REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mohamed Khider – BISKRA

Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie

Département d'informatique

N° d'ordre : /M2/2020

Mémoire

Présenté pour obtenir le diplôme de master académique en **Informatique**

Parcours : **Image et vie artificielle**

Entrainement d'un réseau de neurones par algorithme génétique

---

Par :

SOUISSI LAZHARI

TAREK ABABSA	MCB	Rapporteur

Remerciement

Je tiens premièrement à remercier ALLAH le tout puissant de m'avoir donné le courage et la patience pour terminer ce modeste travail.

Je tiens tous d'abord à remercier mon encadreur Dr *«**TAREK ABABSA**» pour sa patience, ses encouragements, ses conseils, et le temps précieux quelle ma accordé pour l'achèvement de ce travaille.*

Mes grands remerciements à tous les enseignants et collègues pour leur accompagnement, aide, encouragement et soutient tout le long de la réalisation de ce travaille.

Dédicace

Je dédie ce mémoire à toute personne qui a su m'encourager à prendre les bonnes décisions, ma mère, mon père, mes frères et sœurs, ainsi que toute ma famille.

Mes Professeurs, par la profonde considération pour l'effort fournis pour un bon enseignement.

Aux chères personnes qui m'en toujours soutenu et accompagné, durent tout mon parcours universitaire, mes Ami(e)s.

Table des matières

chapitre3 : conception et résultat

Liste des figures

Liste des tableaux

Introduction générale

Introduction général

La résolution d'un problème d'optimisation consiste à explorer un espace de recherche afin de maximiser (ou minimiser) une fonction donnée. Les complexités relatives (en taille ou en structure) de l'espace de recherche et de la fonction à maximiser conduisent à utiliser des méthodes de résolution radicalement différentes. En première approximation, on peut dire qu'une méthode déterministe est adaptée à un espace de recherche petit et complexe et qu'un espace de recherche grand nécessite plutôt une méthode de recherche stochastique (algorithmes génétiques….).

Dans la plupart des cas, un problème d'optimisation se divise naturellement en deux phases recherche des solutions admissibles puis recherche de la solution à coût minimal parmi ces dernières.

Les RN peuvent être regardées comme des modèles non linéaires paramétriques possédant des propriétés d'approximation universelle. Les RN ont une inspiration biologique, ils sont composés d'unités simples (neurones) connectés à travers des liens caractérisées par des valeurs numériques (poids synaptique). Pour résoudre un problème à l'aide de RN il est nécessaire de trouver une architecture adéquate du réseau.

Les algorithmes génétiques sont des algorithmes stochastiques adaptés à l'exploration rapide et globale d'un espace de recherche de taille importante en optimisant une fonction quelconque (même de type « boite noire », i.e. aucune hypothèse n'est formulée sur les propriétés de la fonction) et capables de fournir plusieurs solutions de « bonne » qualité. Dans le cas où l'ensemble de solutions générées sont inadmissibles (i.e. il est difficile d'exhiber une solution qui

satisfasse les contraintes du problème). L'admissibilité peut être intrinsèque à la représentation choisie ou intégrée à la génération des individus (initialisation, mutation et croisement) ou à la fonction à optimiser en attribuant une pénalité à un individu qui viole les contraintes

Quand à l'idée d'évaluation des techniques, elle consiste en la combinaison des réseaux de neurones et des algorithmes génétiques, entre eux ou avec d'autres techniques plus traditionnelles, afin d'en marier les avantages respectifs et d'augmenter ainsi l'efficacité des méthodes développées.

Dans notre travail on utilise un mécanisme évolutif pour surmonter la difficulté de l'entrainement des réseaux de neurones.

Le mémoire est organisé en trois chapitres, le premier étudie les réseaux de neurones et leurs caractéristiques, le deuxième étudie les algorithmes génétiques et leurs caractéristiques. Le troisième propose la conception du système et détaille ses composants et quelques résultats expérimentaux.

Chapitre 1 :

Les réseaux de neurones

INTRODUCTION

Les réseaux de neurones (RN) formels sont des systèmes de traitement de l'information dont la structure s'inspire de celle du système nerveux. Leurs deux grands domaines d'application sont d'une part la modélisation biologique, dont il ne sera pas question ici, et d'autre part, la réalisation de machines destinées à effectuer des tâches auxquelles les ordinateurs et les outils traditionnels semblent moins bien adaptés que les êtres vivants, telles que des tâches perceptives et motrices, ainsi, qu'à la reconnaissance de formes, la classification et à l'identification des systèmes.

Nous commençons donc ce chapitre, en premier lieu, par l'historique des RN, nous présenterons ensuite les définitions essentielles, nous expliquons ce qu'est un neurone formel, ce qu'est un réseau de neurones, ce qu'est l'apprentissage des RN (nous précisons notamment les différences entre l'apprentissage supervisé et l'apprentissage non supervisé).

1 HISTORIQUE

L'origine de l'inspiration des réseaux de neurones artificiels remonte à 1890 où W. James, célèbre psychologue américain, introduit le concept de mémoire associative. Il propose ce qui deviendra une loi de fonctionnement pour l'apprentissage des réseaux de neurones, connue plus tard sous le nom de loi de Hebb. Quelques années plus tard, en 1949, J. Mc Culloch et W. Pitts [1] donnent leurs noms à une modélisation du neurone biologique (un neurone automatique comportement binaire). Ce sont les premiers à montrer que des réseaux de neurones formels simples peuvent réaliser des fonctions logiques, arithmétiques et symboliques complexes. C'est ensuite que D. Hebb, physiologiste américain, présente en 1949 les propriétés des neurones par le conditionnement chez l'animal. Ainsi, un conditionnement de type pavlovien tel que, nourrir tous les jours à la même heure un chien, entraîne chez cet animal la sécrétion de salive à cette heure précise même en l'absence de nourriture. La loi de modification des propriétés des connexions entre neurones qu'il propose, explique en partie ce type de résultats expérimentaux. Les premiers succès de cette discipline remontent à 1957, lorsque F. Rosenblatt développe le modèle du Perceptron. Il construit le premier neuro-ordinateur basé sur ce modèle et l'applique au domaine de la reconnaissance des formes. Notons qu'à cette époque les moyens à sa disposition étaient limités et c'était une prouesse technologique que de réussir à faire fonctionner correctement cette machine plus de

quelques minutes. C'est alors qu'en 1960, l'automaticien Widrow développe le modèle Adaline (Adaptative LinearElement). Dans sa structure, le modèle ressemble au Perceptron, cependant la loi d'apprentissage est différente. Celle-ci est à l'origine de l'algorithme de rétro- propagation de gradient très utilisé aujourd'hui 1969 : M.Minsky et S.Papert publient un ouvrage qui met en exergue les limitations théoriques avec les Perceptrons Multi Couches. M.Minsky et S. Papert publient ensuite en 1969 un ouvrage qui met en évidence les limitations théoriques du Perceptron. Ces limitations concernent l'impossibilité de traiter des problèmes non linéaires en utilisant ce modèle. Quelques années d'ombre se sont ensuite succédé de 1967 à 1982.

Le renouveau de cette discipline reprend en 1982 grâce à J. J. Hopfield, un physicien reconnu. Il présente une théorie du fonctionnement et des possibilités des réseaux de neurones. Il faut remarquer la présentation anticonformiste de son article. Alors que les Chapitre II les réseaux de neurones artificiels 28 auteurs s'acharnent jusqu'alors à proposer une structure et une loi d'apprentissage, puis à étudier les propriétés émergentes, J. J. Hopfield fixe préalablement le comportement à atteindre par son modèle et construit, à partir de là la structure et la loi d'apprentissage correspondant au résultat escompté. Ce modèle est aujourd'hui encore très utilisé pour des problèmes d'optimisation. On peut citer encore la machine de Boltzmann en 1983 qui était le premier modèle connu, apte à traiter de manière satisfaisante les limitations recensées dans le cas du Perceptron. Mais l'utilisation pratique s'avère difficile, la convergence de l'algorithme étant extrêmement longue (les temps de calcul sont considérables). C'est ensuite qu'en 1985 la rétro-propagation de gradient apparaît. C'est un algorithme d'apprentissage adapté au Perceptron Multi Couches. Sa découverte est réalisée par trois groupes de chercheurs indépendants. Dès cette découverte, nous avons la possibilité de réaliser une fonction non linéaire d'entrée/sortie sur un réseau, en décomposant cette fonction en une suite d'étapes linéairement séparables. Enfin, en 1989 Moody et Darken exploitent quelques résultats de l'interpolation multi variables pour proposer le Réseau à Fonctions de base Radiales (RFR), connu sous l'appellation anglophone Radial Basis Function network (RBF). Ce type de réseau se distingue des autres types de réseaux de neurones par sa représentation locale [2]

