. Eiben et al recommandent que la taille d'une population moyenne soit comprise entre 20 et 30.

III.2.7.2 Probabilité des opérateurs génétiques

Le choix de la probabilité de mutation *pmut* et la probabilité du croisement *pcross* est un problème d'optimisation non linéaire complexe à résoudre. En outre, pour les fixer, ces probabilités de mutation et de croisement dépendent de la nature de la fonction objective du problème à résoudre. Man et al **[Man K.F., et al 2000]** ont cité quelques exemples tels que :

Pour une population de taille grande = 100

pmut = 0.001 *pcross* = 0.6

- Pour une population de taille petite = 30

pmut = 0.01 *pcross* = 0.9

Mitchell **[Mitchell M. (1996)].** a, quant à elle, suggéré de fixer *pmut* à 0.001 et *pcross* à 0.6. Einben et al **[A.E Eiben1999]** rapportent une formule proposée par plusieurs auteurs pour déterminer la probabilité de mutation. Cette probabilité dépend de la taille du chromosome et est exprimée par la l'équation suivante

$$p_{mut} = 1/L$$

Où *L* est la taille du chromosome en gènes.

Lorsque la technique de mutation est non uniforme, l'équation ne peut être appliquée. Car cette technique nécessite une mutation sur plusieurs valeurs des gènes. Ainsi, il préférable de choisir une probabilité qui approxime *10*/*L* au lieu de la probabilité *1/L*. La probabilité *1/L* signifie la mutation maximale d'un gène dans un chromosome de taille *L*.

Concernant la probabilité de croisement, les recherches n'ont pas pu déterminer la valeur optimale due au fait que la probabilité de croisement s'opère sur une paire de chromosomes contrairement à la probabilité de mutation qui s'opère sur les gènes de 52 chaque chromosome. Les probabilités de croisement les plus couramment utilisées appartiennent à l'intervalle [0.6, 0.95]. Il est préférable de ne pas utiliser un taux trop faible. Eiben et al **[A.E Eiben1999]** mentionnent que les valeurs inférieures à 0.6 sont rarement utilisées

III.2.8 Avantage des AGs

Les algorithmes génétiques diffèrent fondamentalement des autres méthodes d'optimisation :

- Les AG possèdent une représentation codée et cherchent une représentation dans l'espace des solutions et non pas directement dans le domaine original
- Les AG travaillent sur une population des points, ou lieu d'un point unique
- Les Ag utilisent des règles de transitions probabilistes et non déterministes
- Les AG n'utilisent que les valeurs de la fonction à optimiser, pas sa dérivée ou une autre information auxiliaire

III.2.9 Inconvenants des AGs

- Pas de garantie de convergence en un temps fini

- Faiblesse des fondements théoriques et mathématiques
- Souvent gourmands en calcul donc lents (mais aisément parallélisables)
- Chaque individu doit être évalué, même les individus inadaptés (utilisation offline, sur simulateur)
- Produit toujours des individus inadaptés
- Le comportement de l'algorithme dépend d'un grand nombre de paramètres qui peuvent être difficiles à fixer

III.3 Hybridation des RNA avec les AG

Le concept de système hybride ou de méthode hybride est très large. Ce groupe d'applications inclut toute méthode qui intègre au moins deux approches différentes pour la solution d'un problème donné. Cette hybridation permet de tirer les avantages des deux techniques hybridées.

Plusieurs approches peuvent être hybridées :

- Systèmes Symbolique-Flous
- Systèmes Symbolique-Génétiques
- Systèmes Génético-Flous
- Systèmes Neuro-Symboliques
- Système neuro-génetique

III.3.1 Système neuro-génetique

Bien que des réseaux neuraux soient employés pour résoudre une variété de problèmes, ils ont toujours quelques limitations. Une est associée à l'apprentissage ave c la rétropropagation qui est souvent employé mais elle présente un inconvénient sérieux : **minima locaux** . Les RNA présente Une autre difficulté sérieuse c'est le **choix d'une topologie optimale. (voir chapitre1)**

L'algorithme génétique est une technique d'optimisation efficace qui peut guider, a une optimisation des poids globale et un choix de topologie optimale.

III.3.2 Utilisation des ag pour une optimisation des poids(apprentissage paramétrique)

La partie suivante présentent le concept de base d'une technique d'optimisation de poids genetique (le Montana et David, 1989; blanchement et Hanson, 1989; Ichikawa et sawa, 1992).

Pour une utilisation des algorithmes génétiques, il faut d'abord représenter le domaine de problème comme un chromosome. Par exemple, nous voulons optimiser les poids d'un perceptron multicouche présenté dans la figure 3.14

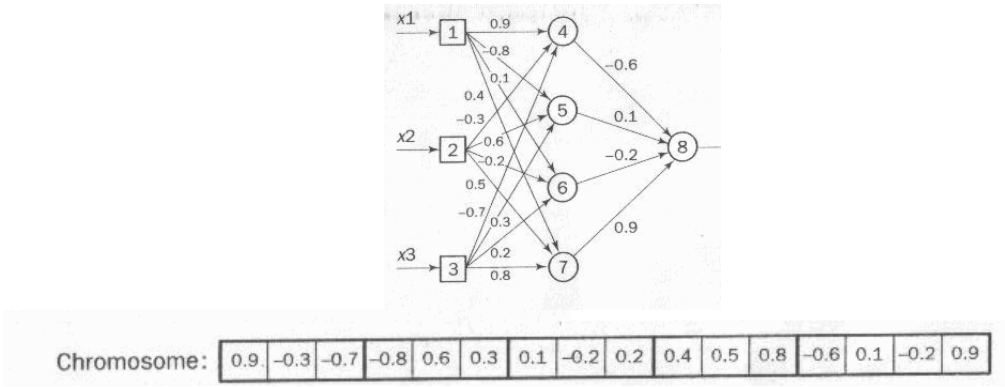


Figure3.14: présentation d'un système neuro-génétique

Dans la première étape on va générer Des poids initiaux dans le réseau choisi aléatoirement dans le petit intervalle [-1,1]. Dans ce perceptron, il y a 16 liaisons pondérées entre les neurones. Puisqu'un chromosome est un ensemble de gènes, l'ensemble des poids peut être représenté par un chromosome à 16 gènes, où chaque gène correspond à une liaison simple pondérée dans le réseau. Ce chromosome présente un individu d'une population c.ad une solution proposé à partir d'un ensemble des solutions

Dans La deuxième étape on doit définir une fonction d'évaluation (fitness) pour évaluer la performance des chromosomes. Cette fonction doit estimer la performance d'un réseau neuronal donné. Nous pouvons appliquer ici une fonction

assez simple définie par la réciproque de l'erreur quadratique telle que l'algorithme génétique essaye de trouver un ensemble des poids (individu) qui réduisent au minimum la somme d'erreurs quadratique

On peut utiliser aussi comme une fonction le taux de classification non correcte et l'algorithme génétique essaye de trouvé l'individu qui réduise aux minimum ce taux

La troisième étape on doit appliquer les deux opérateurs des algorithmes génétiques croissement et mutation. Un opérateur de croisement prend deux chromosomes parentaux et crée un enfant simple avec le matériel génétique des deux parents. Chaque gène dans le chromosome de l'enfant est représenté par vous la transmission des parents aléatoirement choisi. Figure 3.15 montre une application de l'opérateur de croisement.

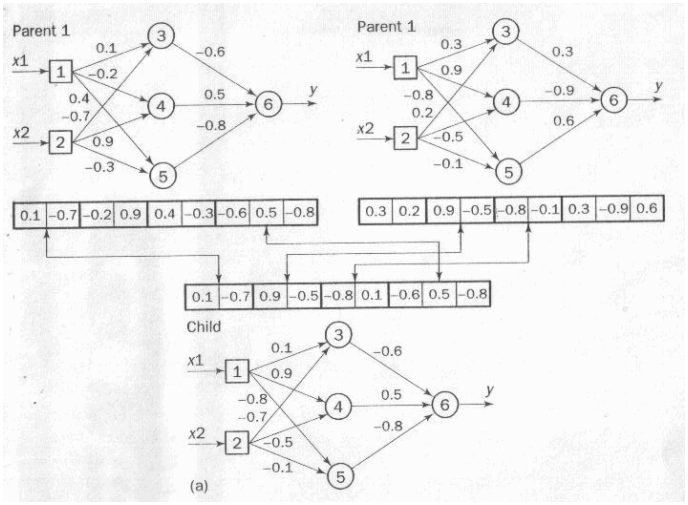


Figure3.15: opérateur de croisement dans un système neuro-génétique

Un opérateur de mutation choisit aléatoirement un gène dans un chromosome et ajoute une petite valeur aléatoire à chaque poids dans ce gène. Figure 3.16 montre un exemple de mutation.

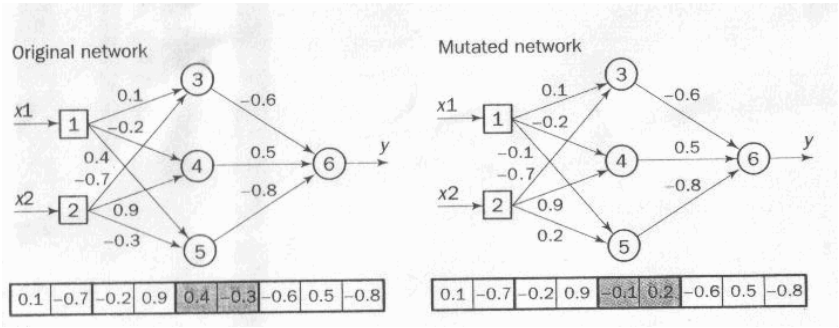


Figure3.16 opérateur de mutation dans le système neuro-génetique

Maintenant nous sommes prêts à appliquer l'algorithme génétique. Bien sûr, nous devons toujours définir la taille de population, c'est-à-dire le nombre de réseaux avec des poids différents, la probabilité de croisement et de mutation et le nombre de générations.

Jusqu'ici nous avons assumé que la structure du réseau est fixée et l'apprentissage génétique est employé seulement pour optimiser les poids dans le réseau donné. Cependant l'architecture du réseau (c'est-à-dire le nombre de neurones et les connections entre les neurones détermine souvent l'échec de l'application.

D'habitude l'architecture de réseau est décidée par l'essai et l'erreur; il y a un grand besoin d'une méthode pour concevoir automatiquement l'architecture pour une application particulière. Des algorithmes génétiques peuvent bien nous aider dans la sélection de l'architecture de réseau

III.3.3 Utilisation des AGs pour une optimisation de la topologie(apprentissage structurel)

L'idée de base derrière le développement d'une architecture de réseau appropriée est de conduire une recherche génétique dans une population d'architecture possible bien sûr, nous devons d'abord choisir une méthode de codage l'architecture d'un réseau dans un chromosome.

Il y a beaucoup de façons différentes de coder la structure. Il faut décider combien d'information est exigée pour une représentation de réseau.

Plus il ya des paramètres d'architecture, plus le coût informatique augmente. Comme une illustration, nous pouvons considérer une méthode simple directe de codage Bien que le codage soit une technique direct limité et peut être appliqué

seulement aux réseaux non récurrente avec un nombre fixé de neurones, et on code les connexions entre les neurones pour les évaluer.

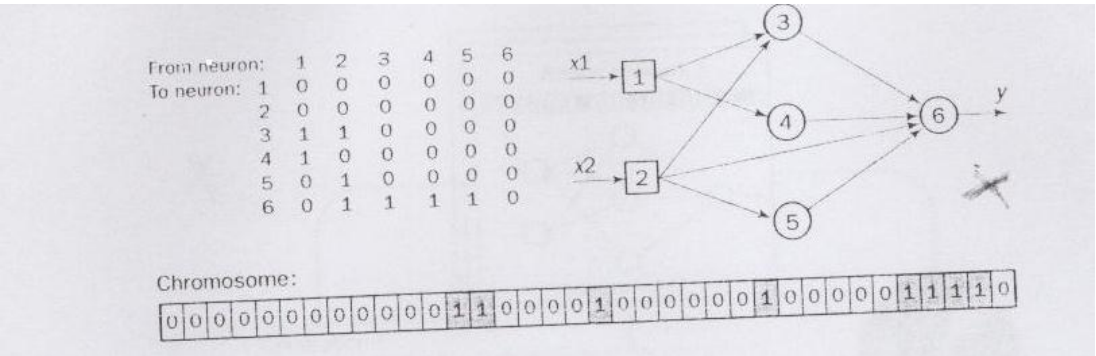


Figure3.17 : système neuro-genetique pour le choix de la topologie

La topologie de connexion d'un réseau neuronal peut être représentée par une matrice de connectivité carrée, comme indiqué dans la figure3.17. Chaque entrée dans la matrice définit le type de connexion d'un neurone (de la colonne a la ligne)où 0 présente aucune connexion et 1 présente la connexion. Pour transformer la matrice de connectivité dans un chromosome, nous devons seulement tendre les rangées de la matrice ensemble, comme indiqué dans la figure3.17.

Étant donné un exemple recevant une formation et une représentation binaire pour des architectures de réseau possibles. AG de base peut être décrite par les étapes suivantes :

Étape 1 : choisissez la taille d'une population de chromosome, probabilité de croissement et mutation et définissez le nombre de génération.

Etape 2 : définissez une fonction d'évaluation « fitness » pour mesurer la performance d'un chromosome individuel. En général, fitness du réseau doit être basée non seulement sur son exactitude, mais aussi sur sa vitesse d' d'apprentissage, la taille et la complexité. Cependant, performance du réseau est beaucoup plus importante que sa taille et donc la fonction fitness peut toujours être définie par la la somme d'erreurs quadratique

Etape 3 : produisez aléatoirement une population initiale de chromosomes

Etape 4 : appliquez chaque chromosome un algorithme d'optimisation des poids : soit rétro propagation, soit AG.

Etape 5 : choisissez une paire de chromosomes pour l'accouplement, avec une probabilité proportionnée à leur fitness

Etape6: créez une paire de chromosomes de résultat en appliquant les opérateurs de croisement et mutation

L'opérateur de croisement choisit aléatoirement un index de rangée et échange simplement les rangées correspondantes entre deux parents, créant deux résultats. L'opérateur de mutation donne un petit coup à un ou deux gènes dans le chromosome avec quelque probabilité basse, dire 0.005.

Etape7:placez les chromosomes créés dans la nouvelle population.

Etape8: répétez le processus de l'étape 4 jusqu'an le nombre de génération maximale

Un cycle génétique pour développer une topologie de réseau neurale est présenté dans figure3.18

En plus de l'apprentissage de réseau neuronal et le choix de topologie, le calcul évolutionnaire (algorithme génétique) peut aussi être employé pour optimiser des fonctions de transfert et choisir des variables d'entrée appropriées.

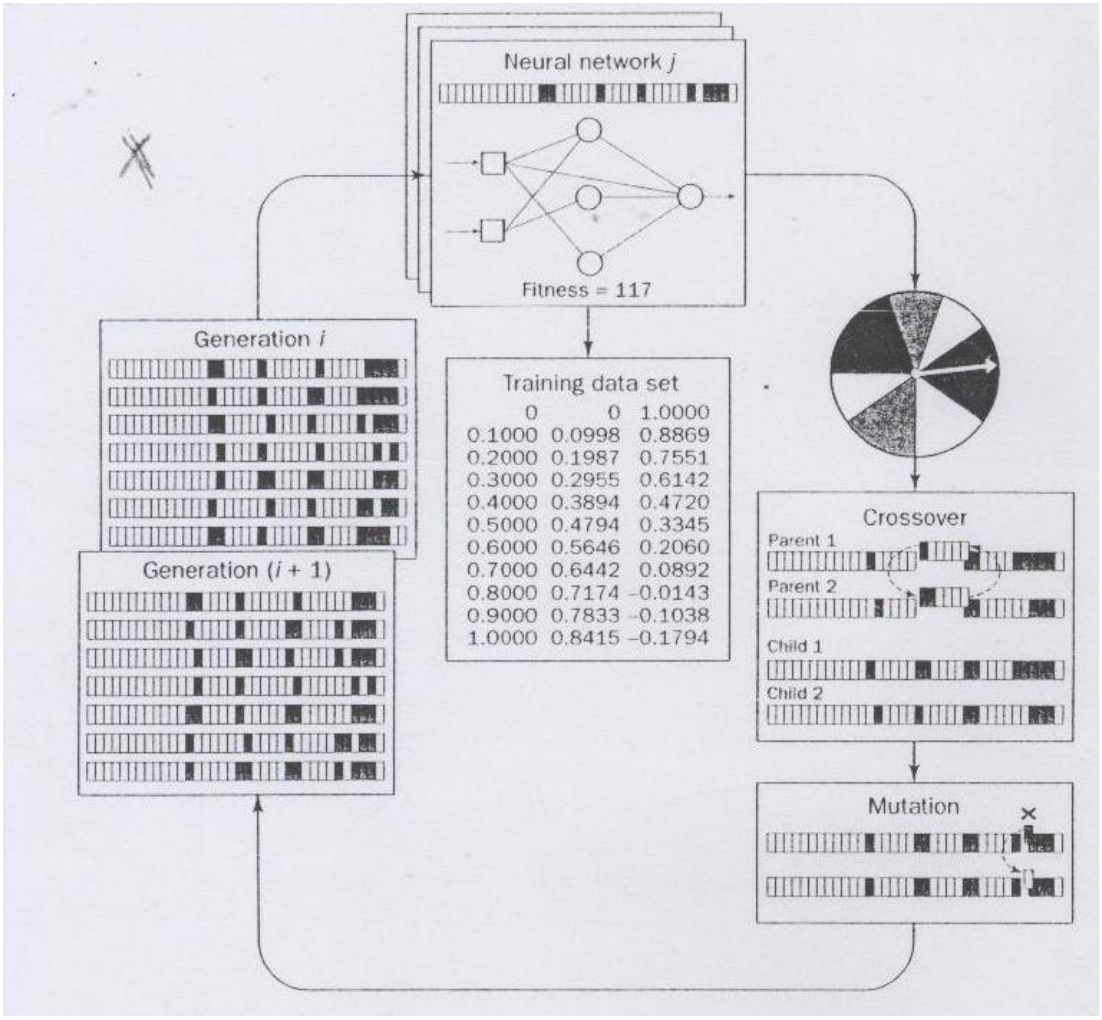


Figure3.18 : Un cycle génétique pour développer une topologie de réseau neurale

III.4 Conclusion

Ce chapitre nous a permis d'avoir une vue générale sur les concepts des AGS, et le principe d'hybridé avec les réseaux de neurone.

Nous pouvons conclure que les AGs sont des algorithmes simples de conception et peuvent résoudre des problèmes assez complexes (choix de la topologie neuronale, apprentissage synaptique). La résolution de ces problèmes est obtenue grâce aux opérateurs de reproduction. Ces AG sont des procédures assez robustes pour résoudre un problème d'optimisation pour la sélection des primitives. Néanmoins elles présentent certaines limites et des difficultés. Ces difficultés reposent sur le choix des bons paramètres tels que : la taille de la population, le nombre de génération, les probabilités de croisement et de mutation et les méthodes des opérateurs de reproduction. Ces paramètres dépendent du problème à résoudre et d'une codification appropriée au problème à solutionner.

CHAPITRE 4

résultats & interprétation

IV.1 Introduction

Dans ce chapitre nous intéressant a hybridé les algorithmes génétiques avec les réseaux de neurones pour réaliser des classifieurs pour les signaux biomédicaux.

L'hybridation neuro-génétique a permet d'éviter le problème des minima locaux. Nous avons mené plusieurs expérimentations pour comparer cette méthode avec la méthode classique (descente de gradient).

Nous avons ciblé quelques arythmies cardiaques comme : l'ESV, BBD, BBG et cas normal.

IV.2 Présentation électro cardiographique

IV.2.1 Signal électrocardiogramme

L'électrocardiographie est une technique non invasive, permettre la détection et l'enregistrement des variations cyclique de l'activité électrique du cœur (voir annexe D) en fonction du temps sous forme d'un signal appelé **électrocardiogramme. (ECG)**

Le signal ECG est enregistré par un appareil appelé **électrocardiographe,** on peut le considère comme un voltmètre qui enregistre des potentiels électriques générés par la dépolarisation du muscle cardiaque (voir annexe D).

IV.2.2 Caractéristiques d'un signal ECG normal

Chaque cycle de dépolarisation et repolarisation du cœur correspond au passage du courant électrique, chez les sujets sains, les oreillettes vers les ventricules qui se contractent dans le même ordre. Comme on a vu ce processus se traduit sur le plan électrocardiographe par l'enregistrement toujours dans le même ordre des différentes ondes **P, Q, R, S, T** et **U.**

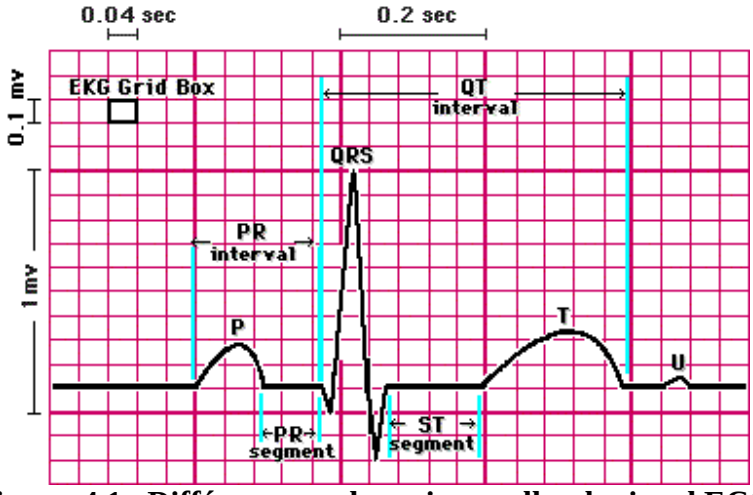


Figure 4.1 : **Différentes ondes et intervalles de signal ECG**

- **Onde P ;** onde correspondant à la dépolarisation des oreillettes droites et gauches ; elle est de forme arrondie, souvent positive de faible amplitude (1-3mv) et avec une durée de 0.12s.

- **Complexe QRS :** ensemble des ondes correspond à la dépolarisation des ventricules.

La durée du QRS est normalement comprise entre 0.06 et 0.10 seconde. Au-delà de0.12s en évoque un trouble « majeur » de conduction intra –ventriculaire, au-delà de 0.12s on peut parler de trouble « mineur » de conduction intra ventriculaire.

- **Onde T :** onde correspondant à la repolarisation ventriculaire **;** sa durée normale est de 0.25s.

- **Onde U :** est le témoin d'une repolarisation tardive de zones myocardiques, c'est une onde inconstante qui suit l'onde T, habituellement elle est moins de 0.2mv

Entre ces différentes ondes s'inscrive sur le tracé des intervalles qui sont :

- **Intervalle PR :** correspond au temps de conduction auriculo-ventriculaire, c'est-à-dire au temps que met l'influx à se rendre du nœud sinusal au myocarde ventriculaire. Il dure normalement de 0.12à0.20s.

- **Intervalle QT :** intervalle entre le début de la dépolarisation ventriculaire et la fin de repolarisation ventriculaire. On le mesure entre le début du complexe QRS et la fin de l'onde T. sa durée et de 0.4s pour un rythme normale.

- **Intervalle RR :** sépare les sommes de deux ondes R consécutives et définie le rythme ventriculaire**. [01]**

- **Intervalle PP :** sépare les sommes de deux ondes P consécutives et définie le rythme auriculaire**. [01]**

IV.2.3 Arythmies cardiaques

L'électrocardiogramme permet de détecter deux catégories d'anomalies des complexes auriculaires et ventriculaires : des anomalies de leurs morphologies d'une part, des anomalies de leurs origines, de leurs durées et / ou de leurs successions d'autre part. Ces dernières anomalies correspondent aux troubles du rythme et aux troubles de conduction.

Il existe plusieurs types d'anomalies d'ECG, mais dans ce travail on va utiliser battement normal plus 3 anomalies les plus rencontrés : extrasystole ventriculaire (ESV) , bloc de branche droit(BBD), bloc de branche gauche(BBG)

IV.2.3.1 Extrasystole ventriculaire

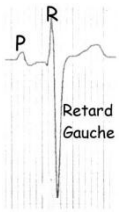
C'est une contraction ventriculaire prématurée, correspondant à une hyperexcitabilité focale. Elles apparaissent donc sur un tracé électrocadiographique comme un complexe QRS prématuré, non précédé d'une onde P et toujours large.

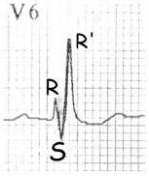
Figure 4.2 : Extrasystole ventriculaire

IV.2.3.2 Blocs de branches gauches

Le tracé ECG est caractérisé par :

- Des complexes QRS larges (>0.12sec).
- Complexes en formes de M dans les dérivations V5 et V6.
- Onde R par perte de Q en D1.





V1 V6

Figure4.3 Bloc de branche gauche dans les dérivations V1ert V6

IV.2.3.3 Blocs de branches droites

Le tracé ECG est caractérisé par :

- Des complexes QRS larges (>0.12s).
- Complexes en formes de M dans les dérivations V1 et V2.
- Onde S large en D1.

V1 V6

Figure 4.4 : **Bloc de branche droite dans les dérivations V1ert V6**

IV.3 classifieurs des arythmies cardiaques à classes

Dans cette partie, Nous appliquons un PMC (perceptron multicouche) pour différencier le battement ventriculaire prématuré (BVP) du cas normal.

Nous définissons 4 groupes de classifieurs (GP1, GP2, GP3, GP4). Et dans chaque groupe, nous définissons un certain nombre de classifieurs neuronaux (selon le nombre de neurones cachés).

Puis nous appliquons pour chaque classifieur les deux types d'apprentissage : classique (rétro propagation) et génétique.

IV.3.1 Sélection de la base d'exemples

Nous utilisons, dans le cadre de ce mémoire, la base de données MIT-BHI (voir annexe G). Cette base se compose de quarante huit enregistrements (deux voies ECG ambulatoire d'environ trente minute), dont les instants et la classe de tous les cycles sont annotés par deux cardiologues différents.les enregistrements comprenant suffisamment de battements ventriculaires prématurés ont été choisis pour construire la base d'exemple, voir tableau 4.1

Nous avons rassemblé les battements de chaque enregistrement en deux groupes :

Le groupe N est composé des battements différents de BVP

Le groupe B est composé des battements BVP

Type d'enregistrement	Nombre des battements « N »	Nombre des battements « V »
105	2549	41
106	1498	97
114	1810	44
116	2273	108
119	1535	442
200	1737	820
210	2423	194
215	3181	164
219	2071	64
221	2022	393
228	1696	97
233	2220	824

Tableau 4.1 : les enregistrements choisis de la base de données MIT-BIH

La base d'apprentissage affecte la performance des classifieurs neuronaux. Le critère général à respecter pour l'élaborer est de sélectionner des exemples représentatifs de toutes les classes. Dans notre cas, nous avons sélectionné 10cycle de type 'V' et 10cycle de type 'N'pour chaque enregistrement, afin d'avoir une représentation équilibrée en nombre de cas par classe. Donc nous avons constitué une base d'apprentissage qui contient 240 battements. (120 de type N&120de type V).

Pour la base de test nous avons choisi tous les battements restants de la base d'apprentissage.

IV.3.2 Sélection des descripteurs d'un cycle cardiaque

Puisque on s'intéresse aux pathologies ventriculaires et plus précisément l'extrasystole ventriculaire (ESV), les paramètres essentiels avec lesquels on reconnait ces

pathologies sont retenus. .ils sont comparables aux paramètres sur lesquels sont construites les connaissances médicales.

- **Intervalle RRp**

Nous appelons RRp la distance entre le pic R du présent battement et le pic R du battement précédent.

- **Intervalle RRs :**

C'est la distance entre le pic R du battement présent et le pic R du battement suivant.

- **Rapport des intervalles RR (RRs/RRp)**

C'est un paramètre qui caractérise une classe donnée. Dans le cas d'un rythme régulier, ce rapport est au voisinage de 1, mais il peut largement dépasser cette valeur dans le cas d'un 'BVP' avec repos compensatoire.

- **Largeur du complexe QRS(LQRS)**

Ce paramètre est une grande importance pour l'identification des battements 'BVP', ces types d'arythmies sont caractérisés généralement par un large complexe QRS.

IV.3.3 Architecture des classifieurs neuronaux

IV.3.3.1 Algorithmes d'apprentissage

Pour l'apprentissage, ils existent généralement trois protocoles utilisés par les réseaux multicouches :

Apprentissage stochastique, hors-ligne,

Apprentissage par lot,

Apprentissage en ligne.

L'apprentissage en ligne est utile lorsque l'ensemble d'apprentissage est très grand ou que la mémoire devient prohibitive, pour la sauvegarde des données. Aujourd'hui, les ordinateurs ne présentent plus de problèmes de limitation en mémoire et en coût. Dans l'apprentissage en ligne, chacune des formes est présentée une seule fois, ce qui donne un apprentissage rapide mais nécessite plus de données d'apprentissage.

L'apprentissage par lot est très utilisé de nos jours. Sous Matlab, on retrouve les trois protocoles d'apprentissage mais nous utiliserons seulement le protocole d'apprentissage par lot. Au cours de cette étude nous avons comparé principalement deux méthodes : dans la 1ere méthode nous avons utilisé un algorithme classique en optimisation locale non linéaire que nous avons décrit au chapitre 2 : ***la méthode de Levenberg-Marquardt*** .pour la 2eme méthode nous utilisons sur les mêmes classifieurs une technique d'optimisation globale issue de l'intelligence artificielle que nous avons décrite au chapitre 3 : ***algorithmes génétiques***

IV.3.3.2 Dimensionnement du réseau pour chaque classifieur

Dans cette étude la structure neuronale est déterminée par le nombre de neurones de couche d'entrée Ne, et celui de la couche cachée NC.

1. **Couche d'entrée** : le nombre de neurones dans la couche d'entrée est lié directement au vecteur d'entrée.

Pour le premier groupe que nous appelons *(GP1)* nous réalisons cinq classifieurs que nous appelons *CLS 1.1, CLS 1.2, CLS1. 3, CLS1.4, CLS1.5* le vecteur d'entrée est représenté par 4 descripteurs décrits précédemment *(RRp, RRS, RRs/RRp, LQRS).*

Pour le deuxième groupe que nous appelons *(GP2)* nous réalisons deux classifieurs que nous appelons *(CLS2.1, CLS2.2).* Nous (*RRs/RRp), donc le vecteur d'entrée est représenté par 3 descripteurs (RRp, RRS, LQRS).*

Pour le troisième groupe que nous appelons *(GP3)* nous réalisons deux classifieurs que nous appelons (CLS3.1, CLS3.1) nous utilisons les 2descripteurs (RRs/RRp, LQRS).

Pour le quatrième groupe que nous appelons *(GP4)* nous réalisons deux classifieurs que nous appelons (CLS4.1, CLS4.2), nous utilisons une seule entrée LQRS.

2. **Couche cachée** : le choix de neurones pour la couche cachée reste toujours un problème, beaucoup de chercheurs qui travaillent sur cette problématique (voir chapitre2). Dans notre étude nous appliquons au premier classifieur de chaque groupe la règle de pyramide présenté dans chapitre2telque N=$\sqrt{m \cdot n}$

N : Nombre de neurone à la couche cachée

m : Nombre de neurone à la couche d'entrée

n : Nombre de neurone à la couche de sortie

Puis pour connaitre l'influence de nombre de neurone de la couche cachée sur l'apprentissage, nous avons varie le nombre de neurones cachés dans chaque groupe de classifieurs

Les fonctions d'activation sont des fonctions de type sigmoïde, le tableau 4.2 représente respectivement les valeurs des paramètres pour chaque classifieur.

3. **La couche de sortie :** la sortie est composé d'un seul neurone correspondant aux deux classes utilisées ('N' et 'V').la fonction d'activation est une fonction linéaire (purelin) qui assure les sorties suivantes.

N=0, V=1

GP	CLS	Ne	NC
GP1	CLS1.1	4	2
	CLS1.2	4	3
	CLS1.3	4	4
	CLS1.4	4	5
	CLS1.5	4	1
GP2	CLS2.1	3	2
	CLS2.2	3	1
GP3	CLS3.1	2	2
	CLS3.2	2	1
GP4	CLS4.1	1	1
	CLS4.2	1	2

Tableau4.2 : le nombre de neurone de différents classifieurs

GP : groupe, **CLS** : classifieur, **Ne** : neurones d'entrées, **Nc** : neurones cachés

IV.3.4 apprentissage neuronal

Dans cette partie, nous appliquons sur les classifieurs présentés précédemment un apprentissage classique, en utilisant la règle de la rétro propagation par la méthode de *Levenberg-Marquardt* .

Puisque les algorithmes génétiques sont des algorithmes probabilistes on ne peut pas faire une comparaison avec un seul essai .pour cette raison, nous appliquons sur chaque classifieurs 10 phases d' apprentissage, avec des poids initiaux différents choisis

aléatoirement, puis nous calculons les performances suivantes pour chaque apprentissage : taux de classification correcte $\tau_{class,}$ la sensibilité *Se*, et spécifité *Sp* déjà définis dans chapitre1.ainsi par l'erreur minimale e attient par le réseau et le temps d'apprentissage *T* : .la comparaison est faite sur les moyennes de ces performances.

- **groupe1(*GP1*)**

Pour les classifieurs de groupe1, le vecteur d'entrée est représenté par 4 descripteurs (RRp, RRs, RRs/RRp et LQRS)

GP1 a été utilisé comme modèle de référence, pour connaitre l'influence de nombre de caractéristiques ou de descripteurs présentés au vecteur d'entrée sur la performance de réseau par les deux méthodes d'apprentissage.

Dans Le premier classifieur CLS1.1du groupe 1, le nombre de neurones cachés est choisi par la méthode de pyramide

$$Nc= \sqrt{Ne*Ns}$$

$$Nc= \sqrt{4*1} =2$$

Puis nous avons mené plusieurs experimentations en changeant chaque fois le nombre de neurones cachés (voir tableau 4.3)

Classifieurs	*Nc*	*e(moy)*	*T (moy)(s)*	*SE(moy)(%)*	*SP(moy)(%)*	$\tau_{class(moy)(\%)}$
CLS1.1	*2*	*0.045*	*10.29*	*97.39*	*95.54*	***95.78***
CLS1.2	*3*	*0.041*	*13.44*	*94.96*	*96.61*	***96.40***
CLS1.3	*4*	*0.033*	*15.21*	*97.79*	*96.45*	***96.63***
CLS1.4	*5*	*0.034*	*16.29*	*98.05*	*96.75*	***96.92***
CLS1.5	*1*	*0.083*	*8.44*	*88.57*	*87.45*	***87.59***

tableau4.3 :performances des classifiueurs deGP1(apprentrissage classique)

- **Groupe2(*GP2*)**

Pour les classifieurs de 2eme groupe nous présentons moins de descripteurs dans le vecteur d'entrée par rapport aux classifieurs de GP*1*. (Nous éliminons (*RRs/RRp*))

Dans le premier classifieur de ce groupe CLS2.1 le nombre de neurones cachés est choisi par la méthode de pyramide

$$Nc = \sqrt{3*1} \approx 2$$

Puis nous avons gardé seulement un neurone dans la deuxieme expirimentation (voir tableau 4.4)

Classifieurs	Nc	e(moy)	T (moy)(s)	SE(moy)(%)	SP(moy)(%)	$\tau_{class(moy)(\%)}$
CLS2.1	2	0.041	12.93	98.24	96.42	**96.65**
CLS1.2	1	0.062	11.65	98.79	86.56	**88.13**

tableau4.4 :performances des classifiueurs deGP2(apprentrissage classique)

a partir des ces expérimentations, nous remarquons les points suivants:

- La suppression de rapport (RRs/RRp) n'a pas influé globalement sur les résultats.
- La variation de Nc (neurones cachés) dans les différents classifieurs influe sur leurs performances finales.

3eme groupe (GP3)

Nous avons utilisé pour les classifieurs du groupe GP3 deux caractéristiques sur le vecteur d'entrée (LQRS,RRs/RRp).

Classifieurs	Nc	e(moy)	T (moy)(s)	SE(moy)(%)	SP(moy)(%)	$\tau_{class(moy)(\%)}$
CLS3.1	2	0.061	10.65	88.13	96.97	**95.84**
CLS3.2	1	0.12	9.73	88.76	67.98	**70.64**

tableau4.5 :performances des classifiueurs deGP3(apprentrissage classique)

Nous remarquons que :

- La suppression de RRp et RRs et la diminution de neurones cachés influe sur la performance des classifieurs.

4eme groupe (GP4)

Pour les classifieurs de ce groupe, nous utilisons une seule entrée, (LQRS).

Classifieurs	Nc	e(moy)	T (moy)(s)	SE(moy)(%)	SP(moy)(%)	$\tau_{class(moy)(\%)}$
CLS4.1	1	0.20	4.74	68.36	76.77	**75.70**
CLS4.2	2	8.55	78.12	74.87	75.29	**8.55**

tableau4.5 :performances des classifiueurs deGP4(apprentrissage classique)

Le manque de caractéristiques dans le vecteur d'entrée influe sur la performance globale des classifieurs

conclusion

Dans cette partie de notre étude, nous avons réalisé plusieurs classifieurs neuronaux en utilisant un apprentissage classique (rétro propagation) à base de la descente de gradient, et nous avons conclu qu'un classifieur neuronal donne des bons résultats si on a une bonne structure (nombre de neurones cachés suffisants) et un nombre suffisant de caractéristiques informatives. Donc la structure réduite et le manque de caractéristiques influe sur la performance globale de réseau

IV.3.5 Apprentissage neuro-génétique

Dans cette partie, nous appliquons sur les mêmes classifieurs présentés précédemment, un apprentissage génétique en minimisant l'erreur quadratique entre la sortie réel et la sortie désirée du réseau, d'apprentissage, pour comparer les deux technique (classique et neuro-génétique)

IV.3.5.1 Choix des options

Pour lancer l'AG, il faut définir certains paramètres tels que : la taille de la population, les probabilités de mutation et de croisement et le nombre de générations. Il est difficile de les fixer ou de trouver les meilleurs avant l'exécution de l'algorithme. C'est un problème de réglage qui doit être optimisé pour chaque type de problème a traité.

Cela constitue une partie importante du travail de l'expérimentateur. Dans la littérature, la définition de ces paramètres diffère d'une application à une autre.

Donc la fixation de ces paramètres est une question d'essai, pour notre étude pour chaque valeur ou technique de ces paramètres, nous réalisons plusieurs phases d' apprentissage, puis nous calculons l'écart type les valeurs minimales et maximales, le taux de classification τ_{class} et l'erreur quadratique pour chaque technique ou valeur.

Les expérimentations sont réalisées sur le classifieur CLS3.1, mais on peut les générer sur les autre classifieur puisque ils ont les mêmes formes de la fonction à optimisé (fitness).

- **Choix de la taille de population**

Le choix de la taille de population dépond à la taille d'individu (le nombre de gène dans un chromosome). Dans notre cas les gènes sont des poids synaptiques c.à.d. la taille de population dépond à la structure de réseau.

Nous avons réalisé un programme qui fait plusieurs apprentissages de différente taille de population : 20, 40, 60, 80,100individus.

Ces expérimentations sont appliquées sur le classifieur CLS3.1.mais le choix de la taille diffère d'un classifieur à un autre (dépend de la structure), mais la fixation d'une taille pour ce classifieur nous donne un repère pour les autres.

taille	Std(*e*)	Min (*e*)	Max (*e*)	Std(τ_{class})	Min(τ_{class})	Max(τ_{class})	T(s)
20	0.09±0.061	0.05	0.25	93.25±5.69	80.01	97.14	35.48
40	0.06±0.018	0.049	0.10	95.83±3.18	86.9	97.64	71.26
60	0.05±0.002	0.065	0.047	96.68±0.21	96.33	97.05	105.79
80	0.05±0.009	0.07	0.046	96.33±0.99	93.89	97.4	140.50
100	0.048±0.0048	0.060	0.047	96.61±0.11	96.34	96.84	173.78

Tableau4.6 : performances d'un classifieur neuro-génétique avec différentes tailles de population

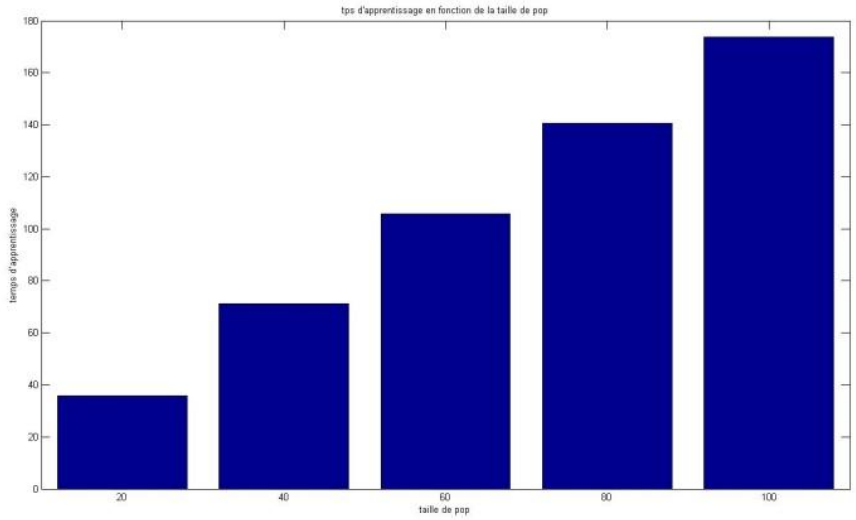


Figure 4.5 : Temps d'apprentissage pour différentes tailles de populations

Nous remarquons bien qu'une population trop petite (ex.20 individus) évolue probablement vers un optimum local peu intéressant ($e_{max}=0.25, \tau_{class}=80\%$ Std(τ_{class})= 93.25±5.69),. Une population trop grande (ex.100) met plus de temps pour converger (T100=173.78setT20=v35.48s) vers des solutions envisageables ($e_{max}=0.047, \tau_{class}=96.84\%$ Std(τ_{class})= 96.61±0.11),.. La taille de la population doit être choisie de façon à réaliser un bon compromis entre le temps de calcul et la qualité du résultat. (ex. pour 60individus, le classifieur est performant ($e_{max}=0.047, \tau_{class}=96.33\%$ Std(τ_{class})= 96.68±0.21)et T=105s.

Donc le bon choixde la taille pour le classieur CLS3.1est 60 individus

Choix de la fonction scaling

La fonction Scaling ou mise a l'échelle permet de modifier la fonction fitness a fin de réduire ou d'amplifier artificiellement les écarts entre les individus.

Il existe plusieurs techniques de scaling. Nous utilisons quatre types

- Fonction Rank
- Fonction proportionnelle
- Fonction top
- Fonction linéaire.

Ces 4 techniques sont présentées dans chapitre3.

taille	Std(*e*)	Min (*e*)	Max (*e*	Std(τ_{class})	Min(τ_{class})	Max(τ_{class})
Rank	0.05±0.007	0.049	0.071	96.50±0.54	95.31	97.29
Prop	0.07±0.025	0.056	0.14	95.05±3.79	85.46	97.93
Top	0.09±0.029	0.058	0.14	95.20±1.61	93.02	97.56
Linaire	0.060±0.008	0.0522	0.07	95.38±0.69	94.54	96.38

Tableau4.7 : performances d'un classifieur neuro-génétique avec différentes techniques de scaling

Ces résultats montrent bien que le classifieur neuro-génétique doit être plus performant pour les fonctions scaling de type « Rank » et « linéaire ». Nous avons par exemple Pour la fonctio,« Prop », le classifieur peut avoir un taux de classification égale seulement à 85% et pour la fonction « Top » un taux égal à 93%.par contre pour les fonctions « Rank » et « linaire » comme une valeur min de τ_{class}≈95%.

Même pour l'erreur (e), le classifieur pour les fonctions « top » et « proportionnelle » peut atteindre seulement 0.14, par contre pour rank et liniar la valeur max =0.07.

D’après ces résultats la fonction « rank » et la méilleur techenique pour notre cas , par contre la fonction « top » à donné des mauvasies résultats

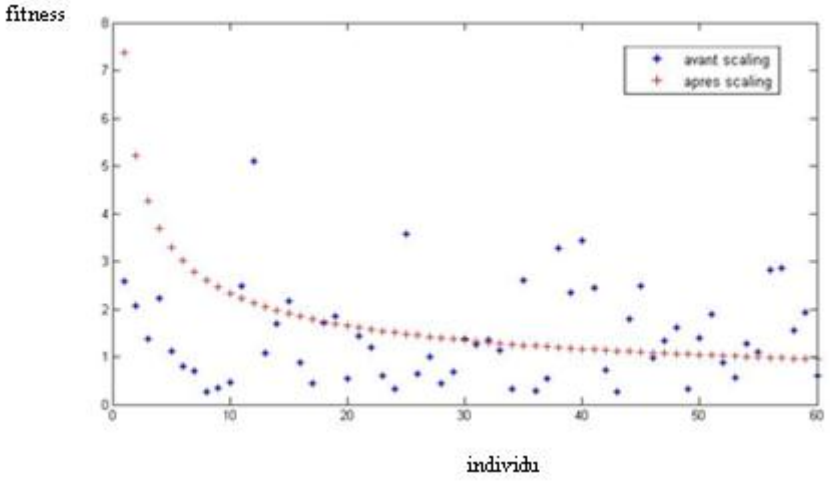


Figure4.6 : Les valeurs de fitness avant et après la fonction scaling « rank »

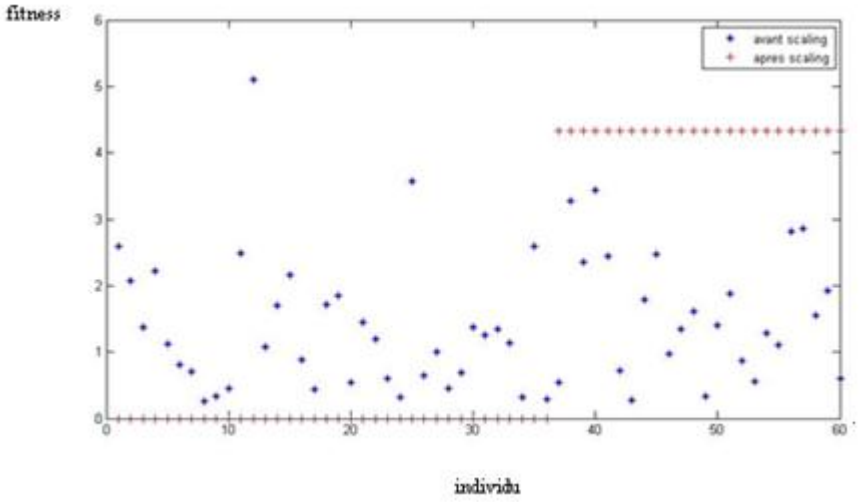


Figure4.7 : Les valeurs de fitness avant et après la fonction scaling « TOP »

Les figures 4.6 et 4.7 présentent les valeurs de fitness de chaque individu de pop après et avant scaling en utilisant les fonctions « top » et« rank »

Les figures 4.6 et 4.7 expliquent bien les résultats non satisfaisants de « top », nous remarquons que la fonction top donne a 40% d’individus la même valeur ce qui influe sur la diversité de population. Ainsi la fonction « top » donne aux reste la valeur « 0 », donc 60% d’individus ils n’ont pas de la chance d’être sélectionné .par contre le rank, Chaque individus se voit associé un rang en fonction de sa position. Le plus mauvais chromosome aura le rang 1, le suivant 2, et ainsi de suite jusqu'au meilleur chromosome qui aura le rang N (pour une population de N chromosomes). Avec cette méthode, tous les chromosomes ont une chance d'être sélectionnés.

Les figures suivantes présentes la diversité (la distance moyenne entre les individus) de population pendant l'évolution de l'algorithme génétique en utilisant « rank » et « top ».