2 NEURONE BIOLOGIQUE

Les neurones, au nombre d'une centaine de milliards, sont les cellules de base du système nerveux central. Chaque neurone reçoit des influx nerveux à travers ses dendrites (récepteurs),

les intègre pour en former un nouvel influx nerveux qu'il transmet à un neurone voisin par le biais de son axone (émetteur) [5] comme le montre la Figure 1

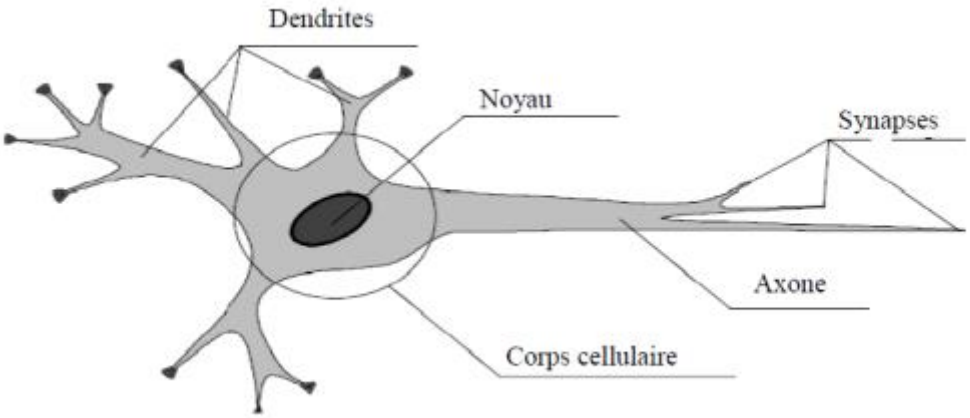


Figure 1: Neurone biologique

A partir de ces concepts on va donner une description des fondamentales du réseau de neurones

2.1 Neurone

Est une cellule vivante, qui peut prendre des formes variables (pyramidale, sphérique ou étoilée) sa forme est définit par une membrane qui sépare l'intérieur du neurone à l'extérieur. Ces neurones sont considérés comme l'élément de base qui constitue les unités élémentaires de traitement dans le cerveau [4].

2.2 Structure de neurone

2.2.1 Le corps cellulaire (soma)

Il est composé d'un noyau qui effectue les transformations biochimiques essentielles à la vie de neurone et se ramifie pour former les dendrites, qui établissent la liaison avec d'autres cellules [5].

2.2.2 Axone

C'est les prolongements unique, qui diffuse le signal du neurone vers d'autres cellules donc son rôle est représenté dans la communication avec d'autre neurone. Il peut diviser a son exterminer pour entrer en contact avec un grand nombre d'autre cellules [5].

2.2.3 Synapse

C'est un élément de jonction qui assure le contact du cytoplasme (membrane) d'un neurone et les membranes de ses voisins, il joue un rôle essentiel dans la transmission des signaux [5].

2.3 Neurone formel

Le neurone formel figure (2) est une modélisation mathématique qui reprend les n principes du fonctionnement du neurone biologique, en particulier la sommation des entrées. Sachant qu'au niveau biologique, les synapses n'ont pas toutes la même « valeur » (les connexions entre les neurones étant plus au moins fortes), les chercheurs ont donc créé un algorithme qui pondère la somme de ses entrées par des poids synaptiques (coefficients de pondération). En général, un neurone formel est un élément de traitement possédant *n* entrées

x1, x2,…, xi,…, xn (qui sont les entrées externes ou les sorties des autres neurones) et une ou plusieurs sorties. Son traitement consiste à effectuer à sa sortie *yi* le résultat d'une fonction de seuillage *f* (dite aussi la fonction d'activation) de la somme pondérée.

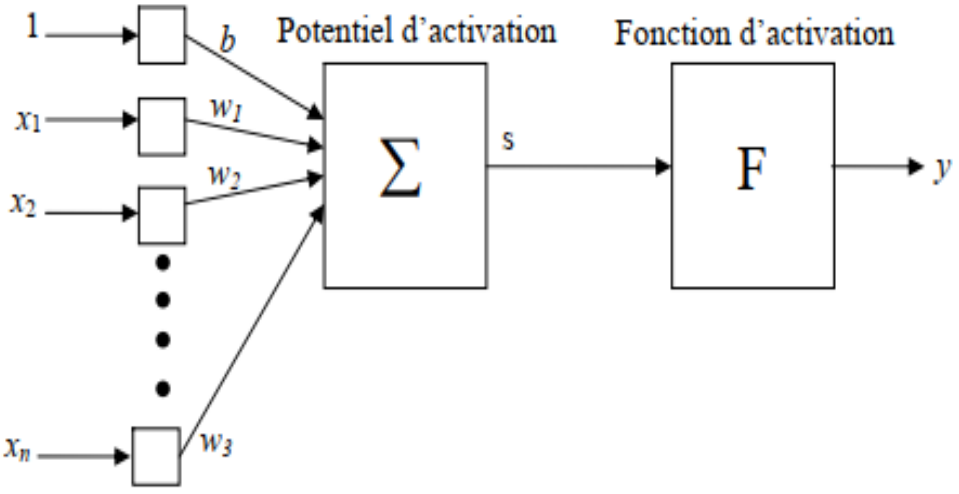


Figure 2: Modèle d'un neurone formel

Avec:

- Les x_i sont les entrées du réseau.
- S est le potentiel d'activation
- Les w_i représentent les poids synaptiques
- y_i la sortie du réseau tels que : $y = f(s)$; $s=\sum_{i=0}^{n} \quad w_i x_i + b$

3 RESEAUX DE NEURONES ARTIFICIELS

3.1 Définition

Un réseau de neurones (RN) est un système d'opérateur non linéaires interconnectés, recevant des signaux de l'extérieur par ses entrées, et délivrant des signaux de sortie, ces (RN) sont une métaphore des structures cérébrales et de traitement parallèle et distribué d'information et comportent plusieurs élément de traitement appelé neurone.

Chaque neurone fonctionne indépendamment des autres de telle sorte que l'ensemble est un système parallèle fortement interconnecte. L'information détenue par le réseau de neurone est distribuée à travers l'ensemble des constituants et non localisée dans une partie de mémoire sous la forme d'un symbole

Le réseau de neurone ne se programme jamais pour réaliser une ou telle tâche, il est entraîné sur des données acquises, grâce à mécanisme d'apprentissage qui agit sur les constituants du réseau afin de réaliser au mieux la tache souhaitée [5].

On peut comparer la correspondance entre les propriétés respectives de neurones biologiques et neurones artificiels comme le montre la Figure 3 [3].

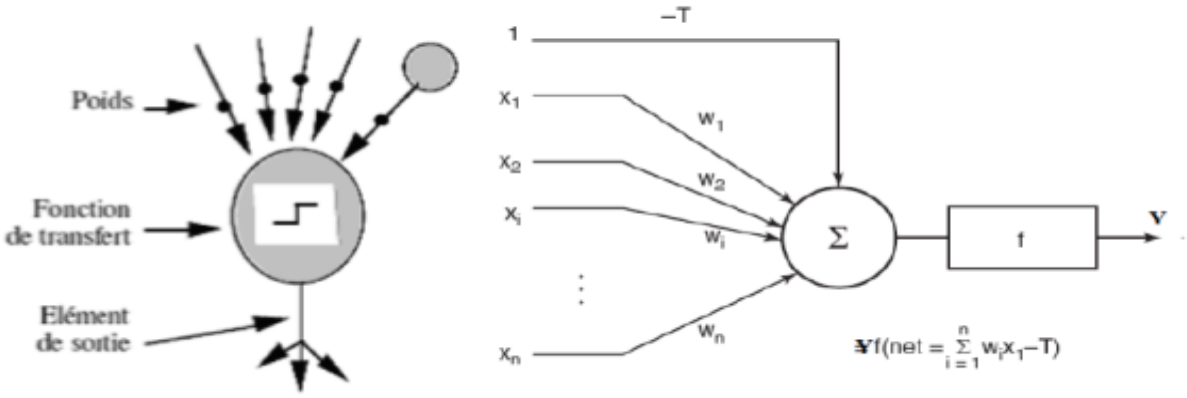


Figure 3: Correspondance entre neurones biologiques et neurones artificiels

Cette comparaison est montrée dans Tableau 1. [6].

système nerveux	réseau neurone Artificiel
Neurone	Traitant élément, nœud, neurone artificiel, neurone abstrait
Le corps cellulaire (soma)	Niveau de l'activation, fonction de l'activation, fonction du transfert, la fonction de la sortie
Axone	la communication avec d'autre neurone
Synapse	poids multiplicatifs

Tableau 1: Comparaison entre neurone Biologique et artificiel

3.2 Domaine d'application des réseaux de neurones (RNA)

Les (RNA) aujourd'hui ont des applications dans des domaines variés parmi lesquelles on cite :

3.2.1 L'application au traitement du signal :

Dans ce domaine les applications réalisées traitent essentiellement de la reconnaissance de signatures radar ou sonner.