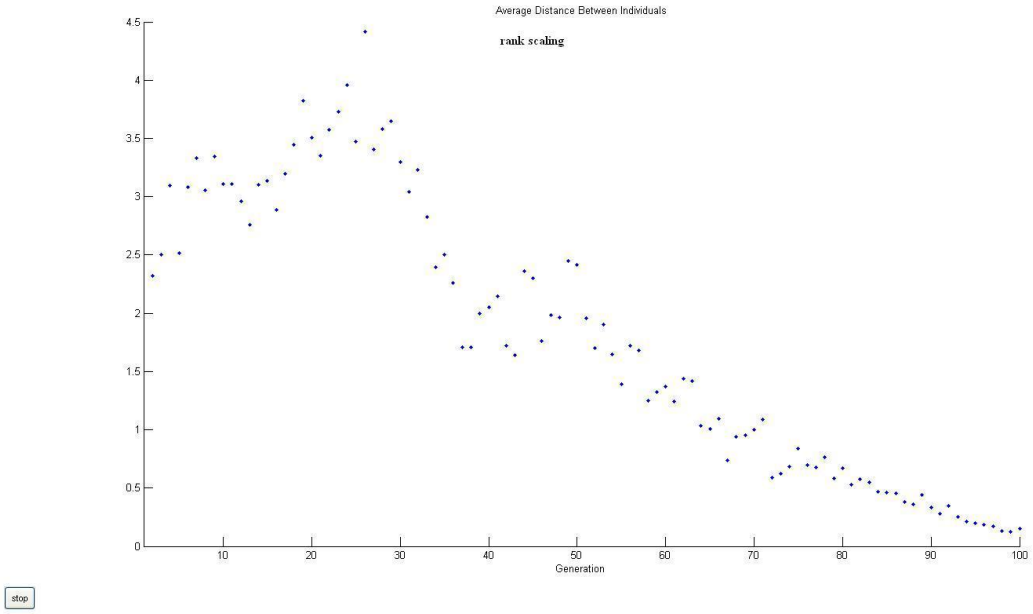


Figure4.8 : **Distance entre les individus crée par scaling « rank »**

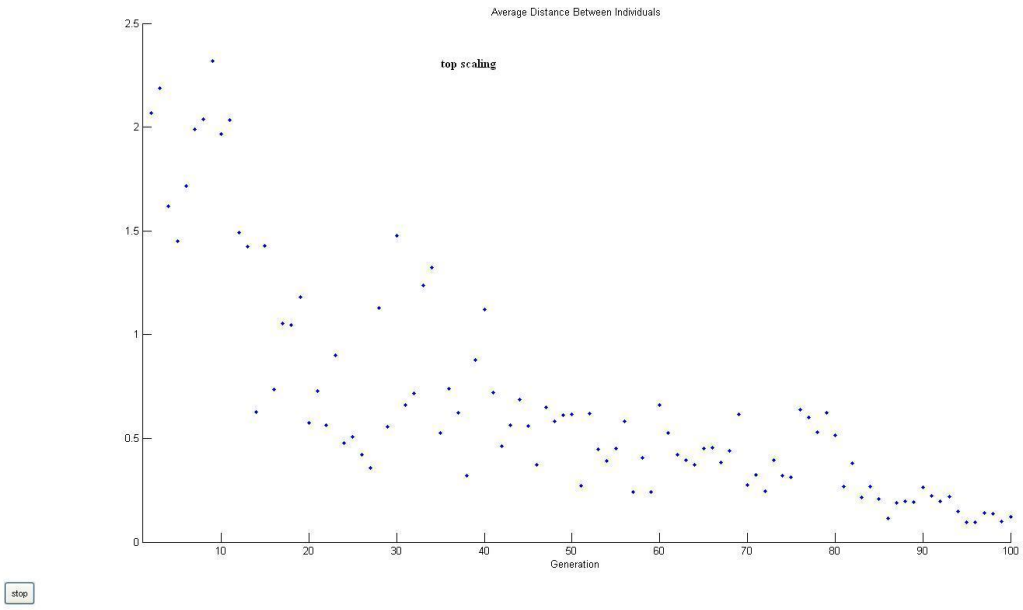


Figure4.9 : Distance entre les individus crée par scaling « top »

La fonction Rank atteint une distance jusqu'à 4.5 par contre la fonction top moins de2.5.

La meilleure technique de scaling pour notre étude est la technique « Rank ».

Choix de sélection

La sélection est le choix des parents pour la génération suivante en basant sur les valeurs de fitness scaling. Un individu peut être choisit plusieurs fois, par contre un autre peut être supprimé. (Voir chapitre3)

Dans notre étude, nous choisissons entre les 5 techniques de sélection : « stochastique » », « remainder », « uniforme » »roulette », et « tournoi ».

selection	Std(*e*)	Min (*e*)	Max (*e*)	Std(τ_{class})	Min(τ_{class})	Max(τ_{class})
stoch	0.061±0.007	0.050	0.071	96.74±0.54	95.49	97.5
remainder	0.058±0.083	0.043	0.070	96.75±0.38	96.33	97.62
uinfor	0.078±0.016	0.065	0.10	94.07±1.67	93.13	95.95
roulette	0.058±0.006	0.048	0.068	96.59±0.50	95.73	97.17
tournoie	0.075±0.021	0.060	0.12	93.45±3.58	89.50	97.24

Tableau4.8 : performancse d'un classifieur neuro-génétique avec différents techniques de sélection

Nous remarquons que les techniques « stochastique », « remainder » et « tournoi », donnent des meilleurs résultats par rapport au tournoi et proportionnelle. par exemple Std(τ_{class}) de remainder égale à 96.59±0.50 (*τclass meilleur et un $\Delta\tau$class très faible)*, donc avec remainder on peut assurer des bon résultats.par contre la technique tournoi Std(τ_{class}) =93.45±3.58 (*τclass insuffisant et un $\Delta\tau$class important)*.

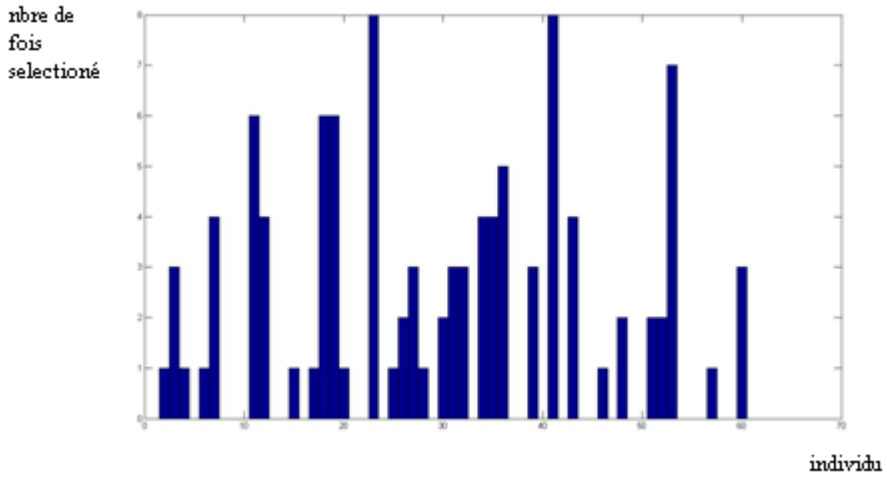


Figure4.10 : Sélection par tournoie

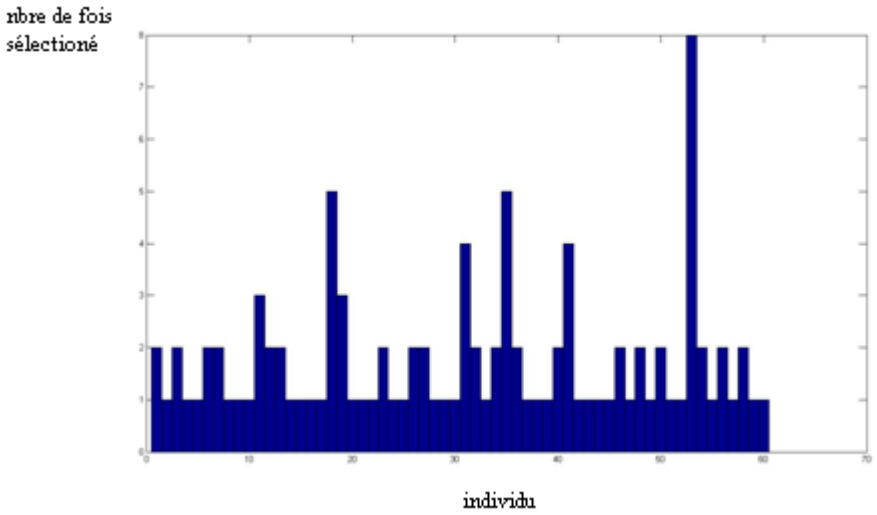


Figure4.11 : **Sélection par remainder**

Ces histogrammes présentent le nombre de fois de sélection de chaque individu de la première génération en utilisant les deux techenique « remainder » et « tournoie ».

Ces histogrammes expliquent bien, les résultats satisfaisants de « remainder ». Avec « remainder » tous les individus sont sélectionné aux moins une fois, donc il n'ya pas de risque que le bon individu soit éliminé. Par contre pour «tournoi » plusieurs individus soient éliminés, et plusieurs individus sont sélectionnés plusieurs fois ce qui influe sur la diversité de population.

Nous avons choisi pour notre application la technique « remainder »

- **Choix de taux de croisement**

Taux de croissement est le taux d'individus qui participe à l'opérateur de croisement, pour que le reste participe à la mutation. même expérimentation que les autre paramètre, nous avons réalisé plusieurs apprentissages (10) de différente taux : 0, 0.2, 0.4, 0.6, 0.8, 1. puis nous calculons Std , min et max pour ***e*** **et** ***πclass*** pour chaque taux.

Taux	Std(*e*)	Min (*e*)	Max (*e*)	Std(τ_{class})	Min(τ_{class})	Max(τ_{class})
0	0.079±0.06	0.048	0.25	87.50±26.27	12.83	97.42
0.2	0.058±0.08	0.044	0.07	96.45±1.07	93.73	97.38
0.4	0.06±0.01	0.044	0.097	96.63±0.59	95.57	97.8
0.6	0.058±0.004	0.05	0.066	96.57±0.653	94.57	97.43
0.8	0.05±0.0057	0.046	0.062	96.76±0.32	97.08	97.32
1	0.23±0.02	0.18	0.25	65.83±33.28	12.83	96.32

Tableau4.9 : performances d'un classifieur neuro-génétique avec différents taux de croisement

Nous remarquons que Le classifieur EST performant pour les cas suivants

- Taux=0.4
- Taux= 0.6
- Taux= 0.8

Les meilleurs résultats sont obtenue pour un taux égale à 0.8. Même dans la littérature plusieurs chercheurs ont trouvé que les meilleurs taux sont entre [0.6, 0.95].

Pour un taux égale à 0(pas de croissement, tous les enfants sont des enfants de mutation), le classifieur peut avoir un τ_{class}=12.83% et e=0.25. La même chose pour un taux égale à1 (pas de mutation toutes les enfants sont des enfants de croisement) std(τ_{class})= 65.83±33.28.ces résultats montrent bien l'importance de mutation et croisement dans les algorithmes génétiques.

Nous vons fixé pour notre cas un taux à 0.8

- **Choix de meilleures techniques de croisement**

Classiquement le croisement est la recombinaison entre deux parents pour avoir deux enfants pour la génération suivantes. Le but de croisement est d'enrichir la diversité de la population.

Dans notre cas, nous utilisons 6 techniques de croisement : « uniforme », « croisement à 1 point », « croisement à 2 points », « croisement étendu », « croisement heuristique » et « croisement arithmétique ».

Croisement	Std(*e*)	Min (*e*)	Max (*e*)	Std(τ_{class})	Min(τ_{class})	Max(τ_{class})
uniforme	0.074±0.052	0.046	0.22	90.11±21.31	29.46	97.79
Un point	0.066±0.011	0.052	0.081	95.44±0.97	93.61	96.69
2points	0.061±0.076	0.047	0.074	95.88±1.32	93.4	97.39
intermed	0.057±0.0068	0.048	0.067	96.69±0.26	96.10	97.12
heuristique	0.063±0.016	0.051	0.10	96.60±0.33	95.92	97.08
arithmétique	0.10±0.059	0.059	0.25	87.43±26.22	12.83	96.95

Tableau4.10 : performances d'un classifieur neuro-génétique avec différents techniques de croisement

Pour notre cas, la seule technique qui a donné des résultats satisfaisants est « inter médiate ».

Choix de la meilleure technique de mutation

La mutation consiste à changer ou permuter les gènes de chromosome parent, pour avoir un chromosome enfant

Dans notre étude, nous utilisons trois techniques « gaussien », « uniforme » et « adaptatif ».

Avant l'utilisation de la mutation gaussien, nous avons réglé deux paramètres « scale » et « shrink » (voir chapitre3 »

La fixation de ces paramètres est faite après plusieurs essais. Nous avons réalisé des expérimentations pour fixer la meilleur valeur de « scale » et « shrink »

Le paramètre scale a été choisi entre [0,10]

scale	*Std(e)*	*Min (e)*	*Max (e*	*Std(τ_{class})*	*Min(τ_{class})*	*Max(τ_{class})*
0	*0.21±0.029*	*0.163*	*0.249*	*73.77±30.65*	*13.59*	*96.37*
1	*0.059±0.008*	*0.049*	*0.076*	*96.84±0.49*	*96.1*	*97.87*
2	*0.054±0.007*	*0.043*	*0.067*	*96.61±0.33*	*96.33*	*96.45*
3	*0.051±0.056*	*0.043*	*0.064*	*96.45±0.38*	*96.11*	*96.91*
4	*0.053±0.084*	*0.041*	*0.064*	*95.69±1.153*	*93.73*	*96.4*
5	*0.059±0.017*	*0.042*	*0.095*	*95.50±1.34*	*93.42*	*96.92*

6	*0.053±0.011*	*0.041*	*0.074*	*96.60±0.56*	*96.10*	*97.17*
7	*0.056±0.018*	*0.042*	*0.097*	*95.07±3.07*	*86.89*	*97.1*
8	*0.055±0.011*	*0.043*	*0.078*	*95.81±1.37*	*93.29*	*97.02*
9	*0.069±0.021*	*0.043*	*0.11*	*96.20±1.19*	*93.39*	*97.3*
10	*0.053±0.012*	*0.043*	*0.081*	*95.47±1.27*	*93.59*	*97.75*

Tableau4.11 : performances d'un classifieur neuro-génétique avec différents valeurs de scale

Pour fixer le shrink, nous avons utilisé les valeurs suivantes : [-1,-0.5, 0,0.5, 1, 2,3]

shrink	Std(*e*)	Min (*e*)	Max (*e*)	Std(τ_{class})	Min(τ_{class})	Max(τ_{class})
-1	0.071±0.015	0.045	0.092	95.74±2.44	90.02	97.76
-0.5	0.071±0.019	0.041	0.097	95.59±1.63	92.33	97.2
0	0.091±0.036	0.048	0.1553	91.00±11.27	61.87	97.39
0.5	0.071±0.023	0.045	0.12	96.47±1.02	94.19	97.89
1	0.056±0.011	0.042	0.071	96.56±0.63	95.34	97.37
2	0.061±0.019	0.040	0.10	95.39±2.68	88.2	97.14
3	0.069±0.025	0.045	0.13	94.36±4.21	83.22	97.14

Tableau4.12 : performances d'un classifieur neuro-génétique avec différents valeurs de shrink

La meilleure valeur pour le paramètre shrink est 1, pour les autres valeurs le réseau est moins performant.

Nous remarquons, pour le paramètre scale égale 0 ou le paramètre shrink égale 0, les résultats sont vraiment insuffisants, ceci montre l'importance de shrink et scale pour l'ajustement de bruit gaussien.

Mutation	Std(*e*)	Min (*e*)	Max (*e*)	Std(τ_{class})	Min(τ_{class})	Max(τ_{class})
gaussien	0.06±0.008	0.045	0.072	96.91±0.41	96.26	97.65
uniforme	0.21±0.024	0.17	0.24	77.43±22.41	30	96.27
adaptatif	0.054±0.075	0.042	0.066	96.46±0.81	95.06	97.55

Tableau4.13 : performances d'un classifieurs neuro-génétique avec différents techniques de mutation

La mutation gaussienne et adaptative donnent des meilleurs résultats.

La technique choisi pour notre étude est gaussien avec scale=1et shrink=1.

Conclusion

Les paramètres, les plus performants pour les algorithmes génétiques changent d'un problème à un autre, il y a pas une règle qui permrt de fixer ces paramètres avant l'exécution de l'algorithme.

Dans notre étude après les expérimentations précédentes les paramètres choisis sont présentés dans le tableau suivant :

Paramètres	Technique ou valeur fixé
scaling	rank
sélection	ramender
Taux de croisement	0.8
Croissement	inter médiate
mutation	Gaussien
scale	1
shrink	1

Tableau4.14: différents paramètres choisis pour l'apprentissage génétiques

IV.3.5.2 Réalisation des classifieurs neuro-génetique en minimisant l'erreur quadratique

Dans cette partie, nous avons hybridé les algorithmes génétiques avec les réseaux de neurones pour optimiser (minimiser) l'erreur quadratique entre la sortie réelle et la sorti désirés de réseau c.à.d. remplacer l'apprentissage classique avec un apprentissage génétique.

Nous appliquons sur les même classifieurs réalisés précédemment un apprentissage génétique. Nous utilisons les paramètres fixés précédemment. Certain change d'un classifieur à un autre, (les paramètres qui changent avec la taille de chromosome (individu)).ces paramètres sont présentés dans le tableau suivant :

GP	CLS	Taille de chromosome	Taille de pop	Nbre de géneration
GP1	CLS1.1	13	80	100
	CLS1.2	19	100	100
	CLS1.3	25	150	200
	CLS1.4	31	200	300
	CLS1.5	7	60	100
GP2	CLS2.1	11	70	100
	CLS2.2	6	50	80
GP3	CLS3.1	9	60	100
	CLS3.2	5	40	80
GP4	CLS4.1	4	40	80
	CLS4.2	7	60	100

Tableau4.15 : paramètres fixés pour différents classifieurs

- **1er Groupe (GP1) :** le vecteur d'entrée est représenté par les 4 descripteurs (RRp, RRs, RRS/RRp, LQRS).

lassifieurs	*Nc*	*e(moy)*	*T (moy)(s)*	*SE(moy)(%)*	*SP(moy)(%)*	$\tau_{class(moy)(\%)}$
CLS1.1	*2*	0.0472	*165.3*	98.44	96.62	**96.85**
CLS1.2	*3*	0.048	*193.5*	98.51	96.07	**96.38**
CLS1.3	*4*	0.0472	*216.54*	98.48	96.87	**97.07**
CLS1.4	*5*	0.047	*232.56*	98.46	96.74	**96.96**
CLS1.5	*1*	0.052	*166.15*	97.72	96.46	**96.65**

tableau4.16 :performances des classifiueurs deGP1(apprentrissage génétique)

Nous ajoustons la structure (nombre de neurones cachés) ,pour ces 5 classifieurs nous remarquons que les performances restent similaires pour les différents neurones cachés.

- **2eme groupe(GP2)**

Le vecteur d'entrée représenté par 3 descripteurs (RRp, RRs, LQRS).

Classifieurs	*Nc*	*e(moy)*	*T (moy)(s)*	*SE(moy)(%)*	*SP(moy)(%)*	$\tau_{class(moy)(\%)}$
CLS2.1	*2*	0.054	*177.45*	97.65	96.06	**96.26**
CLS2.2	*1*	0.050	*150.46*	98.30	96.10	**96.38**

tableau4.17 :performances des classifiueurs deGP2(apprentrissage génétique)

Nous remarquons le point suivant :

- Pour les classifieurs de ce groupe, malgré qu'on a éliminé (RRs/RRp) (moins de caractéristiques par rapport aux classifieurs de premier ensemble) nous avons presque les mêmes résultats, donc le paramètre RRs/RRp n'ajoute pas grand en terme d'information.
- Malgré que CLS2.2 présente un nombre de neurones réduits, les résultats restent satisfaisants .
-

- ***3eme groupe (GP3) :*** le vecteur d'entrée est représenté par les deux descripteurs (RRp/RRs, LQRS).

Classifieurs	*Nc*	*e(moy)*	*T (moy)(s)*	*SE(moy)(%)*	*SP(moy)(%)*	$\tau_{class(moy)(\%)}$
CLS3.1	*2*	0.059	*158.32*	97.51	96.87	**96.95**
CLS3.2	*1*	0.054	*137.56*	97.78	96.69	**96.83**

tableau4.18 :performances des classifiueurs deGP3(apprentrissage génétique)

Nous remarquons qu'avec seulement deux descripteurs, les résultats restent satisfaisants, Même pour un nombre de neurones cachés réduits.

4eme groupe (*GP4*)

Le vecteur d'entrée est représenté seulement par LQRS.

Classifieurs	*Nc*	*e(moy)*	*T (moy)(s)*	*SE(moy)(%)*	*SP(moy)(%)*	$\tau_{class(moy)(\%)}$
CLS4.1	*1*	0.099	*125.46*	96.42	93.07	**93.50**
CLS4.2	*2*	0.096	*141.58*	96.72	92.72	**93.23**

tableau4.19 :performances des classifiueurs deGP4(apprentrissage génétique)

Nous remarquons que Le manque de caractéristiques influe légèrement sur les résultats des classifieurs, mais les résultats restent acceptables.

Conclusion

Dans cette partie de notre travail, nous avons réalisé des classifieurs neuro-génétiques en remplaçant l'apprentissage classique (descente de gradient) par un apprentissage génétique en optimisant l'erreur quadratique de réseau et après différentes expérimentations, nous remarquons que :

- La structure réduite (peu de neurones cachés) n'influe pas sur la performance des classifieurs neuro-génétiques
- La réduction des caractéristiques influe légèrement sur la performance des classifieurs

IV.3.5.3 Réalisation des classifieurs neuro-génetiques en minimisant le taux de classification non correcte (TCNC)

Dans cette partie utilisons un apprentissage neuro-génétique pour réduire le taux de classification non correcte(TCNC).

Nous utilisons la même base d'apprentissage et l'application sera sur les classifieurs suivant (CLS1.2, CLS1.5, CLS2.2, CLS3.1, CLS3.2et CLS4.1).

Classifieurs	*e(moy)*	*T (moy)(s)*	*SE(moy)(%)*	*SP(moy)(%)*	$\tau_{class(moy)(\%)}$
CLS1.2	0.065	*218.69*	98.42	96.95	**97.14**
CLS1.5	0.0672	*181.1*	98.43	96.73	**96.95**
CLS2.2	0.0678	*175.48*	98.45	96.60	**96.84**
CLS3.1	0.0458	*193.5*	97.42	96.75	**96.84**
CLS3.2	0.0606	*159.56*	97.69	96.51	**96.66**
CLS4.1	0.1967	*140.51*	97.82	92.05	**92.57**

Tableau4.20 :performances des classifiueurs neuro-génétiquesen minimisant TCNC

IV.4.5.4 Comparaison entre les deux méthodes :

Nous comparons dans cette partie les performances obtenus par les deux approches d'apprentissage génétique (minimisation de l'erreur quadratique et la minimisation de taux de classification non correcte)

La comparaison est faite selon les paramètres de performances comme : Se, Sp, τ_{class} et T

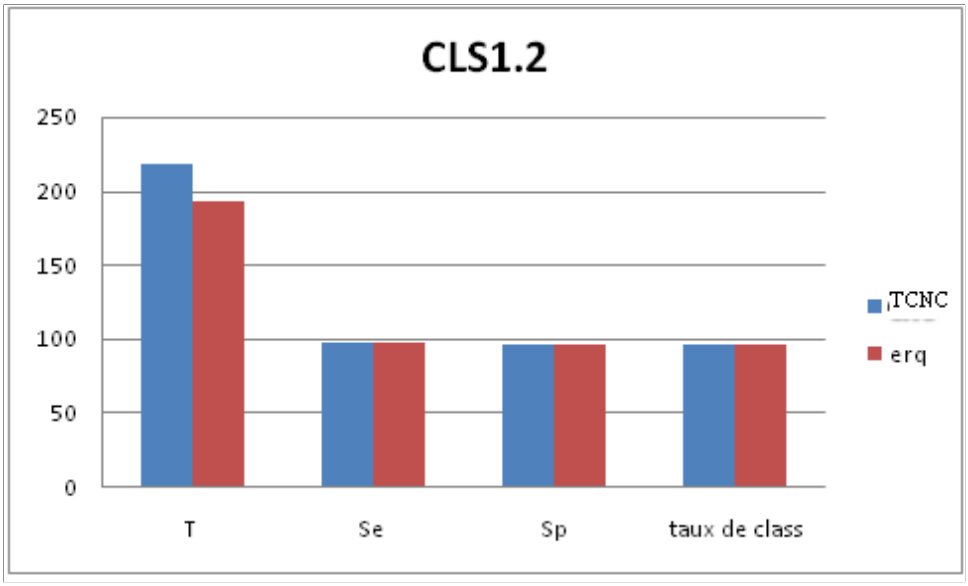


Figure4. 12 : Variation des paramètres de performance de classifieur CLS1.2 d'une technique a une autre

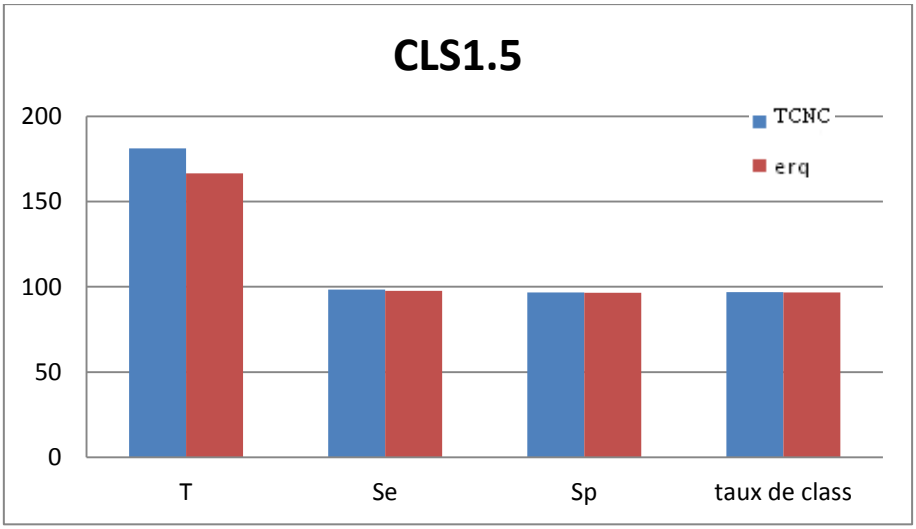


Figure1.13 : Variation de paramètre de performance de classifieur CLS1.5 d'une technique a une autre

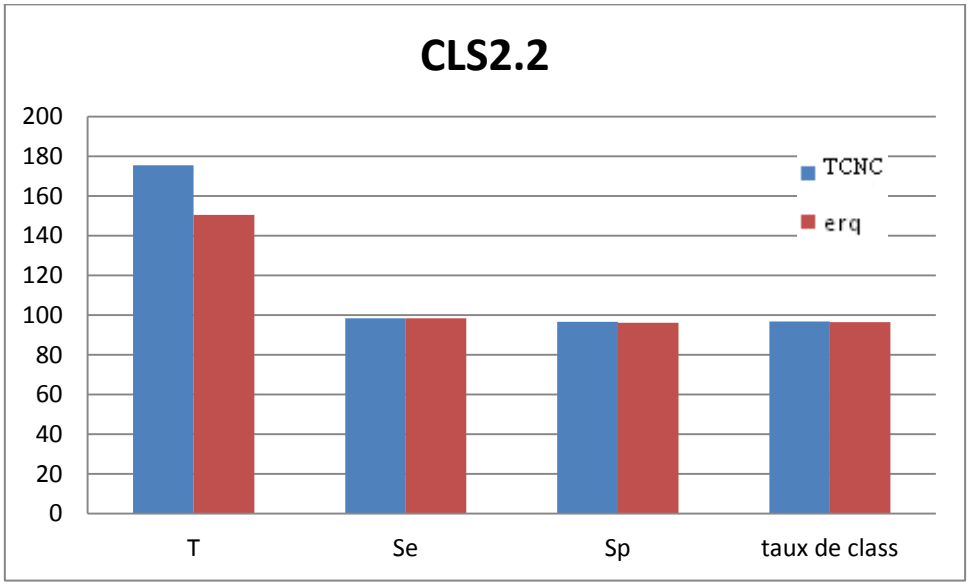


Figure4.14 : Variation des paramètres de performance de classifieur CLS2.2 d'une technique a une autre

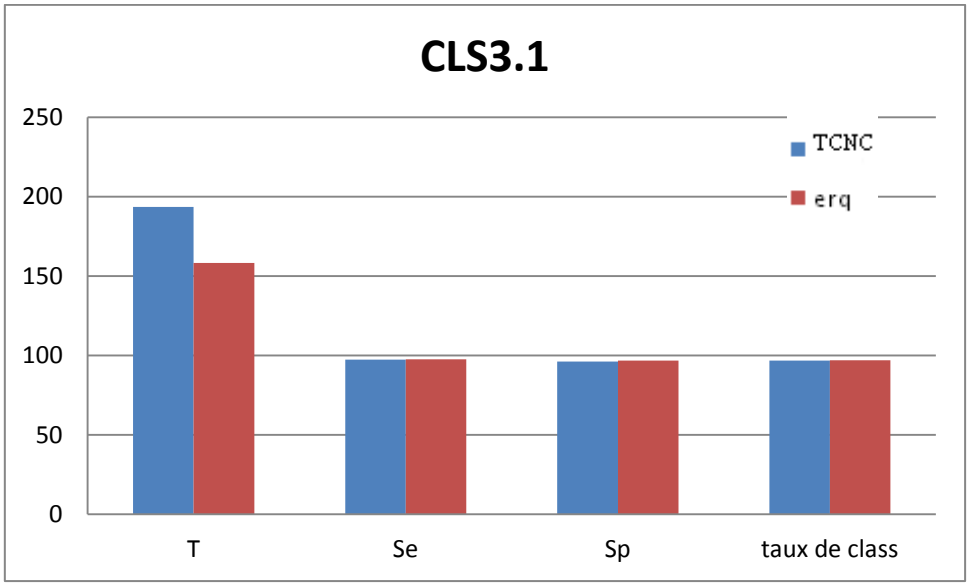


Figure4.15 : Variation des paramètres de performance de classifieur CLS3.1d'une technique a une autre

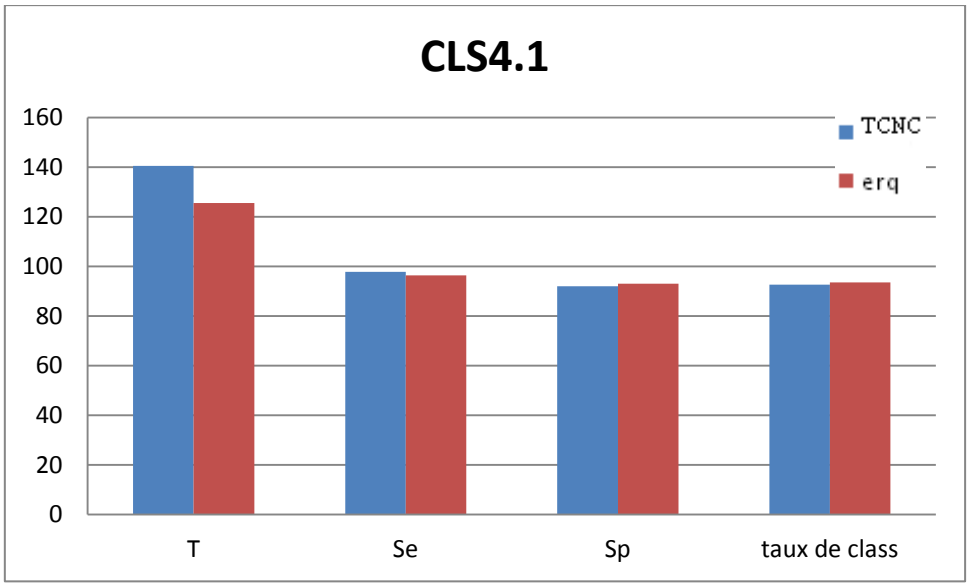


Figure4.16 :Variation des paramètres de performance de classifieur CLS4.1d'une technique a une autre

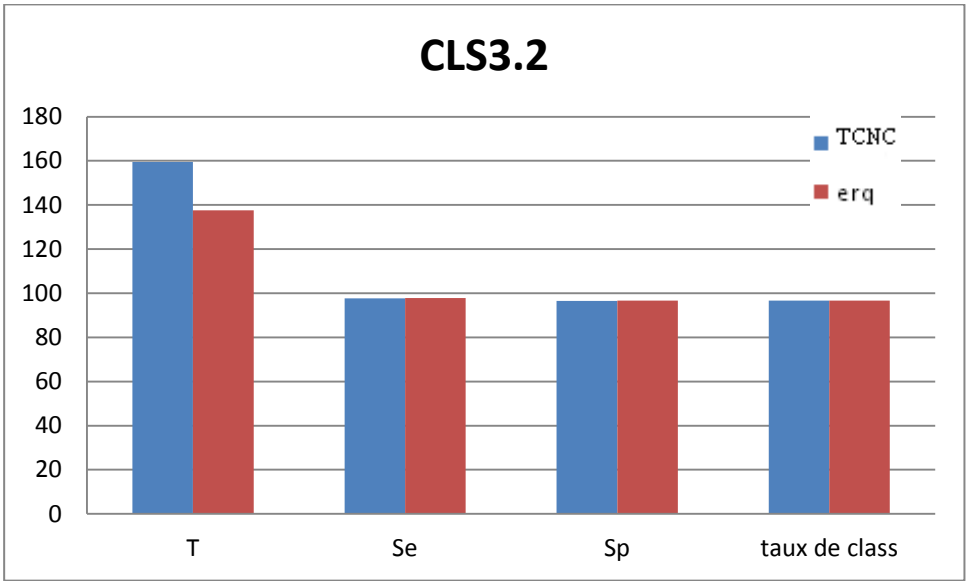


Figure4.17 : Variation des paramètres de performance de classifieur CLS3.2d'une techenique a une autre

Conclusion

Ces histogrammes comparent les deux approches (la minimisation de l'erreur quadratique de réseau et la minimisation de TCNC) par les algorithme génétique.

Nous remarquons que pour tous les classifieurs, les paramètres de performance Se, Sp et τ_{class} sont similaires pour les deux techniques. Et nous remarquons aussi que le temps de convergence de TCNC et plus important que le temps de convergence de l'erreur quadratique.

Les deux techniques donnent des résultats presque similaires, seulement la minimisation de l'erq à un avantage : c'est le temps de convergence qui est réduit par rapport à la minimisation de TCNC

IV.3.5.6 Comparaison entre l'apprentissage classique et l'apprentissage génétique

Dans cette partie, nous comparons l'apprentissage classique avec l'apprentissage génétique.

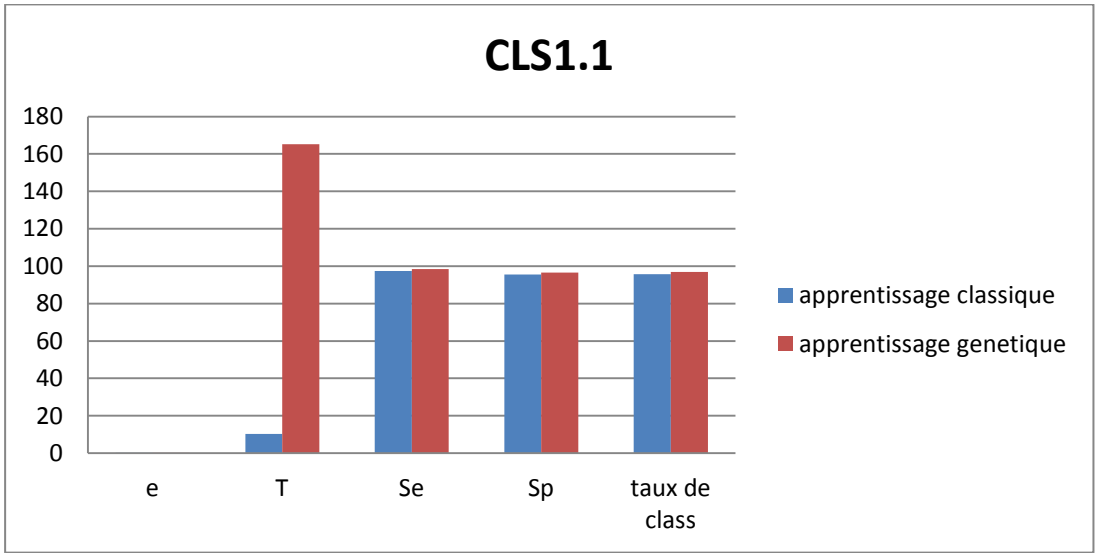


Figure4.18 : Variance des performances de CLS1.1 de l'apprentissage classique à l'apprentissage génétique

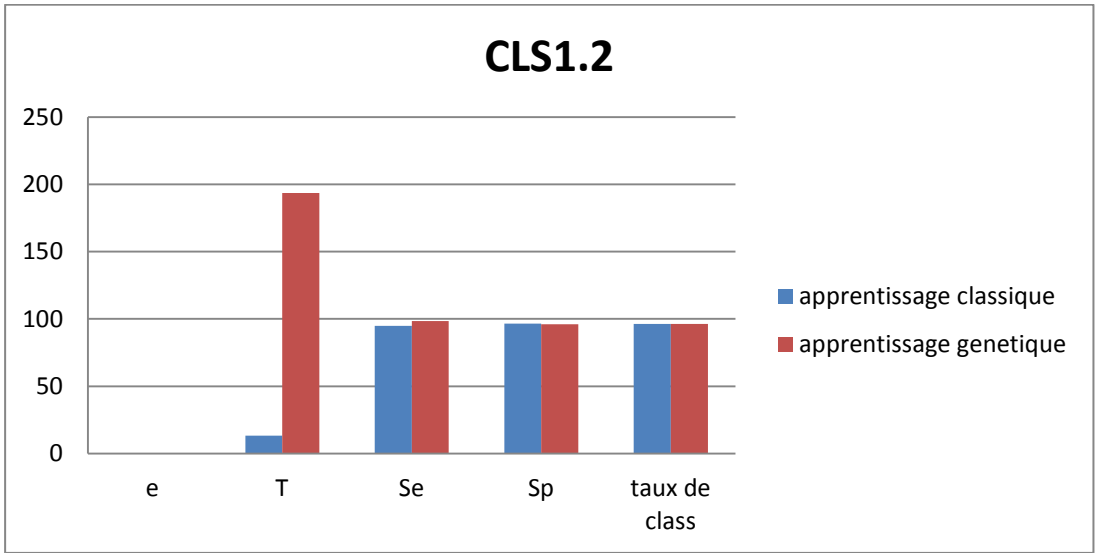


Figure4.19 : Variance des performances de CLS1.2 de l'apprentissage classique à l'apprentissage génétique

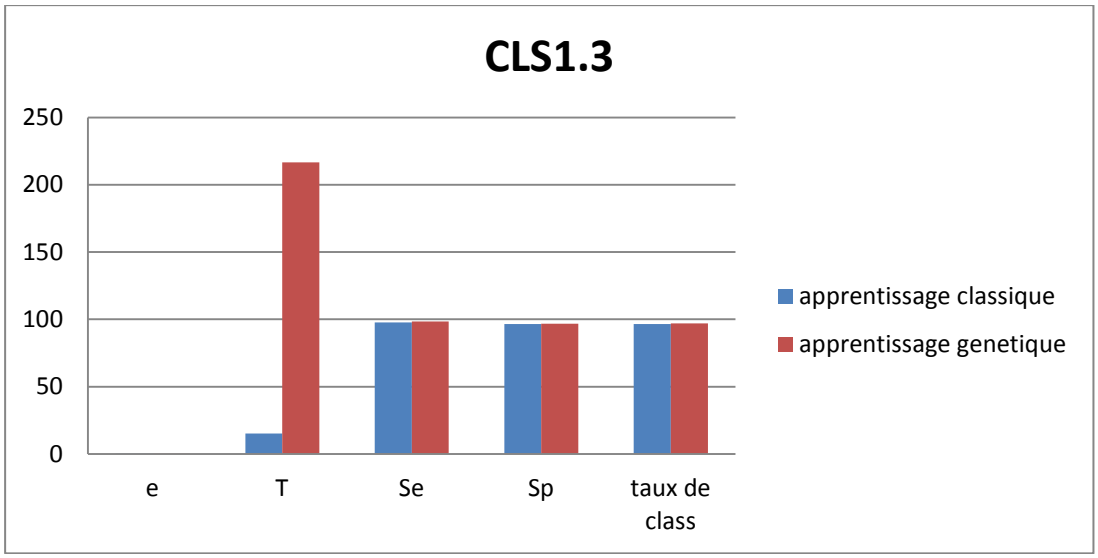


Figure4.20 : Variance des performances de CLS1.3 de l'apprentissage classique à l'apprentissage génétique

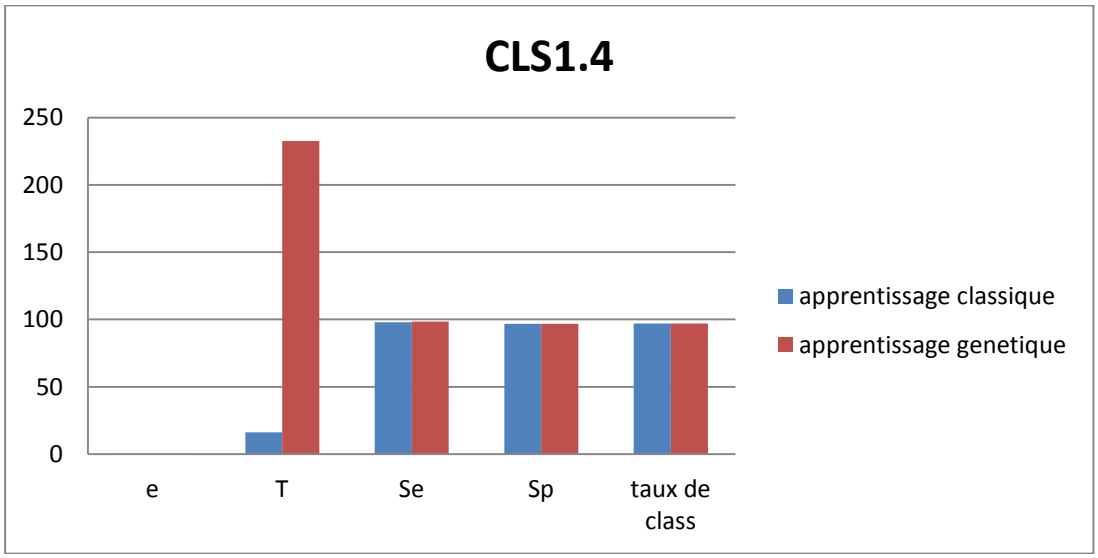


Figure4.21 : Variance des performances de CLS1.4 de l'apprentissage classique à l'apprentissage génétique

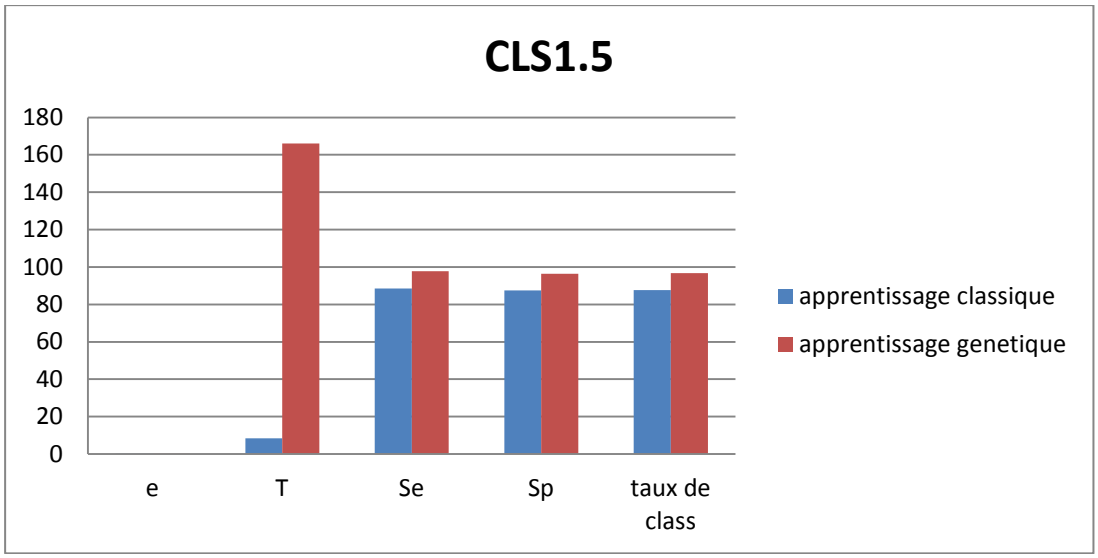


Figure4.22 : Variance des performances de CLS1.5 de l'apprentissage classique à l'apprentissage génétique

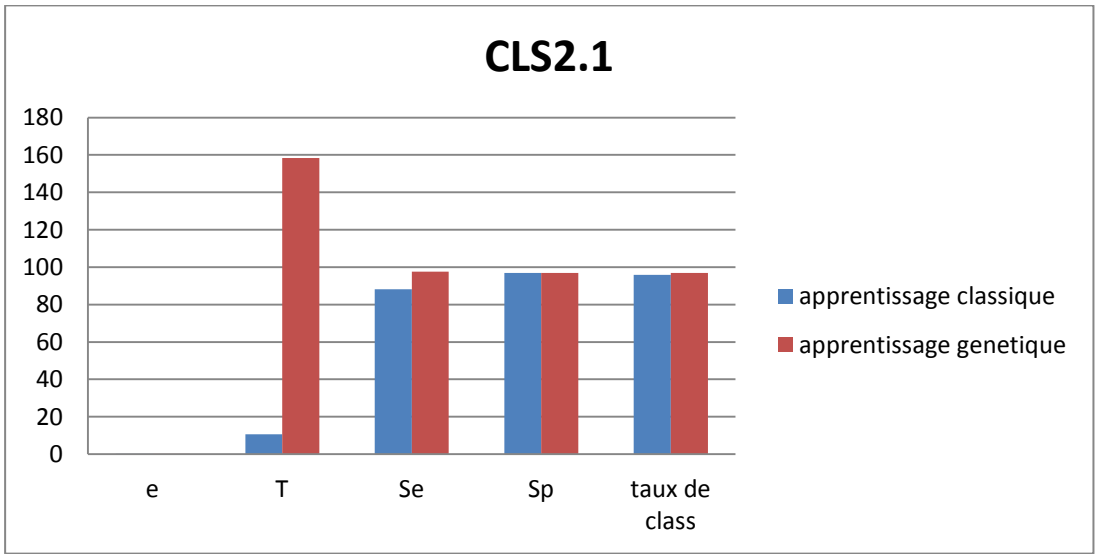


Figure4.23 : Variance des performances de CLS2.1 de l'apprentissage classique à l'apprentissage génétique

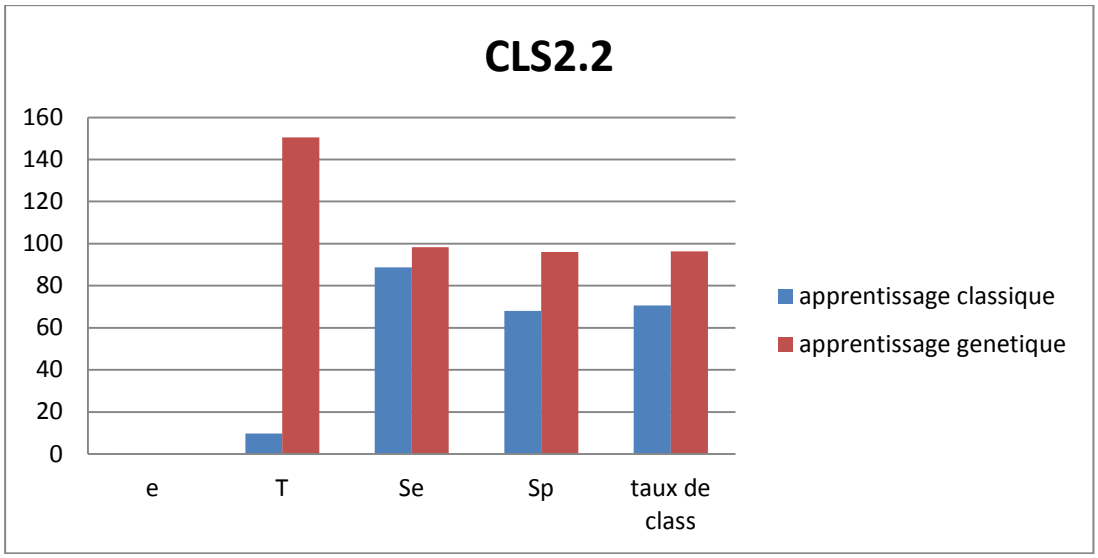


Figure4.24 : Variance des performances de CLS2.2 de l'apprentissage classique à l'apprentissage génétique