La société NESTOR a développé un réseau de neurones qui identifie une cible à coup sûr (100% de réussite) et reconnaît du bruit avec un taux de réussite de 95%. Cette application à été construire à partir de signaux sonars et des traits caractéristiques de ces signaux déterminés par les experts comme utiles pour identifier une cible en environnement bruité. Le réseau agit dans ces comme un filtre pour éliminer le bruit

3.2.2 L'application au contrôle

Grâce à leur propriétés (parallélisme de traitement, capacité d'adaptation, et de généralisation, etc…). Les réseaux de neurones sont appliqués au contrôle intelligent. Par exemple plusieurs tentatives ont été faites pour appliquer les réseaux de neurones ; Ces applications peuvent être classifiés en plusieurs méthode, Telles que : le contrôle supervisé. Le contrôle inverse et le contrôle neuronal adaptatif

3.2.3 L'application au diagnostic

Les réseaux de neurones sont bien adaptés à la résolution des problèmes de diagnostic, utilisant la classification automatique des signaux et des formes. Dans ce contexte on distingue plusieurs applications des réseaux de neurones pour le diagnostic des défaillances et en particulier, pour le diagnostic des pannes des machines électriques. Exemples d'applications de chaque modèle :

Caractéristiques fonctionnelles	Type de RNA

Reconnaissance de formes	*MLP, Hopefield, Kohonen, PNN*
Mémoires associatives	*Hopefield, MLP récurrents, Kohonen*
Optimisation	*Hopefield, ART, CNN*
Approximation de fonctions	*MLP, RBF*
Modélisation et control	*MLP, MLP récurrent, FLN*
Traitement d'images	*CNN, Hopefield*
Classification et clustering	*MLP, Kohonen, RBF, ART, PNN*

Tableau 2: Correspondance RNA - domaines d'application

3.3 Fonctionnement

Avant de pouvoir utiliser les capacités de classification et d'approximation de fonctions d'un réseau de neurones, il faut le construire, ceci se déroule en quatre temps

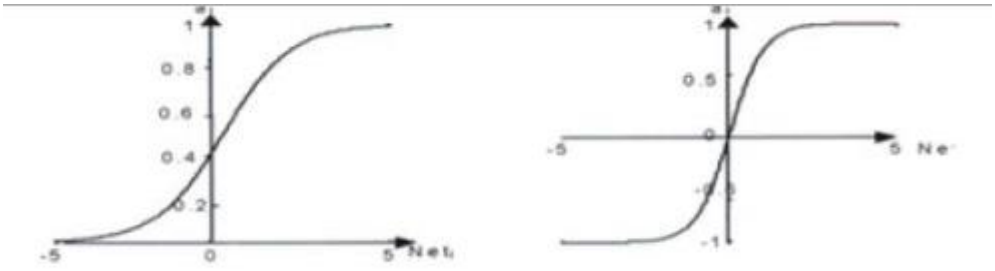
1. La construction de la structure du réseau (généralement empirique)
2. La constitution d'une base de données de vecteurs représentant au mieux le domaine à modéliser. Celle-ci est scindée en deux parties : une partie servant à l'apprentissage du réseau (on parle de base d'apprentissage) et une autre partie aux tests de cet apprentissage (on parle de base de test).
3. Le paramétrage du réseau par apprentissage. Au cours de l'apprentissage, les vecteurs de données de la base d'apprentissage sont présentés séquentiellement et plusieurs fois au réseau. Un algorithme d'apprentissage ajuste le poids du réseau afin que les vecteurs soient correctement appris. L'apprentissage se termine lorsque l'algorithme atteint un état stable.

4. La phase de reconnaissance qui consiste à présenter au réseau chacun des vecteurs de la base de test. La sortie correspondante est calculée en propageant les vecteurs à travers le réseau, La réponse du réseau est lue directement sur les unités de sortie et comparée à la réponse attendue. Une fois que le réseau présente des performances acceptables, il peut être utilisé pour répondre au besoin qui a été à l'origine de sa construction

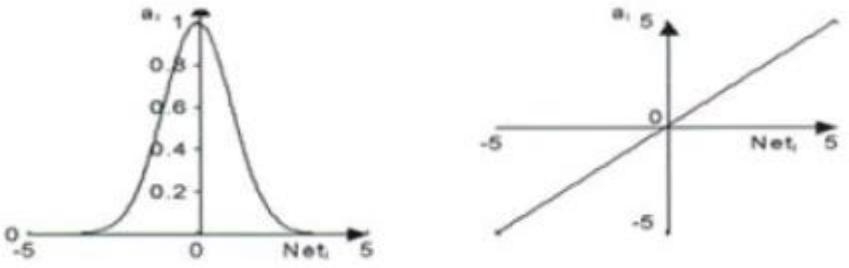
3.4 Modélisation générale

On peut modéliser un réseau de neurone par des élémentaires qu'il s'agit de

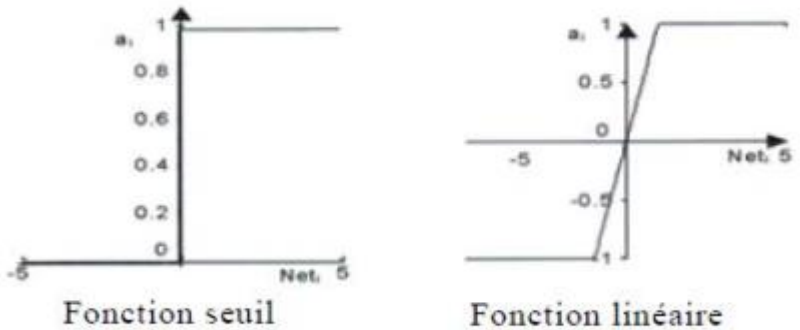
- **La nature de ses entrées :** qu'ils peuvent être binaire (0 ou 1), (-1,1) ou réelles appartenant souvent à intervalle bornée [a, b].
- **La fonction des entrées :** que sa signifie qu'elle peut définir le pré traitement effectuée sur les entrées.
- **Fonction de sortie :** Cette fonction calcule la sortie du neurone en fonction de son état d'activation.
- **Fonction d'activation ou de seuillage :** Il existe de nombreuses formes possibles pour la fonction d'activation. Les plus courantes sont présentées sur la figure 4, la plupart des fonctions d'activations sont continués, offrant une infinité de valeurs possibles comprises dans l'intervalle [0, +1] (ou [-1, +1])



Fonction stochastique (T=1) Fonction sigmoïde



Fonction Fonction linéaire ($\lambda = 1$)



Fonction seuil Fonction linéaire

Figure 4: Différents types de fonctions d'activation pour le neurone formel

Toutes les fonctions d'activation utilisées doivent être différentiables, car l'architectures des réseaux de neurones l'impose pour que l'apprentissage soit possible

3.5 Architecture des réseaux

a- Réseau monocouche

La structure d'un réseau monocouche est telle que des neurones organisés en entrée soient entièrement connectés à d'autres neurones organisés en sortie par une couche modifiable de poids (figure 5)

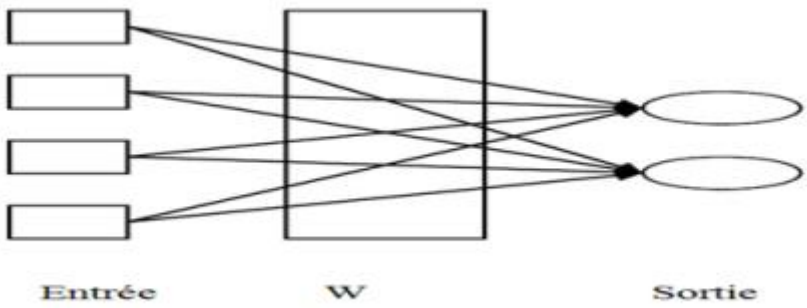


Figure 5: Réseau monocouche

b- Réseau multicouche

Les neurones sont arrangés par couche, les entrée des neurones de la deuxième couche sont en fait les sorties des neurones de la couche amont, les neurones de la

première couche sont reliés au monde extérieur et reçoivent le vecteur d'entrée. Il peut y avoir une ou plusieurs sorties à un réseau de neurone.

Dans un réseau multicouche, il n'y a pas connexion entre neurone d'une même couche et les connexions ne se font qu'avec les neurones de la couche aval, et tous les neurones de la couche amont sont connectés à tous les neurones de la couche aval. On appelle [7]

Couche entrée : contient l'ensemble des neurones d'entrées, cette couche est une

Couche passive, ses neurones n'effectuent aucun traitement

Couche de sorties : contient l'ensemble des neurones de sorties

Couches cachées : les couches intermédiaires n'ayant aucun contact avec l'extérieur.

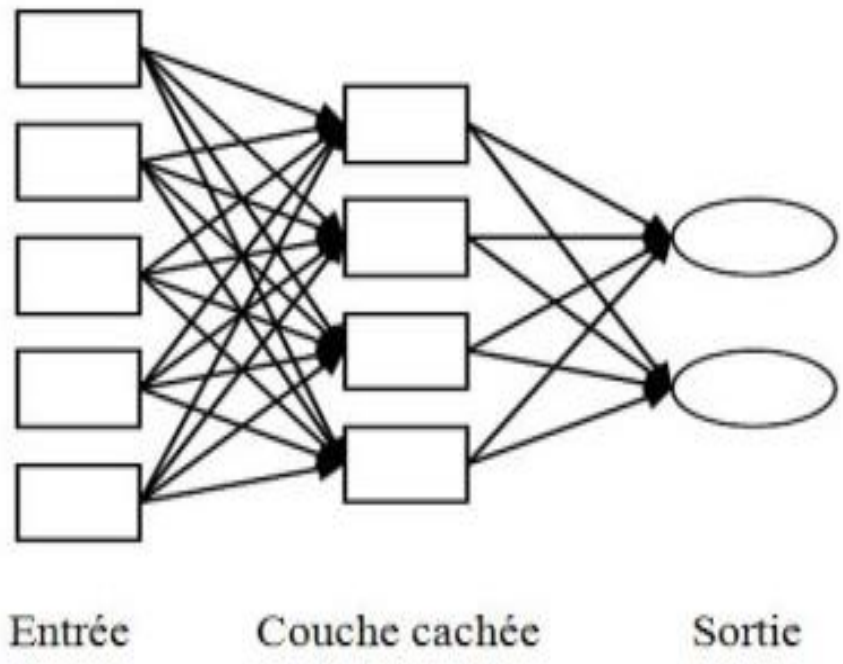


Figure 6: Réseau multicouche

c- Réseau à connexion complète

C'est la structure d'interconnexion la plus générale. Chaque neurone est connecté à tous les neurones du réseau (et à lui-même) (figure 7) [7]

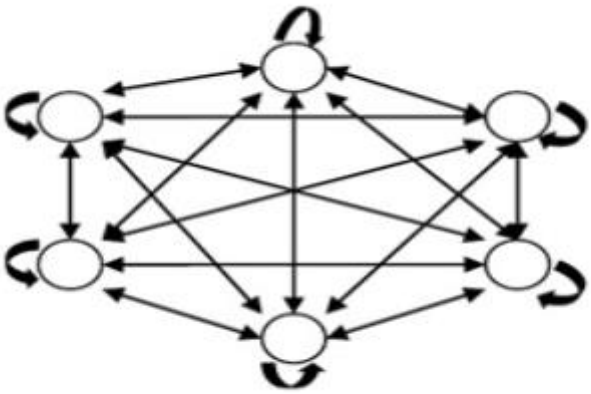


Figure 7: Réseau à connexion complète

d- Réseau à connexions locales

Il s'agit d'une structure multicouche, mais Chaque neurone entretient des relations avec un nombre réduit et localisé de neurones de la couche avale. Les connexions sont donc moins nombreuses que dans le cas d'un réseau multicouche classique (figure 8) [7]

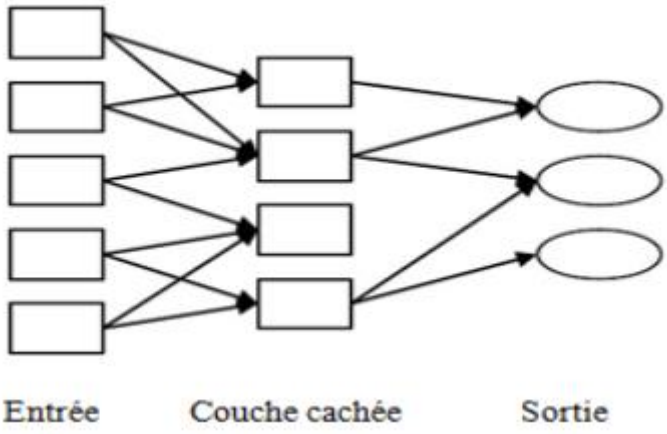


Figure 8: Réseau à connexions locales

e- Les réseaux de neurones boucles (récurrents)

Un réseau de neurone boucle a temps discret réalise une ou plusieurs équations aux différences non linaires, par composition des fonctions réalisées par chacun des neurones et des retards associes à chacune des connexions [8].

Ces réseaux caractérisent par la présence d'au moins une boucle de rétroaction au niveau des neurones ou entre les couches, et la prise en compte de l'aspect temporel du phénomène (figure 9). Mais ce sont des modèles plus durs à mettre en œuvre.

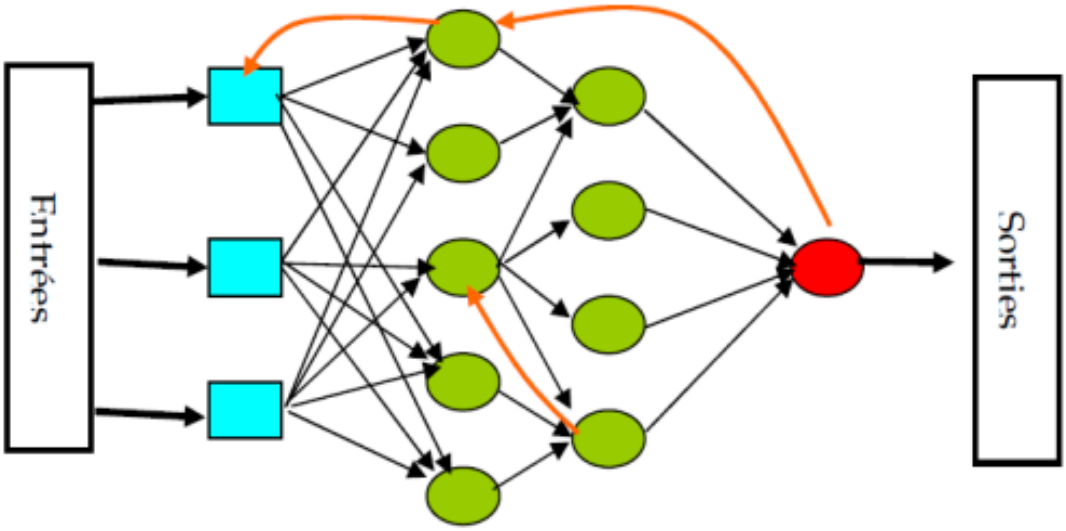


Figure 9: Réseau de neurones boucle

.

4 ALGORITHMES D'APPRENTISSAGE

4.1 Introduction

Toute l'information que peut contenir un réseau neuronal réside dans les poids synaptiques. L'apprentissage consiste donc à ajuster ces derniers de telle façon qu'il puisse générer correctement la sortie correspondante à chaque point de l'espace d'entrée. Ainsi, l'apprentissage peut être défini comme une phase du développement d'un réseau de neurones durant laquelle le comportement du réseau est modifié jusqu'à l'obtention du comportement désiré [9]:

apprentissage ≡ dW/dt

L'une des premières méthodes d'apprentissage a été formulée par Donald Hebb en 1949 pour l'apprentissage des corrélations **[**10], [11].
On peut distinguer deux types d'apprentissage **[**11**]**.

- **Apprentissage supervisé :** un professeur fournit au réseau des couples de données (entrée, sortie désirée correspondante). Les paramètres du réseau sont ajustés de manière à minimiser une certaine norme de l'erreur de sortie constituée par la différence entre la sortie réelle du réseau et la valeur désirée correspondante (fournie par le professeur).
- **Apprentissage non supervisé :** en absence de tout professeur, le réseau organise lui-même les formes d'entrée en classes de façon à minimiser un critère de performances. Ceci peut être fait, par exemple, en désignant un certain nombre de neurones gagnants dans une compétition d'activation ou en désignant un certain nombre de bassins d'attraction dans l'espace d'état [12]

5 CONCLUSION

Les réseaux de neurones artificiels, sont le cœur de plusieurs applications réelles, un outil indispensable qui a fait ses preuves dans la pratique par leurs caractéristiques de généralisation et de robustesses face au bruit qui font rêver les chercher qui veulent avoir le parfait. Alors ces réseaux-là ne sont en fait qu'un :

- Ensemble de neurones formels inspiré du neurone biologique.
- Relié entre eux par des synapses contenant des poids.
- Utilisant un nombre d'algorithmes d'apprentissage afin de réaliser une tache généralement la classification, prédiction, approximation.