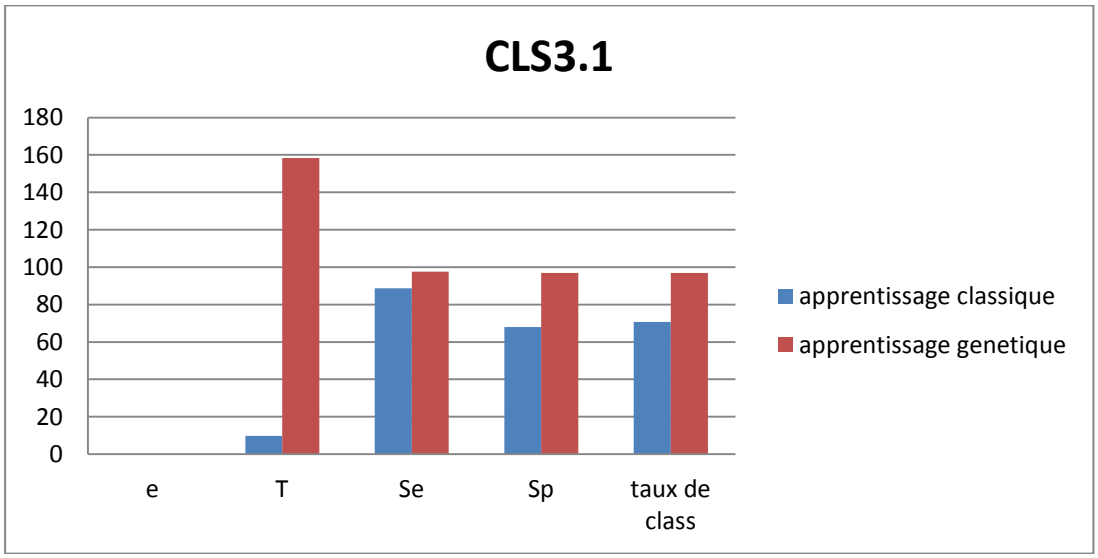


Figure4.25 : Variance de performance de CLS3.1 de l'apprentissage classique à l'apprentissage génétique

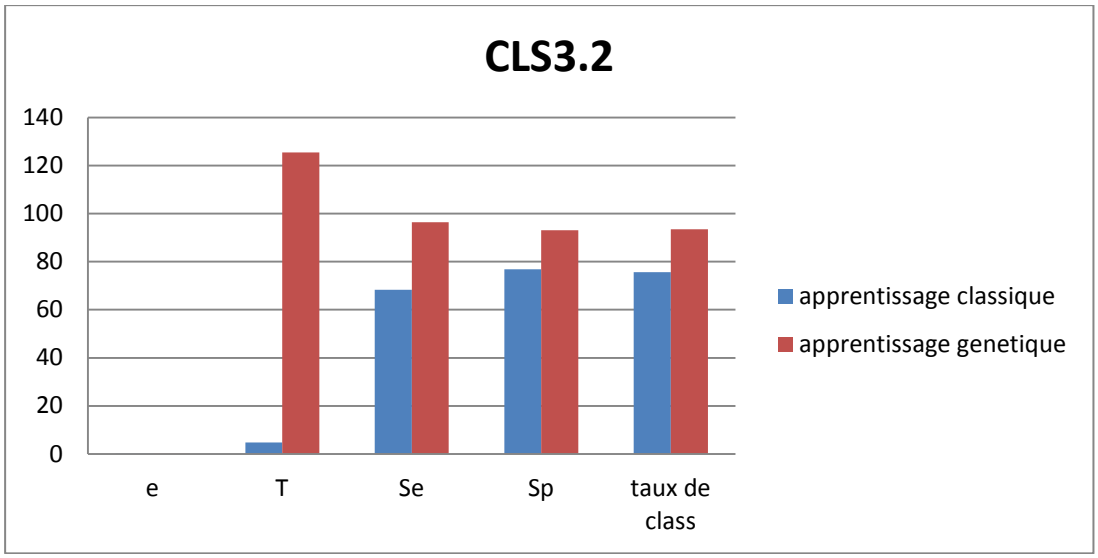


Figure4.26 : Variance de performance de CLS1.2 de l'apprentissage classique à l'apprentissage génétique

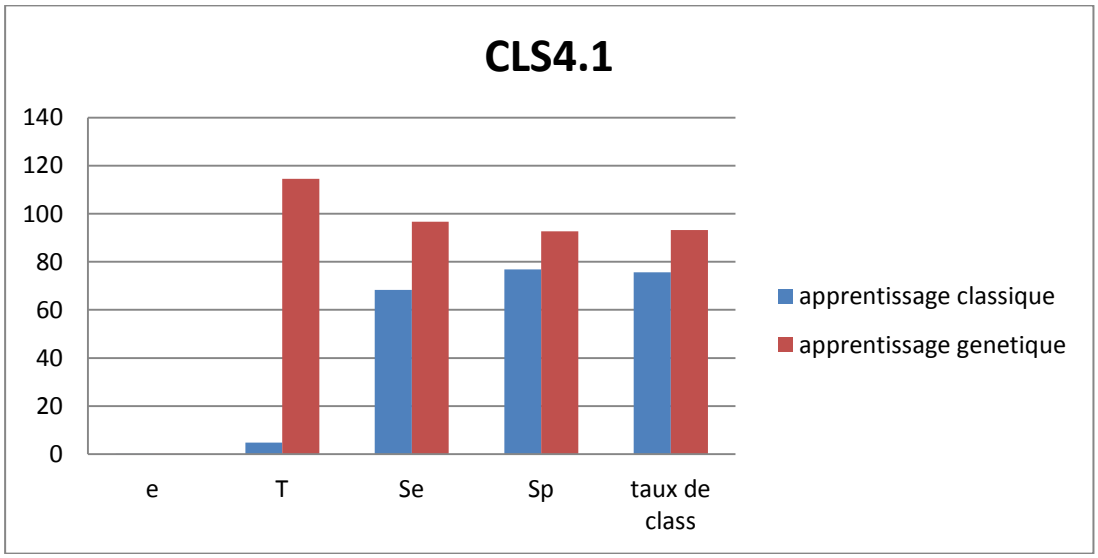


Figure4.27 : Variance des performances de CLS4.1de l'apprentissage classique à l'apprentissage génétique

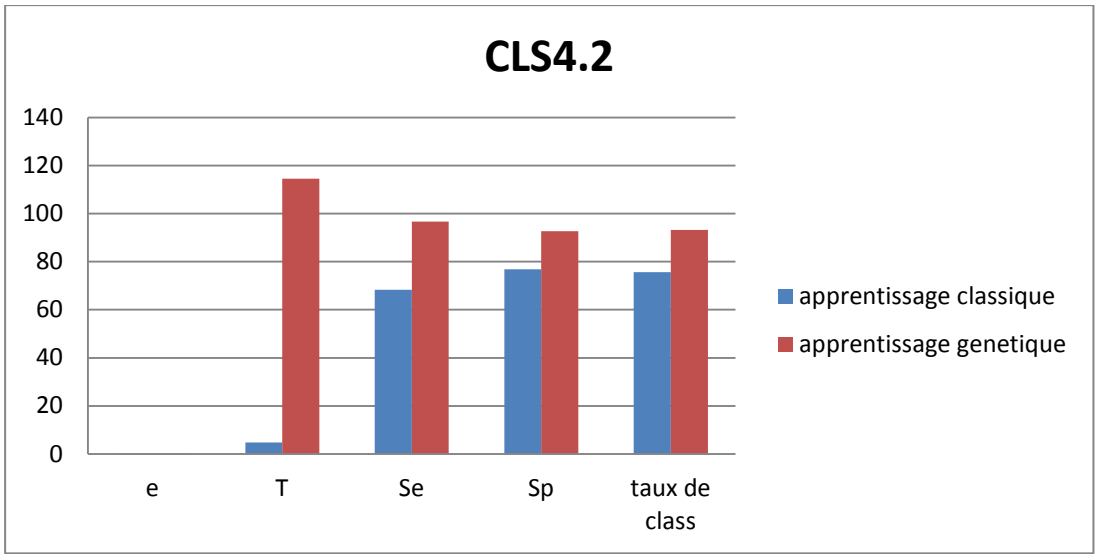


Figure4.28 : Variance des performances de CLS4.2 de l'apprentissage classique à l'apprentissage génétique

Ces histogrammes présentent la variation des performances (T, Se,Sp , τ_{class} *et e)* entre le classifieurs classiques et les classifieurs neuro-génetiques

Nous remarquons que

- Pour les classifieurs avec un nombre de caractéristiques important et un nombre de neurones cachés élevé (CLS1.1,CLS1.2,CLS1.3,CLS2.1), nous avons presque les mêmes performances pour Se,Sp et $\tau_{class)}$, mais l'apprentissage génétique présente un temps de convergence vraiment important par rapport à l'apprentissage classique (consommateur de temps).

- Pour les classifieurs qui présentent un nombre de neurones réduits (ex. CLS1.5, CLS2.2etCLS3.2) les performances de l'apprentissage génétique sont meilleurs par rapport aux performances de l'apprentissage classique, mais le temps de convergence de l'apprentissage génétique est toujours très important parapport à l'apprentissage classique.
- Pour les classifieurs qui présentent un manque de caractéristiques (EX.CLS1.1,2.1) toujours les performances de génetiques sont méilleurs parapport aux classiqsue.

  Les deux figures suivantes présentent la variation de class et e en fonction de nombre de Nc pour les deux types d'apprentissage

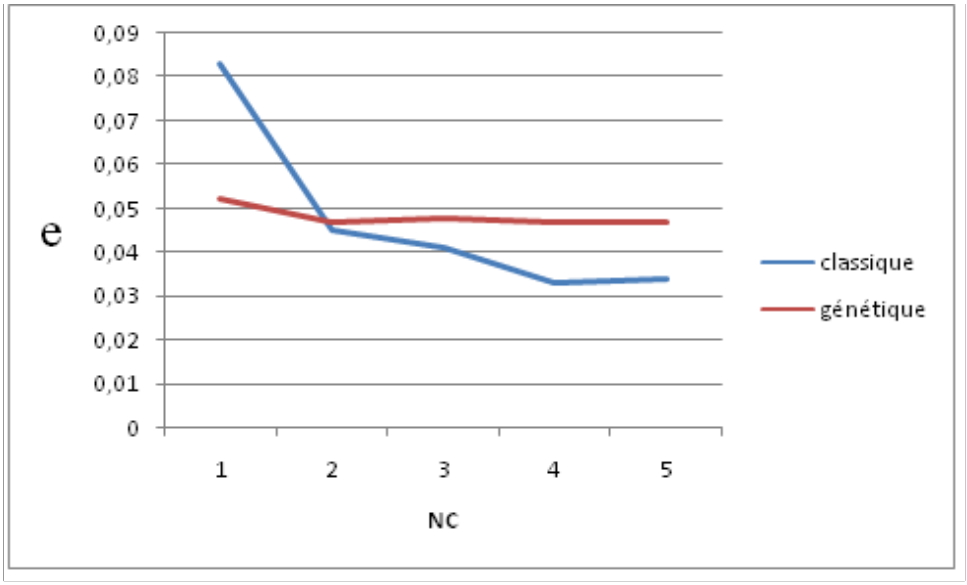


Figure4.32 : Variation de e en fonction de la structure de réseau (NC)

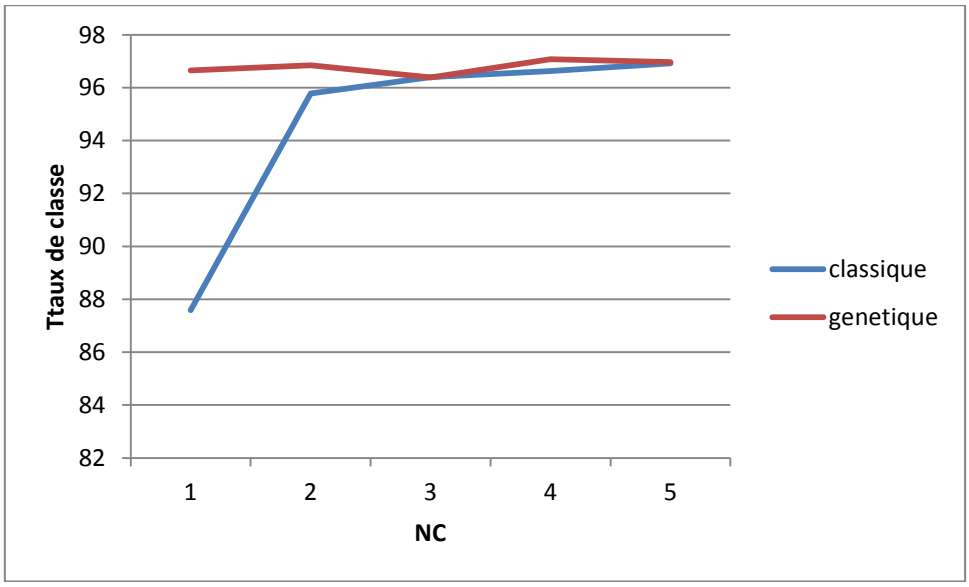


Figure4.33 : Variation de taux de classe en fonction de la structure de réseau (NC)

Nous remarquons que les courbes, qui représentent la variation de taux de classe correcte et erreur en fonction de Nc pour les classifieurs neuro-génétique est presque constante .c.à.d. la réduction de NC n'influe pas sur la performance d'un classifieurs neuro-génétique.

Par contre Nous remarquons que la courbe, qui représentent la variation de taux de classe en fonction de NC de classifieurs classique , change en fonction de Nc surtout pour Nc=1 jusqu'à 2, même la courbe qui représente la variation de ***e*** decsend en fonction de Nc. Ceci montre l'influance de nombre de Nc sur la performance des classifieurs classiques .

Les deux figures suivantes présentent la variation de taux class et l'erreur en fonction de nombre de caractéristiques présentés aux vecteurs d'entrée pour les deux types d'apprentissage .

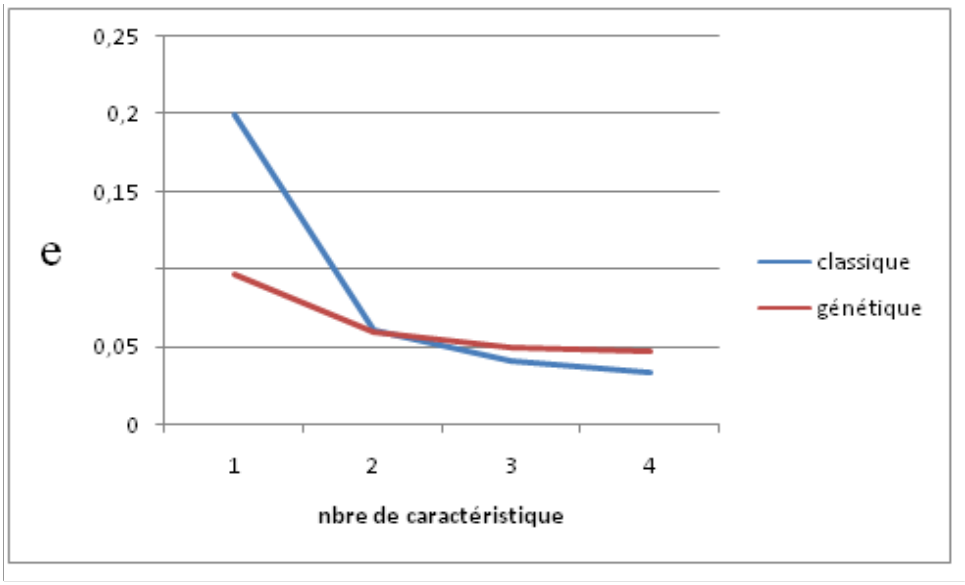


Figure4.34 : Variation de *e* en fonction de nombre de caractéristiques

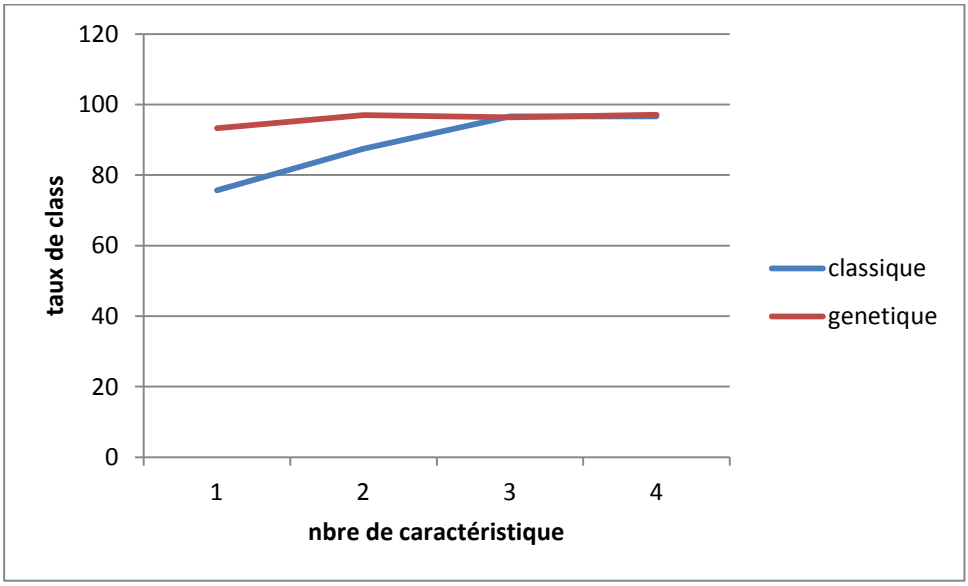


Figure4.35 : Variation de taux de classe en fonction de nombre de caractéristiques

Nous remarquons que les courbes, qui présentent la variation de taux de class et ***e*** de classifieurs neuro-génétique en fonction de nombre de caractéristiques est presque

constant, seulement le taux augmente et e diminue légerment pour une seule caractéristique.ceci montre que avec un peu de caractéristiques le classifieur neuro-génetiques doit être performant.

Par contre Nous remarquons que les courbes, qui présentent la variation de taux et $\boldsymbol{e}$ de classifieurs classique, varie en fonction de nombres de caractéristiques surtout de 1 à3 et ceci montre l'influence de nombre de caractéristiques sur la performance des classifieurs classiques

Conclusion

Après la comparaison entre les classifieurs neuro-génetiques et les classifieurs neuronaux classiques nous trouvons les conclusions suivantes :

- Un nombre de neurones cachés grand et un nombre de caractéristiques importants résultats satisfaisantes pour les deux approches .
- Un nombre de neurones cachés réduits l'apprentissage génétique est meilleur par rapport à l'apprentissage classique.
- Nombre de caractéristiques réduits l'apprentissage génétique est meilleur par rapport à l'apprentissage classique.

Nous voyons cette importance dans les données médicales qui besoin d'un grand réseau pour les classifier (imagerie médicale)

Et pour la classification des signaux biomédicaux (2D ou 3D) où l'extraction des paramètres est difficile (EEG)

- Le classifieur génétique à un inconvenant par rapport au classique : **c'est le temps de convergence**

IV.4 Réalisation des classifieurs d'arythmies cardiaques à quatre classes

Dans cette partie de notre travail, nous avons développé des classifieurs pour trois arythmies cardiaques V (ESV), L (BBG), R (BBD) et les différencier du cas normal, pour confirmer les résultats obtenus pour les classifieurs de 2 classes.

Pour ces classifieurs nous avons besoin plus de paramètre que pour les classifieurs 2 classes, nous utilisons dix paramètres .notre choix est basé sur une approche intuitive, et à l'aide de spécialistes en cardiologie, nous avons utilisé les descripteurs qui semblent les plus importants.

La mesure de ces paramètres a été réalisée en utilisant un algorithme implémenté :

IMPE « interface de mesure des paramètres de l'ECG » **[Bechar,Chikh 2003]** développée sous matlab. Cette interface a été modifiée et développée afin de pouvoir détecter tous les rythmes sur l'ensemble des enregistrements.

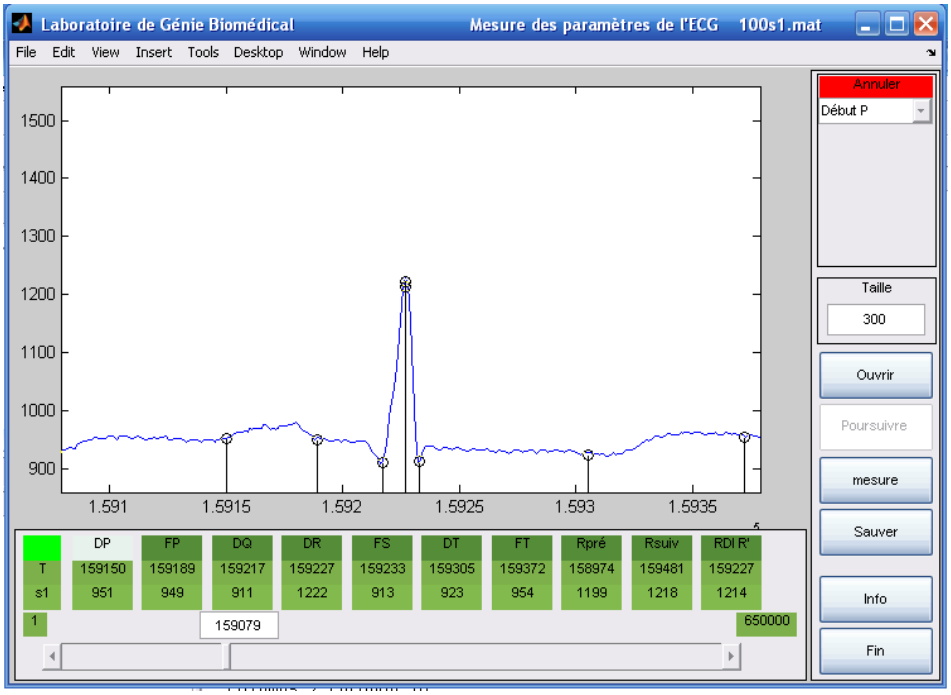


Figure4.36: Interface de Mesure des Paramètres de l'ECG.

IMPE permet de localiser les différentes ondes d'une manière manuelle par un simple clic de souris ça nous permet de palier la difficulté de la détection automatique des ondes P, Q, S et T.

Ces descripteurs permettent de caractériser un battement cardiaque.

IV.4.1 Présentation des différents descripteurs

Les critères électro cardiographiques des différentes arythmies nous ont mené à retenir des descripteurs pertinents.

Les Extrasystoles ventriculaires se caractérisent par :

-Complexe QRS élargi, de durée égale ou plus souvent supérieure ou à 0.12s .Suivi immédiatement d'onde T sans segment RST visible.

-L'absence d'onde P avant le complexe QRS élargi.

-L'existence habituelle de repos compensateur.

-l'intervalle PR varie constamment d'un complexe à l'autre sans périodicité nette.

Le diagnostic des blocs de branche gauche et blocs de branche droit repose sur des critères :

-intervalle PR de durée supérieure ou égale à 0.12s.

-Elargissement important du complexe QRS, toujours supérieur ou égal à 0.12s.

-Onde T négative. Avec un intervalle QT allongé.

-Un retard très important de la déflexion intrinsécoïde (RDI) supérieur ou égal à 0.08s.

Une description plus approfondie de ces arythmies peut être trouvée dans le chapitre1.

Les descripteurs retenus pour l'entrée du classifieur neuronal sont : Durée P ; intervalle PR ; complexe QRS ; segment ST ; intervalle QT ; RR précédent ; RR suivant ; RDI (retard de la déflexion intrinsecoïde); durée battement ; rapport RR suivant \ RR précédent.

- **Intervalle PR** : il se calcule du début de l'onde P jusqu'au début du QRS,il est de 0.12 à 0.23s.
- **Segment ST** : on le mesure de la fin de l'onde S ou R jusqu'au début de l'onde T.il est normalement horizontal ou légèrement oblique +/- isoélectrique.
- **Intervalle QT** : il se mesure du début du QRS jusqu'à la fin de l'onde T.le QT est fonction de la fréquence cardiaque.

- **Intervalle RRp et RRs :** Nous appelons RRp la distance entre le pic R du présent battement et le pic R du battement précédent. Et RRs celle entre le pic R du présent battement et le pic R du battement suivant.

- **Rapport des intervalles RR (RRs/RRp)** : Dans le cas d'un rythme régulier, le rapport RRs/RRp est un paramètre qui caractérise une classe donnée. Dans le cas d'un rythme régulier, ce rapport est voisin de 1, mais il peut largement dépasser cette valeur dans le cas d'une ESV avec repos compensatoire.