Chapitre 2 :

Les algorithmes génétiques

1 HISTORIQUE DES ALGORITHMES GENETIQUE :

En 1860 Charles Darwin expose sa théorie de l'évolution des espèces qui dit que les êtres vivants sont adaptés à leur milieu naturel au travers du processus de reproductions. Et à partir du 20ème siècle la mutation génétique a été mise en évidence pour le traitement de l'information, Les chercheurs en informatique étudient cette méthode pour l'émergence de la programmation évolutionnaire [13].

Dans les années 1960, John Holland à étudier formellement le phénomène d'adaptation tel qu'il se présente dans la nature et de développer des moyens dans lesquels les mécanismes d'adaptation naturelle peuvent être importés dans les systèmes informatiques ce qu'il a aidé à introduire le premier modèle formel des algorithmes génétiques (*the canonical genetic algorithm AGC*) dans son livre : *Adaptation in Natural and Artificial Systems* [14].

L'ouvrage de Goldberg (1989), montre comment appliquer les AG à des problèmes concrets et cela a permis de les populariser et a impulsé de nouveaux travaux. Les AG ont été utilisés avec un succès croissant en optimisation combinatoire, voir par exemple la synthèse de Reeves (1996) et celle en français de Fleurent et Ferland (1996). Parmi d'autres articles de référence, citons par exemple Davis (1991), Dejong et Spears (1989), Grefenstette *et al*. (1985) et Michalewicz (1992).

C'est-à-dire Nous traiterons seulement les AGs fondés sur le Néo-darwinisme l'union de la théorie de l'évolution et de la génétique moderne ; qui s'appuient sur les techniques : croisements, mutation, sélection...

2 LE PRINCIPE DES ALGORITHMES GENETIQUES :

Beaucoup de problèmes informatiques nécessitent un programme d'ordinateur pour être adaptatif afin de continuer à bien performer dans un environnement en mutation. Cela se caractérise par des problèmes de commande de robot dans lequel un robot doit effectuer une tâche une variable d'environnement, et par des interfaces informatiques qui doivent êtres adapter aux particularités des différents utilisateurs.

La Hollande A.G est une méthode permettant de passer d'une population de "chromosomes" (par exemple, des chaînes de zéros et de uns, ou "bits") à une nouvelle population en utilisant une sorte de «sélection naturelle» avec les opérateurs de la génétique d'inspiration croisement, mutation, et inversion. Chaque chromosome est constitué de «gènes» (par exemple, les bits), chaque gène étant un exemple d'un "allèle" particulier (par exemple, 0 ou 1) [13].

L'opérateur de sélection choisit les chromosomes dans la population qui sera autorisé à reproduire, et en moyenne, les chromosomes installateur produisent plus de descendants que ceux qui sont moins en forme. Échanges croisés sous-parties de deux chromosomes, soit environ imitant biologique recombinaison entre deux seul chromosome («haploïdes») organismes ; mutation change de manière aléatoire l'allèle valeurs de certains endroits du chromosome; et l'inversion inverse l'ordre d'une section contiguë du chromosome, ainsi réarranger l'ordre dans lequel les gènes sont disposées

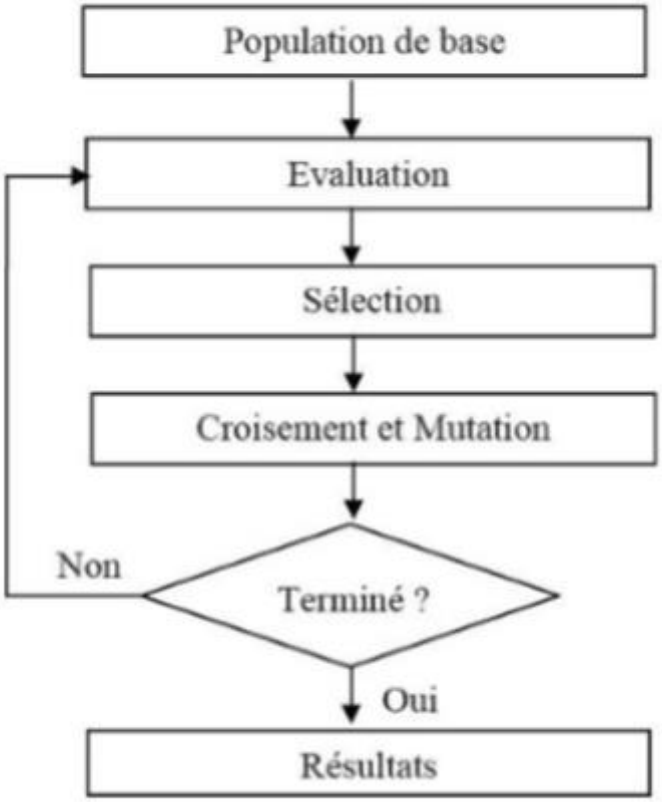


Figure 10: Fonctionnement des AGs

Les algorithmes évolutionnaires (AGs) constituent une approche originale : il ne s'agit pas de trouver une solution analytique exacte, ou une bonne approximation numérique, mais de trouver des solutions satisfaisantes au mieux à différents critères, souvent contradictoires. S'ils ne permettent pas de trouver à coup sûr la solution optimale de l'espace de recherche, du moins peut-on constater que les solutions fournirent sont généralement meilleures que celles obtenues par des méthodes plus classiques, pour un même temps de calcul ; la modélisation de ces phénomènes permet de mieux les comprendre, et ainsi mettre en évidence les mécanismes qui sont à l'origine de la vie d'autre parton peut exploiter ces phénomènes de façon libre et peuvent donc être diverses.

3 LES ETAPES DE L'ALGORITHME GENETIQUE :

Pour utiliser les algorithmes génétiques on doit disposer des éléments suivants :

a) Un principe de codage de l'élément de population. Cette étape associe à chacun des points de l'espace d'état une structure de données après une phase de modélisation mathématique.
b) Un mécanisme de génération de la population initiale Le choix de la population initiale est important car il peut rendre plus ou moins rapide la convergence vers l'optimum global.
c) Une fonction à optimiser appelée fitness ou fonction d'évaluation de l'individu. 4eme Des opérateurs permettant de diversifier la population : L'opérateur de croisement et l'opérateur de mutation 5eme un critère d'arrêt (nombre de générations, probabilités d'application) [13].

Un algorithme génétique doit être à la forme suivante :

1) Initialiser la population initiale P.
2) Evaluer P.
3) TantQue (Pas Convergence) faire :
a) P ' = Sélection des Parents dans P
b) P ' = Appliquer Opérateur de Croisement sur P '
c) P ' = Appliquer Opérateur de Mutation sur P '
d) P = Remplacer les Anciens de P par leurs Descendants de P '
e) Evaluer P
FinTantQue (1)

Le critère de convergence peut être de nature diverse, par exemple :

- Un taux minimum qu'on désire atteindre l'adaptation de la population au problème.
- Un certain temps de calcul à ne pas dépasser.
- Une combinaison de ces deux points.

4 LES OPERATEURS DES A.G :

4.1 Le Codage :

Les paramètres d'une solution est assimilé à un gène et les valeurs qu'il peut prendre sont les allèles de ce gène, on doit coder chaque allèle différent de façon unique.

Un chromosome est une suite de gêne, on peut par exemple choisir de regrouper les paramètres similaires dans un même chromosome et chaque gène sera repérable par sa position : son locus sur le chromosome en question.

Chaque individu est représenté par un ensemble de chromosomes, et une population est un ensemble d'individus.

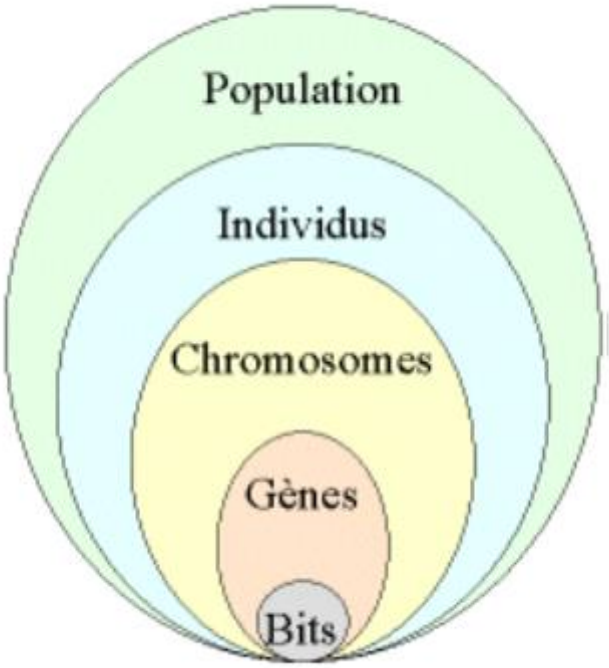


Figure 11: les cinq niveaux d'organisation d'un algorithme génétique [16]

Il y a trois principaux types de codage utilisables :

4.1.1 Le codage binaire :

Le codage binaire est un codage élémentaire dont le principe consiste `a coder la solution selon une chaine de bits. Une chaine de bits est une suite de chiffres, chacun d'entre eux pouvant, prendre la valeur 0 ou 1. La structure de données traditionnellement utilisée est un tableau, appelé aussi vecteur, de variables booléennes. Chaque composante Xj, j = 0, ..., N de ce vecteur est une valeur booléenne prise par la variable. Ce type de codage est le plus utilisé.

4.1.2 Le codage réel (caractères multiples) :

Par opposition au codage binaire, une autre manière de coder les chromosomes d'un algorithme génétique est le codage à l'aide de caractères multiples. Souvent, ce type de codage est plus naturel que le codage binaire.

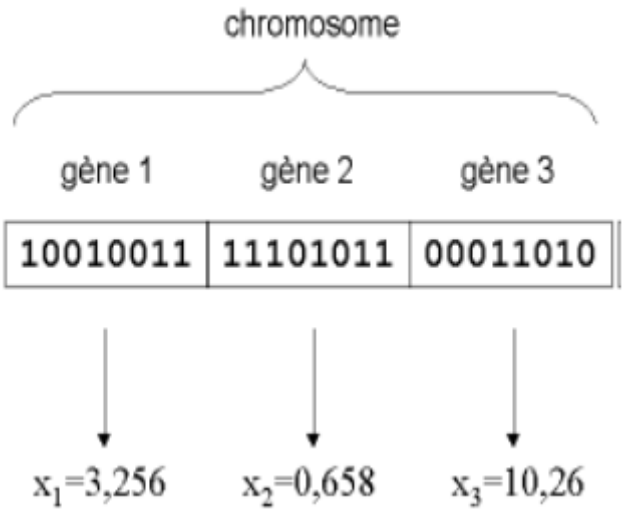


Figure 12: illustration schématique du codage des variables réelles

4.1.3 Codage sous forme d'arbre :

Ce codage utilise une structure arborescente ; un arbre est une structure de données munie d'une racine de laquelle peuvent être issus un ou plusieurs fils. Un de leurs avantages est qu'ils peuvent être utilisés dans le cas de problèmes ou les solutions n'ont pas une taille finie. En principe, des arbres de taille quelconque peuvent être formés par le biais de crossing-over et de mutations [13].

4.2 L'opérateur de mutation :

Cette opération a pour but de créer du désordre dans la population afin de limiter les risques de convergence prématurée vers des optimums locaux. L'opération de mutation est tributaire du codage utilisé et de la probabilité de mutation Pm, qui conditionne la mutation ou non d'un individu. Propose de choisir la probabilité de mutation avec :

$$1/m \leq P \leq 1/m$$

Pour un codage binaire, la mutation consiste à générer un nombre aléatoire compris entre 1 et la longueur du chromosome. La position du code à muter correspond à la valeur du nombre aléatoire généré [14].

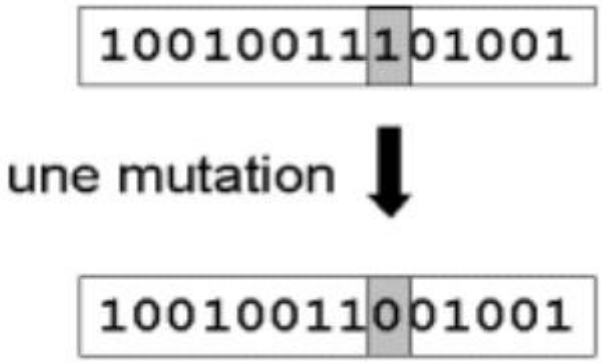


Figure 13: la mutation

4.3 L'opération de Croisement ou Crossover :

La procédure des élections permet d'identifier les individus à reproduire, les parents. Le mécanisme de reproduction est assuré par les opérations de croisement et de mutation. L'opération de croisement doit permettre d'améliorer la performance de la population considérée et de générer de meilleures solutions, les enfants. L'opération de croisement est tributaire de la nature du codage utilisé. Elle est effectuée sur une paire d'individus sélectionnés de la population, les parents.

4.3.1 Croisement simple :

Le croisement entre deux individus est conditionné par la probabilité de croisement et le nombre de points de croisement Le croisement permet de générer deux nouveaux individus dont la structure a été modifiée à partir d'individus de la population. Pour le codage binaire, il s'agit d'abord de déterminer le point de croisement. Chaque chromosome est caractérisé par une longueur déterminée par le nombre de code le constituant. En fonction du nombre de points de croisement voulu, on génère un ou plusieurs nombres aléatoires comprise entre 1 et la longueur du chromosome considéré.

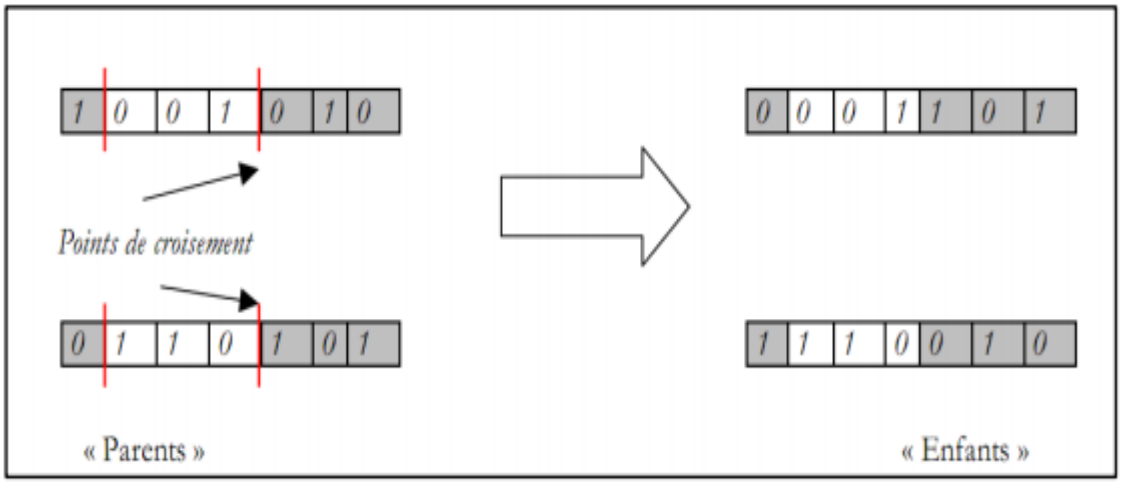


Figure 14: Procédure de croisement dans le cas d'un codage binaire

4.3.2 Croisement uniforme :

Cette opération consiste à considérer pour un chromosome donné un masque formé d'un vecteur aléatoire binaire. Le codage utilisé représente une variable de décision lui correspondant un nombre binaire, réel, entier ou alphabétique. En fonction de la valeur de la composante du vecteur binaire associé à la position sur le chromosome, la valeur du code associé à cette position peut avoir l'une des deux valeurs des parents considérés. Le deuxième chromosome (enfant) généré sera constitué par symétrie par rapport au premier chromosome. En fonction de la position du code, si la valeur est associée au premier parent, alors la valeur de la même position pour le second enfant sera associée au deuxième parent.

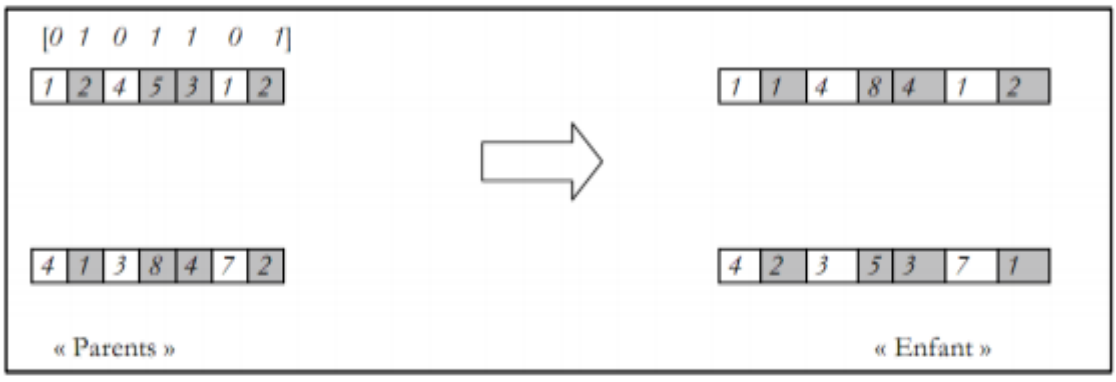


Figure 15: Procédure de croisement uniforme dans le cas d'un codage entier

4.4 L'opérateur de sélection :

Cet opérateur est important puisqu'il permet aux individus d'une population de survivre, de Se reproduire ou de mourir. Il existe plusieurs techniques de sélection :

4.4.1 La sélection par tournois :

On effectue un tirage avec remise de deux individus de P, et on le fait "combattre".