- **Largeur du QRS(LQRS) :** Ce paramètre est important pour l'identification des battements pathologiques, ces types d'arythmies sont caractérisés généralement par un large complexe QRS.

- **Durée de battement :** lors d'une variation de rythmes par une présence d'anomalies cardiaques, cette perturbation peut s'apprécier en mesurant la durée du battement. Elle est calculée du début de l'onde P jusqu'à la fin de l'onde T.

IV.4.2 Sélection de la base d'exemples

Nous avons utilisé toujours la base MIT-BHI.

Les enregistrements comprenant suffisamment d'arythmies à classer ont été choisis pour construire la base d'exemple, voir tableau4.2.

Nous avons rassemblé les battements de chaque enregistrement en quatre groupes :

_ le groupe N est composé des battements classés normaux.

_ le groupe V est composé des battements classés extrasystoles ventriculaires.

_ le groupe R est composé des battements classés blocs de branche droit.

_ le groupe L est composé des battements classés blocs de branche gauche

La base d'apprentissage affecte la performance des classifieurs neuronaux. Le critère général à respecter pour l'élaborer est de sélectionner des exemples représentatifs de toutes les classes.

Dans notre cas nous avons sélectionné plusieurs cycles de type 'N' ,'V', 'R' et 'L' dans différents enregistrements présentés dans le tableau suivant.

enregistrement	Nombre des battements 'N'	Nombre des battements 'V'	Nombre des battements 'R'	Nombre des battements 'L'
100	62	0	0	0
101	5	0	0	0
103	58	0	0	0
105	10	0	0	0
106	27	34	0	0
109	0	0	0	104
111	0	0	0	41
113	6	0	0	0
115	10	0	0	0
116	45	0	0	0
118	0	0	12	0
119	50	34	0	0
122	5	0	0	0
123	5	0	0	0
124	0	0	33	0
200	0	25	0	0
203	0	15	0	0
207	0	0	0	40
208	0	152	0	0
212	5	0	26	0
214	0	50	0	50
215	103	0	0	0

Tableau4.21 : Les enregistrements choisis de la base de données MIT-BIH

Remarque : Pour l'enregistrement 109,nous l'avons divisé en deux :109l1 contient tous les cycles (104 cycles annotés L);par contre 109l ne contient que les 25 premiers cycles annotés L.

IV.4.3 Architecture des classifieurs neuronaux

1. Couche d'entrée :

- **Pour le premier classifieur**, que nous appelons classifieur développé 1 (CLSD1).le vecteurs d'entrée est composée de 10 descripteurs décrits au paragraphe précédent.
- **Pour le deuxième et troisièmes classifieur**, que nous appelons classifieurs développé 2 (CLSD2) et classifieiur dévellopré 3(CLSD3). .le vecteurs d'entrée est composée de 5 descripteurs (.Durée P, L QRS, RR précédent ; RR suivant, durée de battement)
- **Pour le troisième classifieur** que nous appelons le classifieur développé 4 (CLSD4), le vecteurs d'entrée est composé par les 3 descripteurs (Durée P, L QRS, durée de battement)

2. Couche cachée : le choix de la taille de la couche cachée est réalisée par la regle de pyramide dans CLSD1, CLSD2 et CLSD4 pour CLSD3, nous avons utilisé 1 Nc pour voir l'influence de ca sur les résultats. La fonction d'activation est la fonction tansig

3**. Couche de sortie** : La sortie est composée d'un seul neurone qui indiquent les classes suivantes ('N', 'V', 'R' et 'L').

La fonction d'activation est une fonction linéaire (purelin) qui assure les sorties suivantes.

N=1, V=2, R=3, L=4.

IV.4.4 Apprentissage génétique

Dans cette partie, nous appliquons sur les mêmes classifieurs présentés précédemment, un apprentissage génétique en optimisant l'erreur quadratique entre la sortie réel et la sortie désiré de réseau.

Nous utilisons les mêmes paramètres de génétique fixé a la classification 2 classes, puisque nous avons la même forme de fonction à optimisé (fitness)0

Les paramètres qui varient avec la taille de chromosome (individu) sont présenté dans le tableau suivant :

CLS	Taille de chromosome	Nre de génération	Taille de pop
CLSD1	49	500	300
CLSD2	22	300	200
CLSD3	9	100	60
CLSD4	11	100	60

Tableau 4.23 : paramètres fixés pour différents classifieurs (4classe)

IV.4.6 Comparaison entre l'apprentissage génétique et l'apprentissage classique

Les histogrammes suivant présentent une comparaison avec les différents paramètres de performance d'apprentissage neuronale classique et l'apprentissage génétique

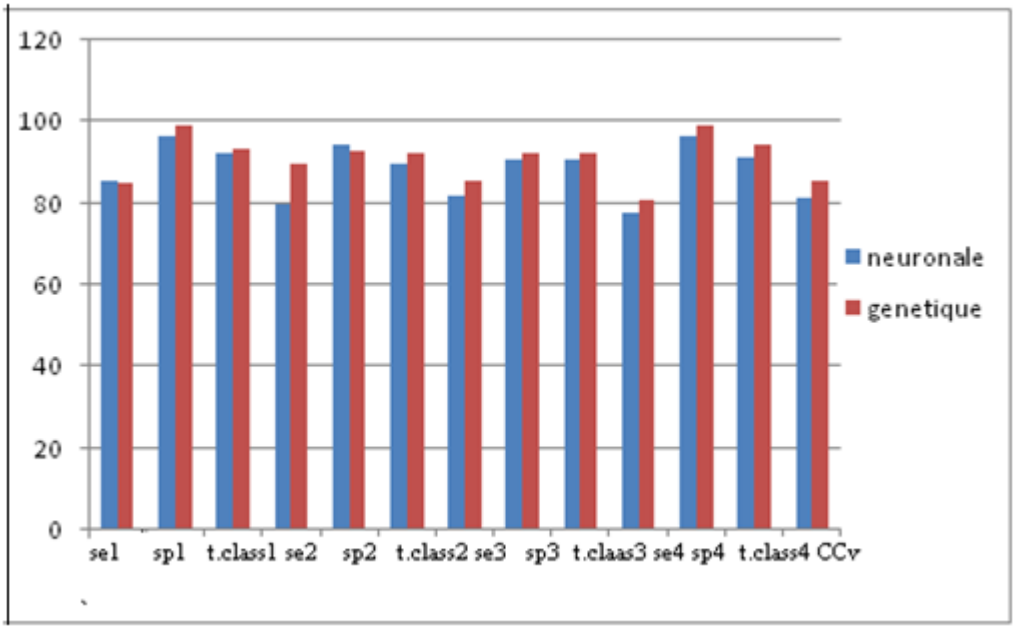


Figure4.37 : Comparaison des performances de CLSD1 entre l'apprentissage neuronal et l'apprentissage génétique

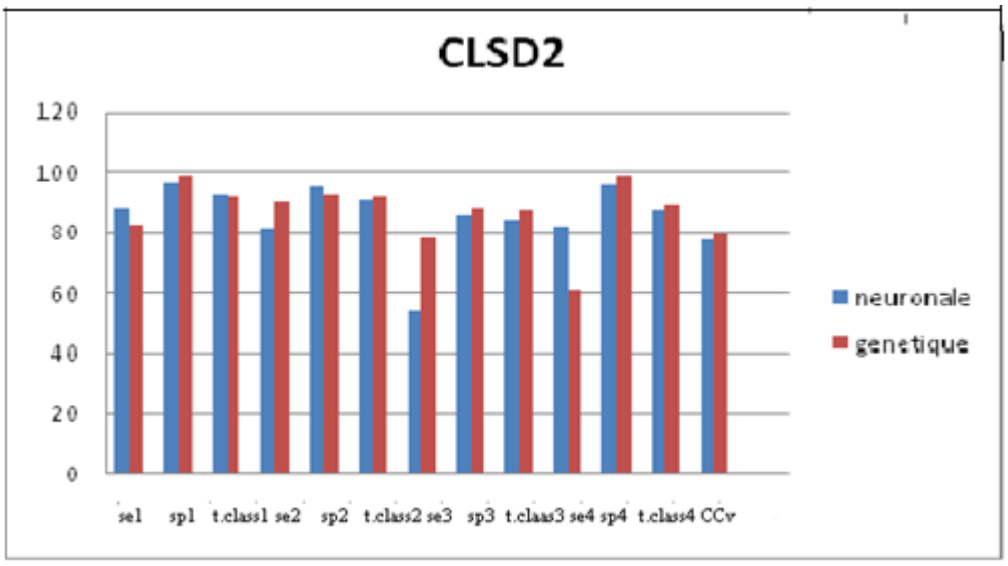


figure4.38 : Comparaison des performances de CLSD2 entre l'apprentissage neuronal et l'apprentissage génétique

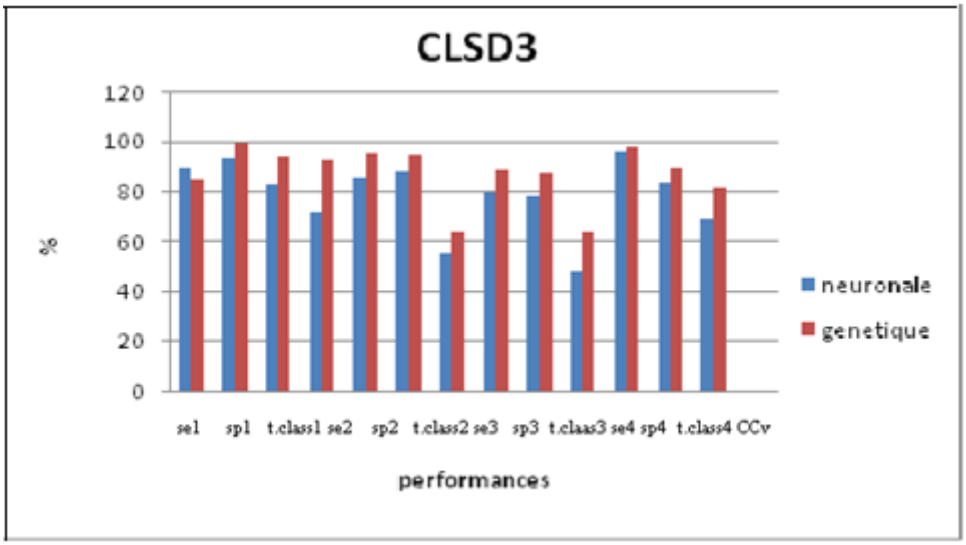


figure4.39 : Comparaison des performances de CLSD3 entre l'apprentissage neuronal et l'apprentissage génétique

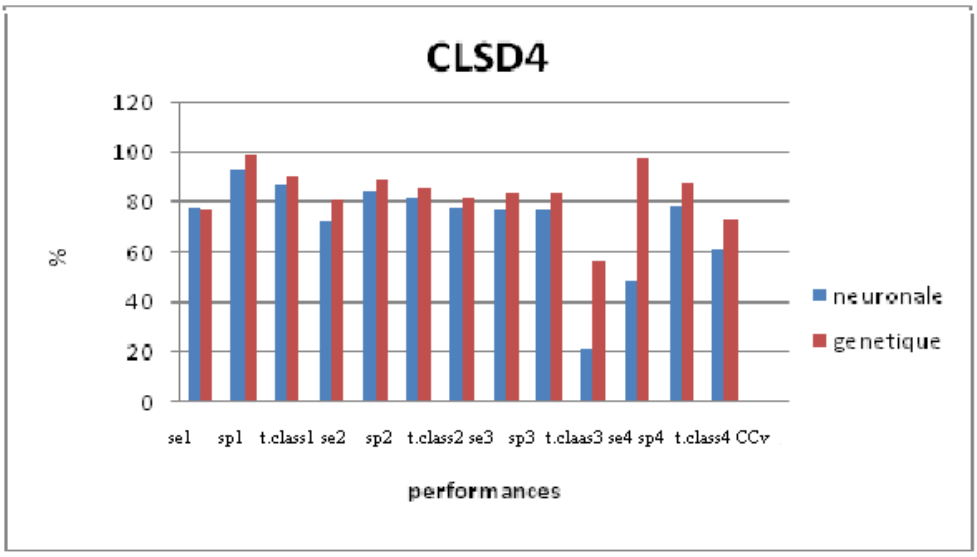


Figure4.40 : Comparaison des performances de CLSD4 entre l'apprentissage neuronal et l'apprentissage génétique

IV.4.7 Conclusion

Après la comparaison entre les classifieurs à quatre classes génétiques et neuronales, nous avons confirmé les résultats trouvés dans la classification à deux classes.

Nous avons remarqué que pour CLSD3 qui présente une architecture réduite (Nc=1) presque toutes les performances de génétiques sont meilleur, et nous avons un taux de classification correcte pour l'approche génétique égale à 81.12% par contre dans l'approche neuronale égale à 69.31%.

Pour CLSD4 qui présente un manque de caractéristique (seulement 3 caractéristique au vecteur d'entrée) nous avons une sensibilité et spécificité de BBG très faible Par contre pour le classifieur neuro- génétique nous avons une bonne SE et SP pour toutes les classes et les situations et un taux de classe égale à 73.06%.

Pour CLSD1qui présente un nombre de caractéristiques importants et un nombre de neurones élevés nous avons presque les mêmes résultats entre génétique et classique.

Donc le classifieur neuro-génétique dans toutes les situations a donné des bons résultats pour toutes les classes par contre le classique

Donc on peut conclure que l'apprentissage génétique donne un très bonne classifieurs par rapport à l'apprentissage classique dans deux cas : une structure réduite et un manque de caractéristiques.

IV.5 Apprentissage structurel

Dans cette partie, nous avons utilisé les AG non pas pour un apprentissage paramétrique, mais pour développer l'architecture de RNA qui présente un des problèmes major dans les réseaux de neurones artificiels.

L'algorithme génétique va choisi et optimisé dans une population d'architecture, une architecture meilleure, bien sur après codage (voir chapitre 3)

Nous utilisons un PMC et la génétique va choisir les liaisons entre les neurones, on passant par les étapes présentées dans chapitre3.

IV.6.1 Expérimentations

Nous avons réalisé un classifieur neuro-genetique pour la classification 2 class (N et V) avec un PMC et une taille de population =20 et les paramètres fixé dans les parties précédentes.

Puis, nous avons utilisé le même réseau mais nous avons laissé aux AG le choix de connexion entre les neurones.

1er classifieurs : c'est le classifieur CLS3.1 qui présente 2 paramètres aux vecteurs d'entrée (RRs/RRp,LQRS) etb2 neurone a la couche caché .

Le tableau suivant présente les performances moyennes après 10 phases d'apprentissage

classifieur	*e(moy)*	*SE(moy)(%)*	*SP(moy)(%)*	$\tau_{class(moy)(\%)}$
CLS3.1	**0.10**	**87.7745**	**83.3625**	**83.92**

Tableau4.24 :performances de classifiueurs CLS3.1en utilisant PMC classique

Puis sur ce réseau nous appliquons un apprentissage structurel pour le chois de liaison entre les neurones tel que « 1 »présente une liaison et « o » pas de liaison (voir chapitre3) Les résultats obtenus **[1, 1, 1, 1, 1, 1, 1,1]**

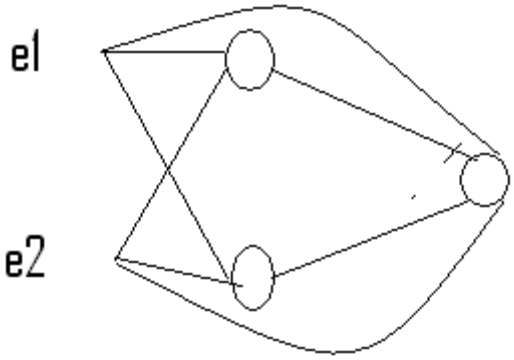


Figure4.41 :La structure de CLS3.1choisie par AG

Puis avec cette structure nous réalisons des apprentissages paramétriques que avec la structure classique et avec les mêmes paramètres.

individu	*E(moy)*	*SE(moy)(%)*	*SP(moy)(%)*	$\tau_{class(moy)(\%)}$
moyenne	**0.0814**	**95.6656**	**98.0681**	**97.7619**

Tableau4.25 :performances de classifiueurs CLS3.1en utilisant la structure choisie par AG

On remarque que les résultats de la nouvelle architecture sont meilleurs par rapport à l'architecture classique .pour la structure développé par AG on a attient un τ_{class} moyen égale 97.76%, par contre pour la structure classique τ_{class} moyenne égale à 83.92%

2er classifieurs : c'est le classifieur CLS2.1 qui présente 3paramètres aux vecteurs d'entrée (RRs,RRp,LQRS) et2 neurone à la couche cachée .

classifieur	*e(moy)*	*SE(moy)(%)*	*SP(moy)(%)*	$\tau_{class(moy)(\%)}$
CLS2.1	**0.1313**	**67.7819**	**97.1660**	**93.3957**

Tableau4.26 :performance de classifiueurs CLS2.1en utilisantun PMC classique

Puis ce sur ce réseau nous appliquons un apprentissage structurelle pour le choix des liaisons entre les neurones.

Les résultats obtenus [1, 0, 1, 0, 1, 1, 1, 1, 1,1]

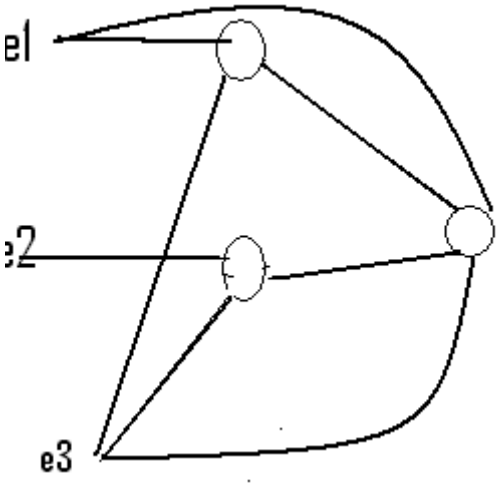


Figure4.42 : La structure de CLS2.2choisie par AG

Puis avec cette structure nous réalisons des apprentissages paramétriques que le premier et avec les mêmes paramètres

classifieur	*e(moy)*	*SE(moy)(%)*	*SP(moy)(%)*	$\tau_{class(moy)(\%)}$
CLS2.1	**0.0596**	**98.0085**	**96.2158**	**96.4460**

Tableau4.61 :performance de classifiueurs CLS3.1en utilisant structure choisie par AG

IV.6.2 Conclusion

Malgré qu'avec une taille de population égale à20 les classifieurs donnent des résultats insuffisants avec un temps de convergence réduit, on peut voir des résultats meilleurs si on utilise les AGs pour choisir la structure des réseaux neuronaux (la connexion entre les neurones).donc d'après nos * résultats les AGs donnent le bons choixd'architecture de RNA.

IV.6 Conclusion de chapitre

L'objectif de ce chapitre est d'hybridé les algorithmes génétiques avec les réseaux de neurones pour réaliser des classifieurs des signaux bio médicaux (application était sur l'ECG), pour voir est ce qu'on peut éviter les problèmes major des classifieurs neuronaux classiques : minima locaux, choix d'architecture.

Et après plusieurs expérimentations classiques et neuro-génétiques nous concluons que dans les cas ou le nombre de neurones cachés est réduit ou il y a un manque de caractéristiques informatives de signal les classifieurs neuro-génétiques donnent des résultats meilleurs par rapport aux cas classiques.

Nous voyons l'intérêt de l'approche neuro-génétiques dans les cas où des donnés médicaux ont besoin d'un grand réseau pour les classifier et dans une extraction de paramètres difficile. Le seul inconvenant de classifieur neuro-génétiques est la consommation énorme du temps.

Aussi nous avons utilisé les algorithmes génériques pour une autre approche ; le choix d'architecture. Nous avons laissé pour ce dernier le choix de liaisons entre les neurones d'un PMC et nous avons conclue que l'algorithme génétique propose une architecture optimale.

Conclusion générale

La classification des différents signaux biomédicaux, (de deux dimensions ou trois), est une tâche essentielle des systèmes aide au diagnostic médical (ADM).

Dans ce travail, nous avons présenté une méthode pour classifier ces signaux dite :**neuro génétique** . Cette technique qui combine les réseaux de neurones artificiels avec la technique d'optimisation globale (algorithmes génétiques), pour améliorer la performance par rapport aux classifieurs classiques qui utilisent la rétro propagation (descente de gradient).

Le Modèle neuro-genetique implémenté appliqué sur le signal (ECG), a un indice de succès par rapport au modèle classique dans les deux cas : nombre de neurones cachées réduits ou manques de caractéristiques informatives au vecteur d'entrée. Et ceci montre la puissance des AGs pour avoir des solutions optimales menées dans des cas difficiles

L'importance des AGS, est vue dans le cas des signaux biomédicaux qui ont besoin d'un grand réseau pour les classifier, où l'extraction des paramètres (caractéristiques) est difficile (EEG). Mais ce modèle a deux inconvenants : non explicites (boite noir), et le temps de convergence important (consommateur de temps).

Aussi dans ce travail, nous avons utilisé les AGs pour choisir l'architecture des RNAs, cette structure a donné des meilleurs résultats par rapport aux architectures neuronales classiques.

Perspectives

Comme perspective a ce travail, nous proposons :

- D'appliquer ce modèle sur différents signaux biomédicaux : EEG, EMG et les différentes images médicales.
- Hybridé ce modèle, avec la logique flou c.à.d. réalisation d'un modèle neuro-flou-génétique pour assurer l'explicité et l'interprétabilité des résultats.

Annexes

Annexe A

Les neurones biologiques

Le cerveau est un organe caractérisé par l'interconnexion d'un nombre ´élevé d'unités de traitement simples : **les neurones**.

Le comportement de ce réseau de neurones est détermine par son architecture (le nombre des unités et la manière dont elles sont connectées) et par les poids affecté à chaque connexion ou synapses.

Le cerveau humain comporte environ 1011 neurones, chacun relié en moyenne `a plus de 1000 autres.

Un neurone est une cellule constituée principalement de trois parties qui, vis-à-vis des transferts d'information, ont un rôle fonctionnel bien défini : sont les dendrites, le soma et l'axone.

- **Soma (le corps cellulaire)**

Le soma, aussi appelé le corps cellulaire, constitue l'élément principal du neurone. Il contient le noyau cellulaire renfermant lui même le génome, c'est-à-dire l'information génétique de l'organisme dont le neurone est issu.

Ce noyau baigne dans un *cytoplasme* (pour faire simple, un fluide) contenant une très grande quantité de molécules, dont particulièrement les éléments de la "machinerie cellulaire" qui permettent à la cellule de traduire ses gènes en protéines fonctionnelles (ex : enzymes) ou constitutives (ex : protéines du cytosquelette).

- **L'axone**

L'axone est le prolongement cellulaire principal et c'est grâce à lui que le neurone va transmettre les informations. A son extrémité se trouve l'arborisation terminale (= pôle émetteur de la cellule) où se trouve un nombre plus ou moins important de terminaisons synaptiques.

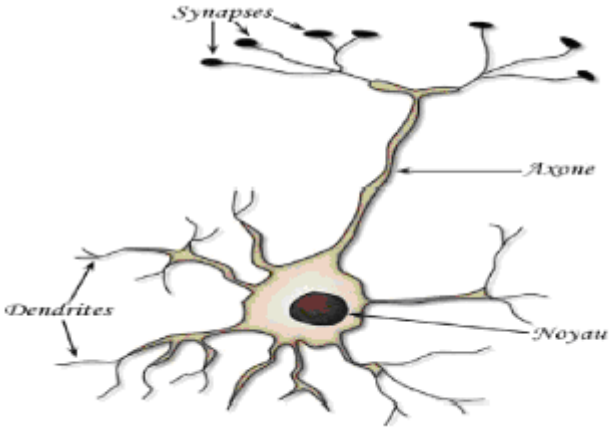
- **Les dendrites**

Les dendrites représentent le deuxième type de prolongement cellulaire et se trouvent en grande quantité autour du soma.

Les dendrites constituent le pôle récepteur de la cellule et c'est à cet endroit que vont se faire les connexions avec les autres neurones.

- **La synapse**

C'est une jonction entre deux neurones et généralement (car il existe des synapses axo-axonales par exemple) entre l'axone d'un neurone et une dendrite d'un autre neurone



.

Figure A.1 : neurone biologique

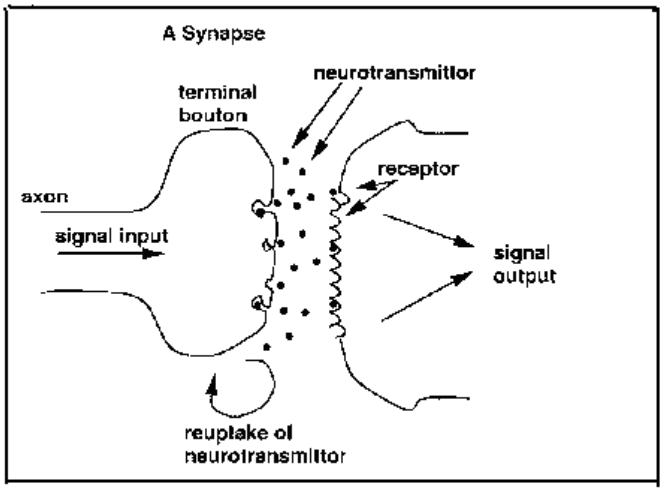
Fonctionnement

Au point de vu fonctionnel, il faut considérer (pour simplifier) le neurone comme une entité polarisée, c'est-à-dire que l'information ne se transmet que dans un seul sens : des dendrites vers l'axone.

Pour rentrer un peu dans le détail, le neurone va donc recevoir des informations, venant d'autres neurones, grâce à ses dendrites. Il va ensuite y avoir sommation, au niveau du corps cellulaire, de toutes ces informations et via un *potentiel d'action* (=un signal électrique) le résultat de l'analyse va transiter le long de l'axone jusqu'aux terminaisons synaptiques. A cet endroit, lors de l'arrivée du signal, des vésicules synaptiques vont venir fusionner avec la membrane cellulaire, ce qui va permettre la libération des neurotransmetteurs (=médiateurs chimiques) dans la fente synaptique. Le signal électrique ne pouvant pas passer la synapse (dans le cas d'une synapse chimique), les neurotransmetteurs permettent donc le passage des informations, d'un neurone à un autre.

Au niveau post-synaptique, sur la membrane dendritique, se trouvent des récepteurs pour les neurotransmetteurs. Suivant le type de neurotransmetteur et le type des récepteurs, l'excitabilité du neurone suivant va augmenter ou diminuer, ce qui fera se propager ou non l'information.

Les synapses possèdent une sorte de «mémoire» qui leur permet d'ajuster leur fonctionnement. En fonction de leur l'«histoire», c'est-à-dire de leur activation répétée ou non entre deux neurones, les connexions synaptiques vont donc se modifier.



<u>Figure A.2:</u> fonctionnement d'un neurone biologique

Annexe B

L'histoire des réseaux de neurones

Historique

- **Les prémices**

Dans la période de 1940 à 1956, les sciences et techniques de la cognition s'élaborèrent peu à peu et de l'ancienne cybernétique naquit les sciences cognitives. La volonté de fonder une science "naturelle" de l'esprit émergea quasiment au même moment à cette époque, en Europe et aux Etats-Unis.

Par les efforts conjugués de chercheurs d'origines très diverses (Neurosciences, Intelligence artificielle, Linguistique, Psychologie et Epistémologie principalement), un nouveau champ d'investigation pluridisciplinaire, que sont les sciences cognitives, vit donc le jour.

L'apparition des réseaux de neurones artificiels s'inscrit complètement dans cette genèse des sciences cognitives, et leur origine tient dans la volonté de modéliser, de façon mathématique, les neurones biologiques.

- **Les pionniers**

Les premiers à proposer un modèle, sont deux biophysiciens de Chicago, McCulloch et Pitts, qui en 1943 inventent le premier neurone formel qui portera leurs noms (neurone de McCulloch-Pitts ou automate à seuil).

Quelques années plus tard, en 1949, c'est Hebb qui proposera une formulation du mécanisme d'apprentissage, sous la forme d'une règle de modification des connexions synaptiques (règle de Hebb).

- **Le premier RNA**

C'est en **1958** qu'apparaît, proprement dit, le premier réseau de neurones artificiels, grâce aux travaux de **Rosenblatt** qui conçoit le fameux **Perceptron**.

Le Perceptron est inspiré du système visuel et possède une couche de neurones d'entrée ("perceptive") ainsi qu'une couche de neurones de sortie ("dcisionelle"). Ce réseau parvient à apprendre à identifier des formes simples et à calculer certaines fonctions logiques. Il constitue donc le premier système artificiel présentant une faculté jusque là réservée aux êtres vivants : la capacité d'apprendre par l'expérience.

Malgré tout l'enthousiasme que soulève le travail de Rosenblatt dans le début des années 60, la fin de cette décennie sera marquée en **1969**, par une critique violente du Perceptron par **Minsky et Papert**. Ils montrent dans un livre (« *Perceptrons* ») toutes les limites de ce modèle et soulèvent particulièrement l'incapacité du Perceptron, à résoudre les problèmes non linéairement séparables, tels que le célèbre problème du *XOR*. Il s'en suivra alors, face à la déception, une période noire d'une quinzaine d'années dans le domaine des réseaux de neurones artificiels**.**

- **Le RNA multicouches**

Il faudra attendre le début des années **80** et le génie de **Hopfield** pour que l'intérêt pour ce domaine soit de nouveau présent. Hopfield démontre, en **1982**, tout l'intérêt d'utiliser des réseaux récurrents (dit "feed-back") pour la compréhension et la modélisation des processus mnésiques. Les réseaux récurrents constituent alors la deuxième grande classe de réseaux de neurones, avec les réseaux type perceptron.

En parallèle des travaux de Hopfield, **Werbos** conçoit son algorithme de rétro propagation de l'erreur, qui offre un mécanisme d'apprentissage, pour les réseaux multicouches de type Perceptron (appelés MLP pour Multi layer Perceptron), fournissant ainsi un moyen simple d'entraîner les neurones des couches cachées.

Cet algorithme de "back propagation" ne sera pourtant popularisé qu'en **1986**, via un article publié dans *Nature* et un livre (*« Parallel Distributed processing »)* de **Rumelhrt et al.**

Depuis la fin des années **80**, l'intérêt pour les réseaux de neurones artificiels croître et ceci dans de plus en plus de domaines

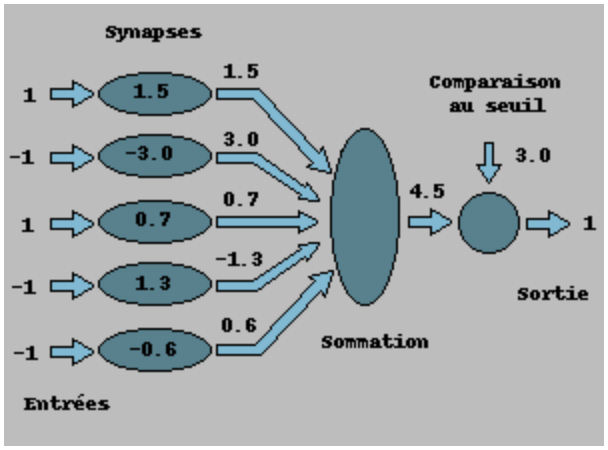
Annexe C

Neurone formel

A. Présentation

Comme je l'ai indiqué dans l'historique, le premier neurone formel est on apparu en 1943 et on le doit à messieurs Mac Culloch et Pitts.

Schématiquement on peut représenter leur modèle de la façon suivante :



figureC.1 : présentation d'un neurone formel

Le neurone formel est donc une modélisation mathématique qui reprend les grands principes du fonctionnement du neurone biologique et particulièrement, la sommation des entrées. Sachant qu'au niveau biologique, les synapses n'ont pas toutes la même «valeur» (pour simplifier disons que les connexions entre les neurones sont plus ou moins fortes), les auteurs ont donc créé un algorithme qui pondère la somme de ses entrées par des poids synaptiques (=coefficients de pondération).

En résumé, un neurone formel réalise simplement une somme pondérée de ces entrées, ajoute un seuil à cette somme et fait passer le résultat par une fonction de transfert pour obtenir sa sortie.

B. Interprétation mathématique

D'un point de vue mathématique, le neurone formel peut être représenté de la manière suivante :

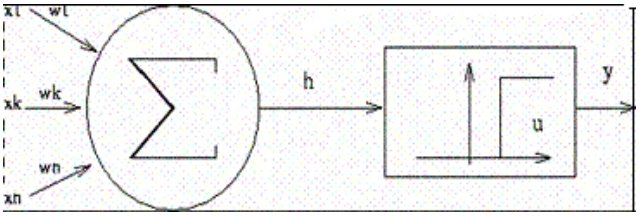


Figure C 2 : présentation mathématique d'un neurone de formel

La formule mathématique s'écrit ainsi :

$$y = f\left(\sum_{j=1}^{n} w_j x_j - \theta\right)$$

f : La fonction seuil

1-Les entrés

Elles peuvent être :

- Booléennes.
- Binaires (0, 1) ou bipolaires (-1, 1).
- Réelles.

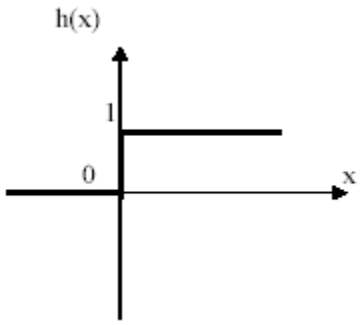
2-Fonction d'activation

Cette fonction permet de définir l'état interne du neurone en fonction de son entrée totale, citons à titre d'exemple quelques fonctions souvent utilisées :

2-a-Fonction binaire a seuil

Exemple : Fonction Heaviside définie par

$$h(x)=\begin{cases}1 & \text{si } x \geq 0\\ 0 & \text{sinon}\end{cases}$$



FigureC 3 : Fonction Heaviside

Fonction Signe définie par :

$$Sgr(x)=\begin{cases}+1 & \text{si } x \geq 0\\ -1 & \text{sinon}\end{cases}$$

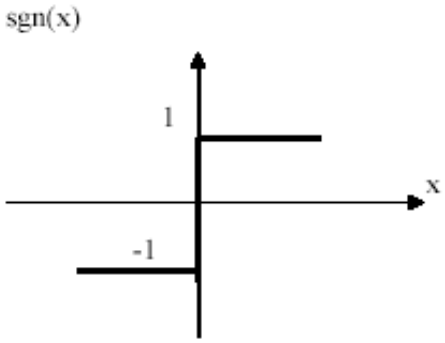


Figure C4 : Fonction Signe

Le seuil introduit une non linéarité dans le comportement du neurone, cependant il limite la gamme des réponses possibles à deux valeurs.

2. b-Fonction linéaire

C'est l'une des fonctions d'activations les plus simples, sa fonction est définie par :

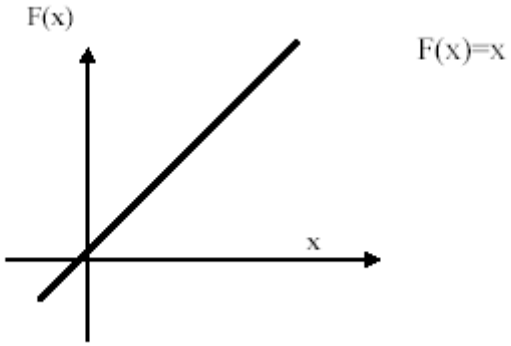


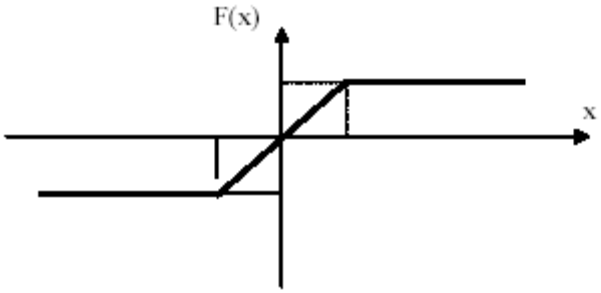
Figure C5 : fonction linéaire

2. c- Fonction linéaire à seuil ou multi seuils

On peut la définir comme suit :

$$F(x) = \begin{cases} x & x \in [\,u, v\,] \\ v & \text{si } x \geq v \\ u & \text{si } x \leq u \end{cases}$$

Cette fonction représente un compromis entre la fonction linéaire et la fonction seuil : entre ses deux barres de saturation, elle confère au neurone une gamme de réponses possibles. En modulant la pente de la linéarité, on affecte la plage de réponse du neurone.

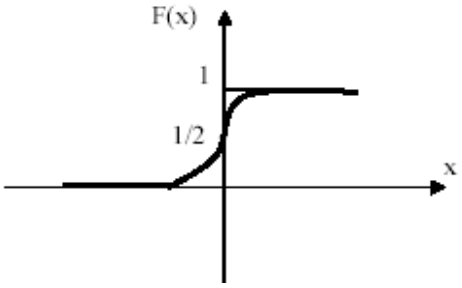


. **Figure C6** : **Fonction linéaire à seuil ou multi seuils**

2. d- Fonction sigmoïde

Elle est l'équivalent continu de la fonction linéaire. Etant continu, elle est dérivable, d'autant plus que sa dérivée est simple à calculer, elle est définie par :

$$f(x) = \frac{1}{1 + e^{-x}}$$



FigureC7 : **Fonction sigmoïde**

3- Fonction de sortie

Elle calcule la sortie d'un neurone en fonction de son état d'activation. En général, cette fonction est considérée comme la fonction identité.

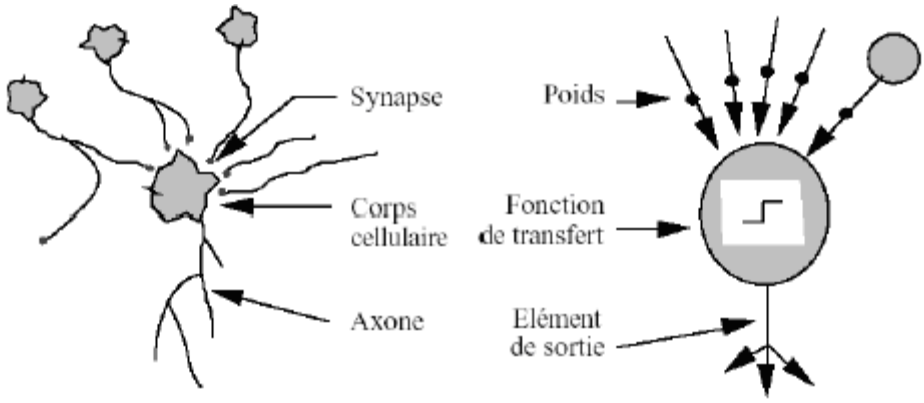
Elle peut être :

- Binaire (0, 1) ou bipolaire (-1, 1)
- Réelles.

On pourra résumer cette modélisation par le **tableau 1**, qui nous permettra de voir clairement la transition entre le neurone biologique et le neurone formel.

Neurone biologique	Neurone artificiel
Synapses	Poids de connexions
Axones	Signal de sortie
Dendrite	Signal d'entrée
Somma	Fonction d'activation

Tableau C1 : la transition entre le neurone biologique et le neurone formel



FigureC 8 : comparaison entre un neurone biologique et un neurone artificiel

C. L'utilisation

Pour fonctionner, un neurone formel utilise des *entrées* qui sont des grandeurs réelles. Si on "relie" un neurone au monde extérieur par l'intermédiaire de capteurs, il peut réaliser une simple analyse de ce qu'il "perçoit". Si on représente les valeurs observées par le neurone sous forme d'un vecteur, le neurone réalise alors un découpage de son espace d'entrée (l'espace vectoriel auquel appartient le vecteur d'observation) en deux zones : la zone *d'activité* dont les vecteurs donnent une sortie égale à 1 et la zone *d'inactivité* dont les vecteurs donnent une sortie égale à 0. Comme le calcul effectué est en fait linéaire, la

séparation l'est aussi. Les coefficients synaptiques et le seuil définissent l'équation d'un hyperplan qui est la frontière de la séparation entre les deux zones.

D. Les limitations

En fait, un unique neurone est particulièrement limité car il ne sait calculer qu'une séparation linéaire. Or, si on choisit aléatoirement deux ensembles de vecteurs dans un espace vectoriel de dimension n, la probabilité qu'ils soient linéairement séparables décroît très vite avec le nombre de vecteurs (en fait pour n grand, elle peut être considérée comme nulle dès que le nombre de vecteurs en plus grand que 2n). En termes plus simples, les ensembles linéairement séparables sont plutôt rares.

Annexe D

Anatomie et physiologie du cœur

1.Anatomie du cœur

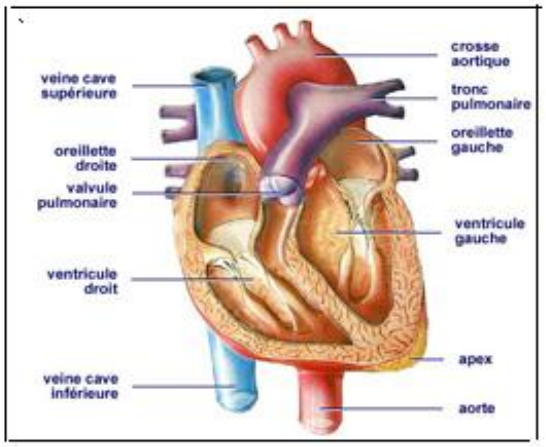
Le cœur est un organe musculaire creux situé dans la partie antéro-inferieur du thorax entre les poumons dans le médiastin.

Il est constitué de trois tissus : l'endocarde, le myocarde et le péricarde.

Le cœur comprend quatre cavités, deux oreillettes et deux ventricules, et quatre valvules : deux valvules auriculo-ventriculaire (mitrale et tricuspide) et deux valvules à la sortie des cavités ventriculaires (valvules aortique et valvule pulmonaire).

Endocarde : tunique interne du cœur, elle enta pise les cavités constitues les valvules.

Myocarde : tissus musculaire cardiaque.
Péricarde : enveloppe fibreuse du cœur
Valvules : 4 replis membraneux canalisent le sang à l'intérieur du cœur.



figureD1: Coupe du cœur

2.Fonctionnement du cœur

La fonction du cœur est de faire circuler le sang vers les différentes parties du cœur humain par deux processus distincts.

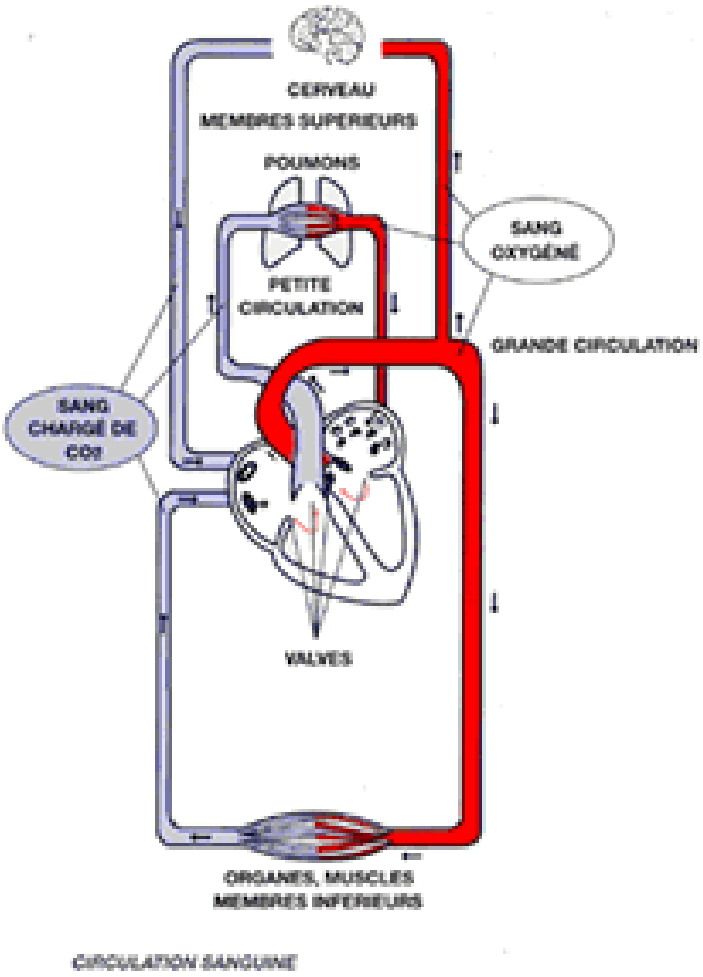
Lorsque le cœur est relâcher (diastole), les oreillettes aspirent le sang venant des veines : les veines pulmonaires pour l'oreillette gauche, les veines caves supérieures et inférieures pour l'oreillette droite. Elles se remplissent de sang (oxygéné pour la gauche, vicié pour la droite.

La contraction du cœur (systole) commence par celles des oreillettes, le sang est chassé dans les ventricules respectifs avec ouvertures des valves mitrale (à gauche) et tricuspide (à droite).la contraction atteint les ventricules qui éjectent alors le sang dans l'aorte (à gauche) et le tronc pulmonaire (à droite) avec ouverture des valve correspondantes et fermeture des valve mitrale et tricuspidienne. En suite relâchement du cœur avec fermeture des valves aortique et pulmonaire.

Donc le cœur est la pompe d'un double système circulatoire :

Circuit gauche (le grand circuit) : comprend la partie gauche du cœur (l'oreillette et le ventricule gauche et l'aorte), qui va distribuer l'oxygène à tout l'organisme.

Circuit droit (le petit circuit) : c'est la circulation pulmonaire. Elle comprend la partie droite du cœur (l'oreillette et le ventricule droit, l'artère pulmonaire, les poumons, et les veines pulmonaires. Elle permet au sang de se charger en oxygène.



figureD.2 : **circulation sanguine**

3.Électrophysiologie cellulaire

Les phénomènes physiologiques qui précèdent les mécanismes de contraction et relaxation sont de nature électrochimique.

Les liquides à l'intérieur et à l'extérieur des membranes cellulaires du corps sont des solutions d'électrolytes composées d'ions positifs et négatifs.

Le courant circule entre des ions de polarité opposée. Il se propage des ions négatifs vers les ions positifs dans le liquide extracellulaire.

Lorsque les cellules musculaires du cœur sont au repos, le liquide extracellulaire est positif (polarisation). Il n'y a pas donc de passage de courant, mais lorsqu'une cellule musculaire du cœur est stimulées, il se produit une modification dans la polarité de part et d'autre de la membrane cellulaire. Une différence existe alors entre cette cellule et sa voisine, et un courant prend naissance ou circule entre les deux jusqu'à ce que toutes les cellules de la même masse musculaire aient été stimulées et aient modifie leur polarité (dépolarisation). Après la dépolarisation les cellules reviennent à son état de repos. Il se produit une nouvelle inversion de potentiel, les ions positifs allant de nouveau à l'extérieur des cellules. Ce processus appelé repolarisation.

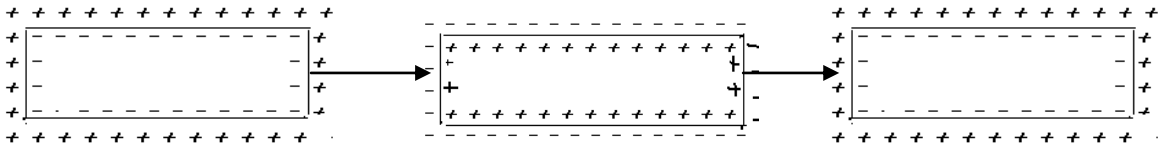


Figure D.3 : dépolarisation et repolarisation de la cellule

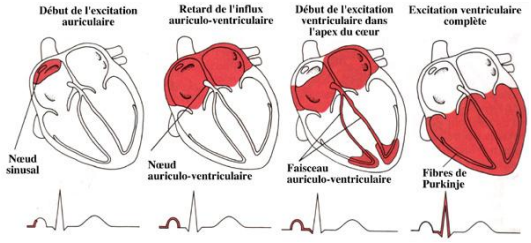
4. Activation du cœur

Pour que le cœur puisse se contracter, il doit être d'abord ; Stimulé ; le stimulus doit pouvoir diffuser rapidement et efficacement à toutes les régions du myocarde**.**

Une onde électrique part du nœud sinusal (de Keith&Flack) déclanche la dépolarisation de cellule myocardique auriculaire et de proche en proche l'excitation se répond dans les oreillettes (contraction auriculaire) pendant 0.1s. Puis l'excitation gagne le nœud auriculo ventriculaire (d'Aschoff-Tawara), à ce niveau, il se produit une pause de 0.12 à0.20s, permettre au sang de pénétrer dans les deux ventricules. Après cette pause le nœud (AV) est réellement excité et l'onde d'excitation propage dans le tronc commun du faisceau de His, puis dans sa branche droite et ses deux branches gauches (antérieur & postérieur)

.l'influx diffuse à traverse le fin réseau de Purkinje et atteint enfin le myocarde ou il se propage de l'endocarde vers l'épicarde permettant leur dépolarisation puis leur contraction pendant une durée de 0.06s à 0.08s.