Celui qui a la fitness la plus élevée l'emporte avec une probabilité p comprise entre 0.5 et 1. On répète ce processus n fois de manière à obtenir les n individus de P' qui serviront de parents.

4.4.2 La méthode élitiste :

Cette méthode consiste à sélectionner les n individus dont on a besoin pour la nouvelle génération P' en prenant les n meilleurs individus de la population P après l'avoir triée de manière décroissante selon la fitness de ses individus0

4.4.3 La sélection par roulette :

Cette méthode est la plus connue et la plus utilisée. Elle consiste à dupliquer chaque individu de la population proportionnellement à son milieu. Ainsi, les individus ayant la plus grande valeur de fitness auront plus de chance d'être choisis. Chaque individu de la population occupe une section de la roue proportionnellement à son adaptation et qui indique aléatoirement quel individu peut se reproduire [15].

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

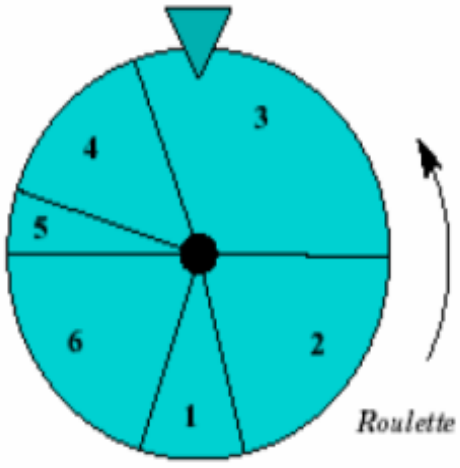


Figure 16: Exemple de sélection par roulette

4.5 La fonction fitness (la fonction d'évaluation)

Holland a définit la fonction fitness par : f̂ où f est la fonction d'évaluation de population.

Pour calculer le coût d'un point de l'espace de recherche, on utilise une fonction d'évaluation.

L'évaluation d'un individu ne dépende pas de celle des autres individus, le résultat fournit par la fonction d'évaluation va permettre de sélectionner ou de refuser un individu pour ne garder que les individus ayant le meilleur coût en fonction de la population courante : c'est le rôle de la fonction *fitness*.

Cette méthode permet de s'assurer que les individus performants seront conservés, alors que les individus peu adaptés seront progressivement éliminés de la population.

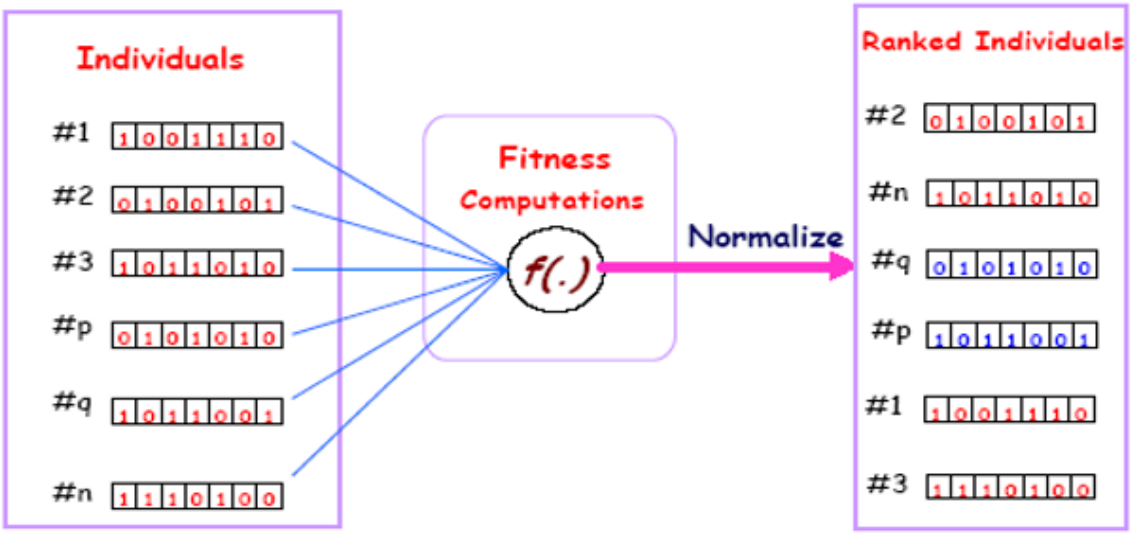


Figure 17: La fonction fitness

5 LE CRITERE D'ARRET DE L'ALGORITHME :

L'arrêt de la progression d'un algorithme génétique n'est pas souvent difficile de savoir si on a trouvé Actuellement les critères les plus.

- Arrêt de l'algorithme après un certain nombre de génération.
- Arrêt de l'algorithme lorsque le meilleur individu n'a pas été amélioré depuis certain nombre de génération.
- Arrêt de l'algorithme lorsqu'il y a perte de diversité génétique

6 CONCLUSION

Nous avons présenté dans ce chapitre les principes de base d'un outil de résolution qui a fait ses preuves en intelligence artificielle (IA), à savoir l'algorithme génétique. Nous retenons principalement qu'un algorithme génétique est une reproduction artificielle de mécanismes naturels liés à la génétique. Son principe, basé sur une description fidèle d'une évaluation naturelle, est un processus cyclique manipulant une population et assurant une recherche efficace dans un espace de solutions candidates d'un problème donné, par application des trois opérateurs qui sont la sélection, le croisement et la mutation, dans le but d'optimiser la fonction d'adaptation associée, définie sur un espace éventuellement complexe.

Chapitre 3

Conception et résultat

1 INTRODUCTION

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'évolution*. 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

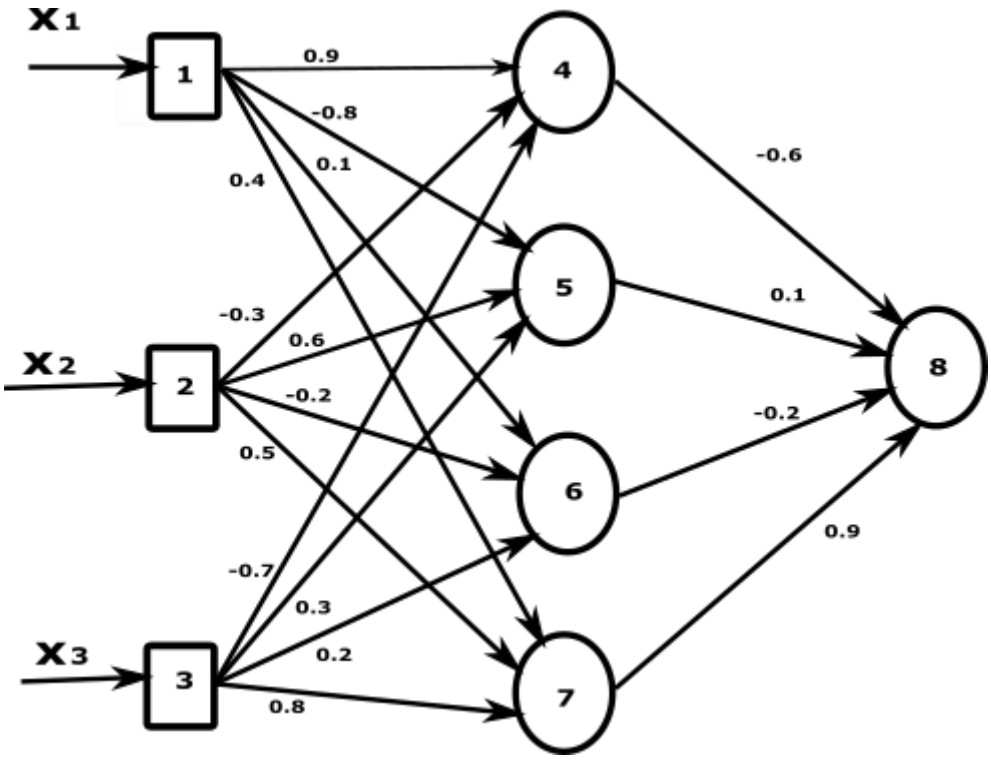
Par analogie avec l'évolution naturelle, les AG font évoluer un ensemble de solutions candidates, appelé une « population d'individus ». Un « individu » n'est autre qu'une solution possible du problème à résoudre. Chaque individu de cette population se voit attribuer une fonction appelée fonction d'adaptation (*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.