Après cette phase d'activité, le cœur passe dans un état repos momentané permettant aux cellules myocardiques de se repolarisés .ce qui les rend de nouveau stimulable**.**



<u>figureD.4</u> : différentes étapes d'activation du coeur

Annexe E

Principes et techniques d'enregistrement d'un électrocardiogramme

Pour faire un électrocardiogramme, on capte les courants d'origine cardiaque sur la surface cutanée l'aide des électrodes métalliques reliées à un appareil qui amplifie ces courants pour permettre leur enregistrement.

L'activité électrique peut être captée à partir de n'importe quel point de la surface cutanée mais pour que les tracés soient interprétables par des médecins différents et pour qu'on puisse les comparer d'un sujet à l'autre, on a recours à des dérivations standardisées.

Une dérivation est le circuit électrique constitué par l'emplacement des électrodes, par les électrodes elles-mêmes, par les fils conducteurs et par le galvanomètre de l'électrocardiographe.

Les dérivations sont choisies de façon à recueillir l'activité électrique cardiaque dans deux plans de l'espace thoracique orientés perpendiculairement l'un à l'autre : le plan frontal et le plan horizontal.

1. Les dérivations dans le plan frontal

On enregistre des dérivations dites périphériques parce que les électrodes sont placées loin du cœur, ces dérivations se divisent en deux catégorie : bipolaire et unipolaire.

Dérivations bipolaires (standards)

Les dérivations bipolaires comportent deux électrodes placées de façon symétrique et équidistance du cœur : les deux électrodes sont actives (exploratrice) l'un positive et l'autre négative.

Les électrodes sont placées sur trois membres : bras gauche et droit, et sur la jambe gauche formant ainsi le triangle d'Einthoven, elles sont dénommées D1, D2, D3 tel que :

D1 : enregistre les projections des phénomènes électriques entre le poignet droit (pole -) et le poignet gauche (pole +).

D2 : enregistre les projections entre le poignet droit (pole -) et la jambe gauche (pole +).

D3 : enregistre les projections entre le poignet gauche (pole -) et la jambe gauche (pole +).

Figure E.1 : dérivations bipolaires

Dérivations unipolaires

Les dérivations unipolaires ne comportent qu'une électrode active qui explore un territoire localisé du cœur, et l'autre électrode indispensable pour boucler le circuit. **[LETAC 1994]**

Il y a trois dérivations unipolaires des membres, aVR, aVL, aVF. La lettre " a " signifie amplifie, terme qui fut rajouté lorsqu'on découvrit qu'en éliminant une électrode négative l'amplitude de l'enregistrement était augmenté de 50%. " V " signifie vecteur. Les lettres majuscules en indice "R", "L", "F " indiques l'emplacement de l'électrode positive, bras droit « Right », bras gauche « Left » et jambe gauche « Foot ».

Les dérivations unipolaires comparent les potentiels électriques du cœur avec zéro, ce potentiel zéro représente les potentiels électriques au centre du cœur.

L'axe de la dérivation est une ligne imaginaire joignent le point de dérivation au centre du cœur. **[MARY & EDWIN 1977]**

aVR : enregistre les onde du cœur au bras droit.

aVL : enregistre les onde du cœur au bras gauche

aVF : enregistre les onde du cœur au jambe gauche.

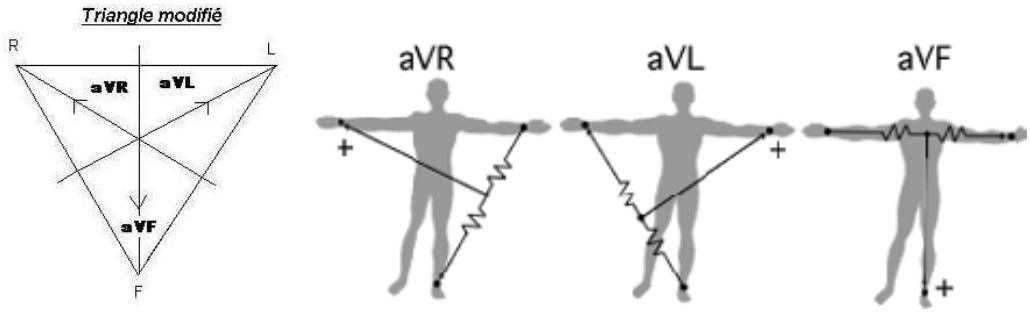
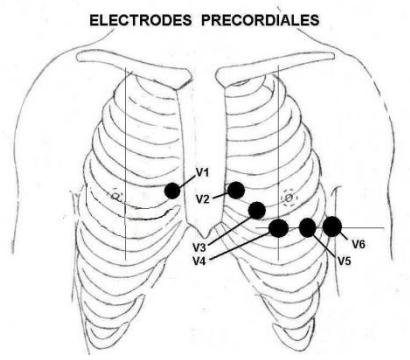


Figure E.2: dérivations unipolaires

2.Les dérivations dans le plan horizontal

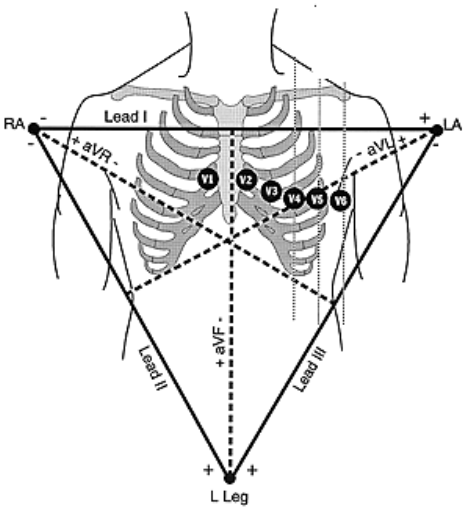
A partir des électrodes placées sur le thorax on enregistre 6 dérivations précordiales unipolaires, c'est-à-dire que chaque dérivation comporte une électrode positive et un point de référence de potentiel zéro (centre du cœur)**.**

Ces dérivations sont dénommées de V1 à V6**,** elles sont progressivement de la droite à la gauche. Noter que les dérivations thoraciques couvrent le cœur dans ces rapports anatomiques avec thorax tel que : la première électrode (V1) est sur la partie interne du quatrième espace intercostal droit, la seconde (V2) sur la partie interne du quatrième espace intercostal gauche. La quatrième électrode (V4) est à la verticale du milieu de la clavicule gauche, sur la sixième côte; la troisième électrode (V3) est à mi –distance entre (V2) et (V4), la cinquième est au même niveau que (V4) mais à la verticale de la ligne axillaire antérieure, la sixième électrode (V6) est au même niveau que (V4) et (V5) mais à la verticale de la ligne axillaire moyenne.



figureE.3 : position des électrodes précordiales

Le système d'électrodes standard utilisant 12 dérivations est finalement constitué de l'ensemble des dérivations : aVR, aVL, aVF, D1, D2, D3 et V1à V6.



<u>figureE4</u> : **les 12 dérivations de l'électrode standard**

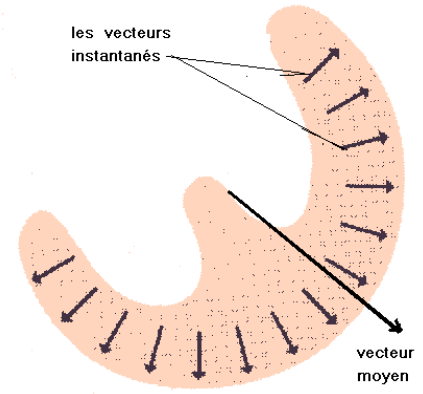
3-Notions vectorielles

Le vecteur est la représentation de la propagation du courant du tissu négatif (dépolarisé) vers le tissu positif (polarisé).

Toutes les régions du cœur ne subissent pas de dépolarisation et repolarisation au même instant, ces deux processus diffusent d'une région à l'autre du cœur pendant chaque cycle cardiaque.

Les vecteurs instantanés résultant de la dépolarisation de la masse musculaire.

La somme de tous ces vecteurs instantanés à pour résultante un vecteur moyen correspond a l'activation globale des oreillettes puis des ventricules.

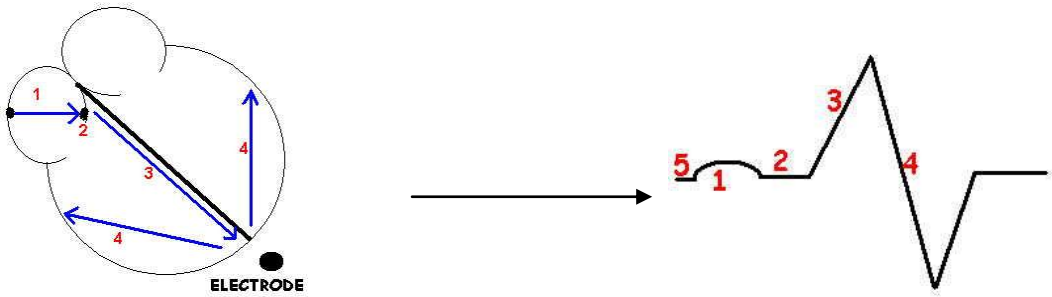


<u>figureE5</u> : **Représentation des vecteurs instantanés**

L'ECG enregistre la vitesse et la direction de ce vecteur.

Un cœur au repos, repolarisé (tout négatif) ou complètement dépolarisé (tout positif) donnera un enregistrement « nul » : le tracé correspondant sera la ligne de base, la ligne isoélectrique.

Une électrode voyant le front positif se rapprocher enregistra un signal positif, et l'inverse, si se front s'éloigne, le signal enregistré est négatif.



<u>figureE 6</u> : technique d'enregistrement

1 : onde positive, le vecteur allant vers l'électrode.

2 : tracé isoélectrique : pas de déplacement de dépolarisation.

3 : onde positive, le vecteur allant vers l'électrode.

4 : onde négative, le vecteur s'éloignant de l'électrode.

5 : tracé isoélectrique : myocarde au repos.

Annexe F

LES BASES DE DONNÉES

La validation de n'importe quel traitement automatisé des signaux ECG, exige son application plus ou moins à un groupe important des signaux réels d'ECG et qui couvriront également le maximum des pathologies des pathologies et des dérivations existantes.

En général, les caractéristiques qui s'imposent à ces bases de données sont :

- Elles contiennent des signaux réels et représentatifs d'une large variation de pathologies cardiaques pour permettre une bonne analyse des signaux d'ECG.
- Elles contiennent des signaux rarement observés mais cliniquement significatifs. Bien qu'il ne soit pas difficile d'obtenir des enregistrements des électrocardiogrammes correspondant à certaines anomalies, souvent plus significatives sont rarement enregistrées.
- Elles contiennent des signaux standards, c'est-à-dire, qu'il faut faire la comparaison des différents algorithmes de traitement des signaux d'ECG sur la même base de données.
- Elles contiennent des signaux annotés. Habituellement, chaque complexe QRS, il a été annoté manuellement par deux cardiologue ou plus, travaillant d'une manière indépendante. Ces annotations servent comme référence pour comparer les résultats produise par les méthodes automatisées.
- Elles contiennent des signaux discrets accessibles aux programmes machines. De cette façon, il est possible d'effectuer un test totalement automatique et reproductible.
- Les paramètre des signaux, comme la fréquence d'échantillonnage (fs) ,.. etc. doivent être connus.

Actuellement, il y a plusieurs bases de données des signaux ECG, donne comme exemple :

- **AHA DB :** the American heart association database for evaluation of ventricular arrhythmia detectors (80enregistrements, de 35 minute chacun)

- **MIT DB :** the Massachusetts institute of technology-Beth Israel hospital arrhythmia database (48 enregistrements de 30minutes chacun).
- **ESC DB :** the European society of cardiology ST-T database (90enregistrements de deux heures chacun).
- **NST DB :** the noise stress test database**.** (12 enregistrements de 30minutes chacun)
- **CU DB :** the Creighton university ventricular sustained arrhythmia database (35 enregistrements de 8minutes à chacun).

Annexe G

la base de données MIT- BHI

MIT, Massachusetts institute f Technology : universitée à Cambridge, une ville des états unis dans le Massachusetts.

En 1865 à boston, l'école ouvrit ses portes et son fondateur fut le géologue WILIAME BARTON ROGER qui devient son premier président. Le MIT fut l'un des premiers établissements à proposer des laboratoires de recherche à développer la profession d'ingénieur chimiste et à offrir des cours en aéronautique et en physique appliquée.

Alors en 1985, un laboratoire MIT a été crée par Nicholas Negroponte et Jerome B.Wiesner, dédié aux technologies de pointes et à leurs applications les plus innovantes.

Depuis 1975, les laboratoires de l'hôpital Bethe Israël à boston et MIT ont réaliser une base de données MIT/BHI qui a été commencée a être distribuée en 1980. Cette base de données contient 48 enregistrements extraits d'une demi-heure des enregistrement ambulatoires à deux voies d'ECG, obtenus à partir de 47 sujets étudies par le laboratoires d'arythmies BIT entre 1975 et1979.

Cette même base est constituée de deux séries, la première série (les séries « 100 » comporte vingt-trois enregistrements ambulatoires de 24 heurs d'ECG rassemblées d'une population mélangée des patients hospitaliers (60%) et les patients non hospitaliers (40%) à l'hôpital de Beth Israël.

Les 25 enregistrements restants ont été choisis parmi les mêmes enregistrements mais qui en considération des arythmies rarement observées ont une signification clinique (les séries des « 200 ». les enregistrements ont été échantillonnés à une fréquence fe=360Hz avec une résolution de 11 bits sur une gamme de 10mv. deux cardiologues ou plus ont indépendamment annoté chaque enregistrement.

Quelques travaux du laboratoire MIT

IBM a développé une souris maligne recouverte de multiples capteurs. Sensibles aux rayons infrarouges ces capteurs peuvent déterminer notre rythme cardiaque en analysant le bout de nos doigts. Une puce thermosensible détermine la température de notre main et des capteurs de mouvement indiquent le type de mouvement effectué (volontaire ou non). Grâce à ces différents capteurs la souris est capable de déterminer si nous ressentons de la colère, de la tristesse, de l'énervement.....

Sur le même principe il existe aussi des manettes de jeux capables de déterminer l'état émotionnel du joueur et d'adapter le jeu en fonction de cet état.

Thad Starner, chercheur en informatique à l'institut de technologie de Géorgie a mis au point en 1993 le wearable computing c'est-à-dire l'informatique qui se Porte. Il s'agit d'une paire de lunettes développée par Micro Optical équipées d'un écran miniature qui lui permet d'accéder à Internet sans que l'on rendre compte.

Doté de 16 couleurs et d'une fréquence de 60Hz, il offre une résolution de 320×240 points.

L'objectif est de rendre la technologie accessible, à tous, de façon permanente et l'intégrer au mieux à l'homme (la rendre "invisible").

Bibliographie

[A.Berro2001]_ Alain Berro . « Optimisation Multi objectif et Stratégie d'évolution en environnement Dynamique » *thèse de Doctorat; Université des sciences sociales ToulouseI*. 18 Décembre 2001

[A.E Eiben1999] Eiben A.E., Hinterding R., & Michalewicz Z. « Parameter Control in Evolutionary Algorithms. *Evolutionary Computation », IEEE Transactions on,* 3(2), 1999.

[A.Wright 1991]. Wright A. « Genetic Algorithms for Real Parameter Optimization. *Foundation of Genetic Algorithms 1 »*. San Mateo: G.J.E Rawlin (Morgan Kaufmann).1991

[Belaid 1992] Belaid,A.,Belaid,Y. « reconnaissance des formes. Méthodes et applicatons », *Inter Edition, 1992.*

[Belgacem, 2002] Belgacem, N, « détection et classification des arythmies cardiaque par application des réseaux de neurones *», these de magister,* université de Tlemcen 2002.

[Bautista et al1999] Maria J. Martin-Bautista, & Maria-Amparo Vila. « A survey of Genetic Feature Selection in Mining Issues. Evolutionary Computation », *CEC, Proceedings of the 1999 Congress on,* 2, 1314-1321.1999

[C. Bontemps 1995] Christophe Bontemps. « Principes Mathématiques et Utilisations des Algorithmes Génétiques » ; 18 Novembre 1995.

[Chikh, 2005] Chikh Mohammed Amine « analyse de signal ECG par les réseaux de neurones et la logique floue : application a la reconnaissance des battements ventriculaires prématurés », *THESE doctorat d'état Es sciences mention Electronique,* université de Tlemcen.2 juillet2005

[C. Igel and M. H□usken.2003] Sylvain Tertois. « Reduction des effets des non-linearites dans une modulation milti porteuse a l'aide de reseaux de neurones ». *PhD thesis, Rennes 1, 2003.*

[D. Martinez & D. Estève.1992], D. Martinez and D. Estève. « The O_set algorithm: building and learning method for multilayer neural networks ». *Europhysics Letters*, 18:95_100, 1992.

[Eshelman L.J1989] Eshelman L.J., Caruana R., & Schaffer J.D. « Biases in the Crossover Landscape ». *Proc. 3rd Int'l Conference on Genetic Algorithms. J.David Shaffer(Morgan Kaufmann Publishing).*1989

[d.meurier2007]. David Meunier. « *Evolutionary supervision of a dynamical neural network allows learning with on-going weights ». Proceedings of International Joint Conference on Neural Networks (IJCNN)* 2005

[Hedeili, 2004] Hedeli,N.« classification des arythmies cardiaques par l'analyse de la composante principale et les réseaux de neurones ».*these de magister* , université de tlemcen2004

[I. Charon, A. Germa, O. Hudry, 1996], « Méthodes d'Optimisation Combinatoire »; 1996.

[Juan-Manuel TORRES-MORENO1997], Torres Moreno and M.B Gordon. « E_cient adaptive learning for classification tasks with binary units . Neural Computation », *In Press, 1997.* « For classifcation tasks with binary units. Neural Computation », *In Press*, 1997.

[Leschi 1991] Leschi C., « Aspects fondamentaux et évolutifs de la reconnaissance de formes. Approche pluridisciplinaire, notion d'approximation », *thèse de doctorat en informatique et automatique appliqué* : INSA de Lyon ,1991

[L.Saadi 2005] L.saadi. « optimisation miltiobjectif, prigramation génétique ».*magister informatique.* Université de batena 2007.

[Man K.F., et al 2000] Man K.F., Tang K.S., & Kwong S. « *Genetic algorithms. Concepts and design »*. Springer.2000

[Man K.F, & Tang K.S. 1997] Man K.F, & Tang K.S. « Genetic Algorithms for Control and Signal Processing. Industrial Electronics, Control and Instrumentation IECON ». *23rd. International Conference on,* 4, 1541-1555.1997

[Mc Donald 1976] Mc Donald CJ. "Protocol-based computers remindes, the quality of care and the non-perfectability of man".*N Engl J med* 1976

[Milgram 1993] Milgram M., « *reconnaissance des formes : méthodes numériques et connexionnistes* » *Armand Collin, paris 1993*

[Miller 1986a] Miller P."Expert critiquing systems/ practice-based medical consultation by computer" .*NEW YORK/ SPRINGER-VERLAG* 1986

[Mitchell M. (1996)]. Mitchell M. « An Introduction to Genetic Algorithms ». *MIT Press.*1996

[M. Karouia, R. Lengellé, & Denoeux T.1995] M. Karouia, R. Lengellé, and Denoeux T. « Performance analysis of a MLP weight initialization algorithm ». In Michel *Verleysen, editor, European Symposium on Artifcial Neural Networks,* Brussels, 1995. D facto.

[Muhlenbein .H1993] Muhlenbein H., & Schlierkamp Voosen D. « Predective Models for the Breeder Genetic Algorithm I. Continuous Parameter Optimization. Evolutionary Computation 1 »,1993.

[N. Benhamed2002] Nadia Benhamed **«** optimisation de réseaux de neurones pour la reconnaissance de chiffres manuscrites : sélection et pondération des primitives par algorithmes génétiques » . *these de doctorat, université de Québec Montréal* , kanada 2002

[N. Dunkin, J. Shawe-Taylor, and P. Koiran.1997] N. Dunkin, J. Shawe-Taylor, and P. Koiran. « A new incremental learning technique ». *In Springer Verlag, editor, Neural Nets Wirn Vietri-96. Proceedings of the 8th Italian Workshop on Neural Nets,* , 1997.

[P.Deg&M.Fie2005] P. De goulet, M. Fieshi. « Informatique médicale », *Masson Edition* ,2005

[S. E. Fahlman and C. Lebiere.1990] S. E. Fahlman and C. Lebiere. « The cascade-correlation learning architecture ». *In D. S. Touretzky, editor, Advances in Neural Information Processing Systems,* , Denver 1989, 1990. Morgan Kaufmann, San Mateo.

[shortliffe1976],shortliffe EH.” Computer-based medical consultation”: *MYCIN.NEW YORK.AMERICAN ELSEVIER*. 1976

[T.Vallée&M.Yudizoglu2001] Thomas vallées&murât yidizglu « présentation des algorithmes génétiques et leurs application en économie ». *Université de Nante* ; septembre2001.

[Y. Shang&B.W. Wha.1996] Y. Shang and B.W. Wha. « A global optimization method for neural networks training. » *In IEEE International Conference on Neural Networks*, Washington, 1996

[Y. LeCun, J. Denker1991], Y. LeCun, J. Denker, S. Solla, R. E. Howard, and L. D. Jackel. « Optimal brain damage ». *In D. S. Touretzky, editor, Advances in Neural Information Processing Systems II, San Mateo,* CA, 1990. Morgan Kaufman.

[Y.OUSSAR 2002] OUSSAR Yacine : “Réseaux d'ondelettes et réseaux de neurones pour la modélisation statique et dynamique de processus” .*Thèse ou HDR soutenue*à ESPCI, 2002

[Z.Michalewicz 1992]. Michalewicz Z. « Genetic Algorithms + Data Structures = Evolution Programs ». *New York: Springer Verlag*.1992

IV.4.4 Apprentissage neuronal

Classifieurs	Se1	Sp1	τ_{class} 1	Se2	Sp2	τ_{class} 2	Se3	Sp3	τ_{class} 3	Se4	Sp4	τ_{class} 4	ccv	e
CLSD1	85.39	96.62	92.25	79.73	94.51	89.96	81.84	90.97	90.57	77.51	96.43	91.54	**81.45**	0.096
CLSD2	87.43	96.30	92.86	81.23	95.31	90.97	54.21	85.44	84.60	61.86	95.65	86.91	**77.45**	0.19
CLSD3	83.00	93.29	89.28	71.41	95.71	88.22	55.52	79.67	78.62	48.48	96.32	83.45	**69.31**	0.78
CLSD4	77.52	92.31	86.55	72.50	84.40	81.08	77.36	76.85	76.87	21.3	48.51	78.56	**61.44**	0.039

tableau4.22 :performances des classifiueurs à 4 classes (apprentrissage classique)

se1,sp1, τclass 1 : les performances de la classe « N »

se2,sp2, τclass 2 : les performances de la classe « V »

se3,sp3, τclass 3 : les performances de la classe «R »

se4,sp4, τclass 4 : les performances de la classe « L »

CCv : taux de classification correcte de classifieur pour toutes les classes

Classifieurs	Se1	Sp1	Cc1	Se2	Sp2	Cc2	Se3	Sp3	Cc3	Se4	Sp4	Cc4	ccv	*e*
CLSD1	84.92	98.94	93.48	86.96	93.13	92.16	85.78	92.46	92.17	80.62	99.27	94.44	**85.40**	0.1388
CLSD2	82.65	98.32	92.21	90.11	92.99	92.10	78.15	87.50	87.09	61.24	98.38	88.78	**79.21**	0.23
CLSD3	85.30	98.83	93.56	92.46	95.13	94.31	64.47	88.58	87.52	64.13	97.78	89.08	**81.12**	0.1891
CLSD44	76.87	98.32	89.96	80.63	88.45	86.04	81.31	93.71	83.60	56.93	97.75	87.19	**73.06**	0.28

tableau4.24 :performances des classifiueurs à 4 classes (apprentrissage génétique)