2 UTILISATION DES AG POUR UNE OPTIMISATION DES POIDS (APPRENTISSAGE PARAMETRIQUE)

La partie suivante présentent le concept de base d'une technique d'optimisation de poids génétique (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 18.



Chromosome :

0.9	**-0.3**	**-0.7**	**-0.8**	**0.6**	**0.3**	**0.1**	**-0.2**	**0.2**	**0.4**	**0.5**	**0.8**	**-0.6**	**0.1**	**-0.2**	**0.9**

Figure 18: 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 19 montres une application de l'opérateur de croisement.

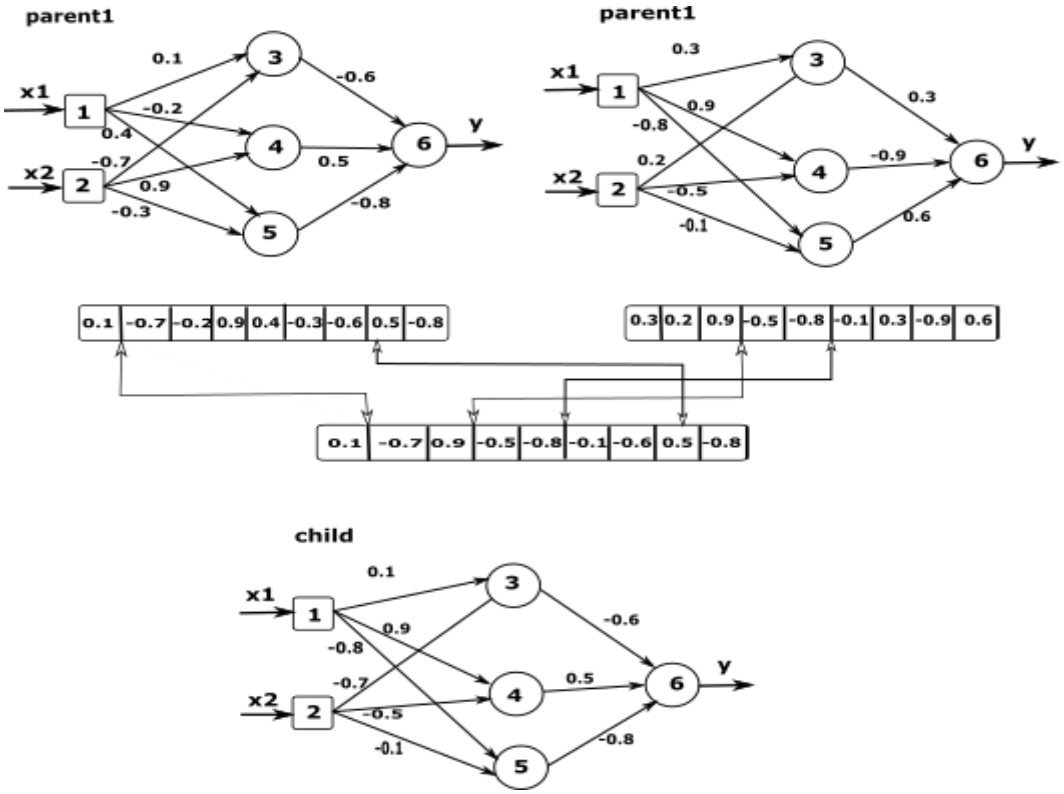


Figure 19: opérateur de croisement dans un système neuro-génétique

Un opérateur de mutation choisit aléatoirement un gène dans un chromosome et ajoute une petite valeur aléatoire à chaque poids dans ce gène. Figure 20 montres un exemple de mutation

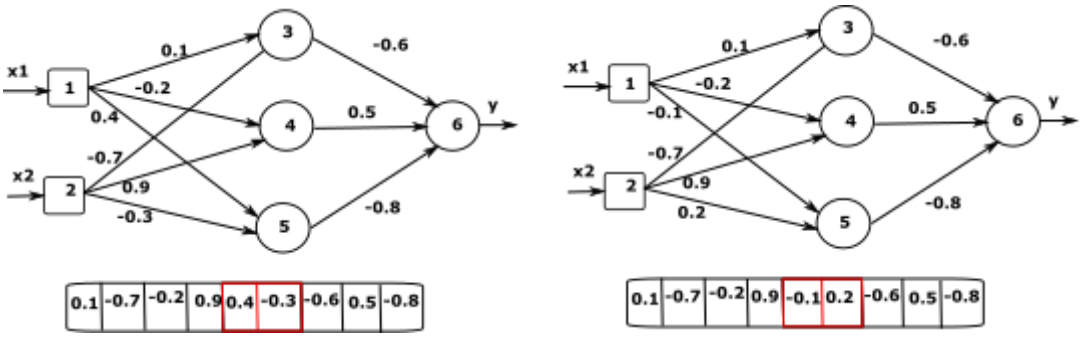


Figure 20: opérateur de mutation dans le système neuro-génétique

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.

3 UTILISATION DES AG 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 y a 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 directe limité et peut être appliqué seulement aux réseaux non récurrents avec un nombre fixé de neurones, et on code les connexions entre les neurones pour les évaluer

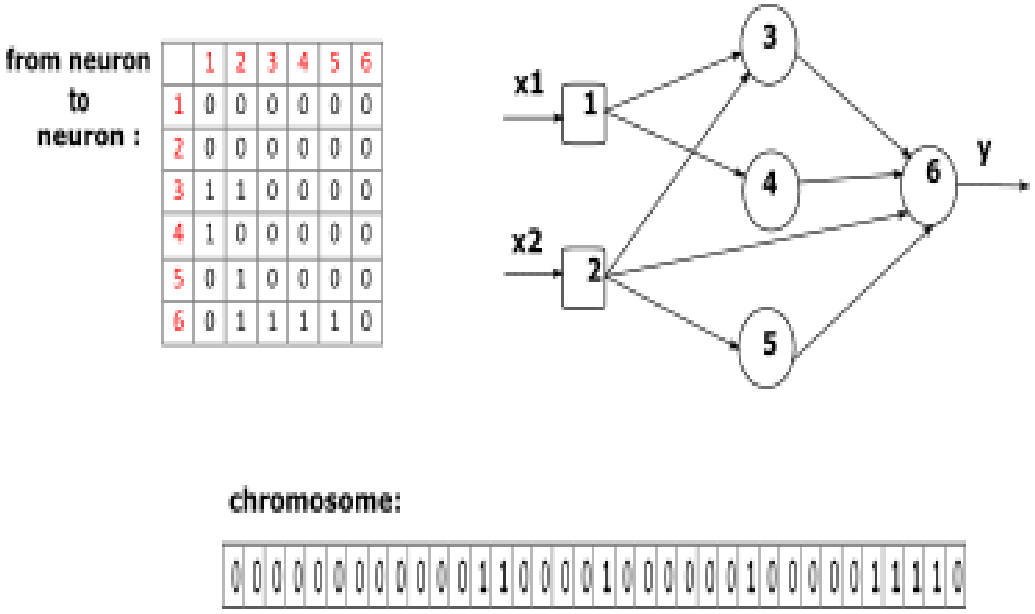


Figure 21: système neuro-génétique pour le choix de la topologie

La topologie de connexion d'un réseau neuronal peut être représentée par une matrice de connectivité carrée, comme indiqué dans la figure 21. 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 figure 4

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 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.

Etape 6 : 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.

Etape7 : placez les chromosomes créés dans la nouvelle population.

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

4 CONCEPTION DETAILLE

4.1 Les bases de données :

Une base de données est un ensemble des données personnalisé pour un domaine particulier. Dans notre cas, on traite des tables des données avec des attributs de type numérique pour simplifier les calculs.

4.2 Module d'évolutif avec les algorithmes génétiques :

4.2.1 Génération de population initiale :

On commence par construire une population initiale composée d'un ensemble de règles sous la forme suivante :

Initialisation de population consiste à initialiser la population par les individus.

```
Init_population ()
début
vector tab_indv ;
pour i=0 a vec_size()
System.out.print(" "+ tab_indv  [i]);
Fin pour
fin
```

La taille de la population initiale est un paramètre très important dans le système.

4.2.2 Fonction de fitness :

La fonction de fitness se change d'un algorithme génétique à un autre selon le problème traité. Dans notre cas la fonction de fitness sera assez simple définie par 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 de la forme :

$$Err = \sqrt{\sum_{i=j=1}^{N} (y_i - y_j)^2}$$

Avec :

- y_i : Sortie désirée.
- y_j : sortie obtenue.

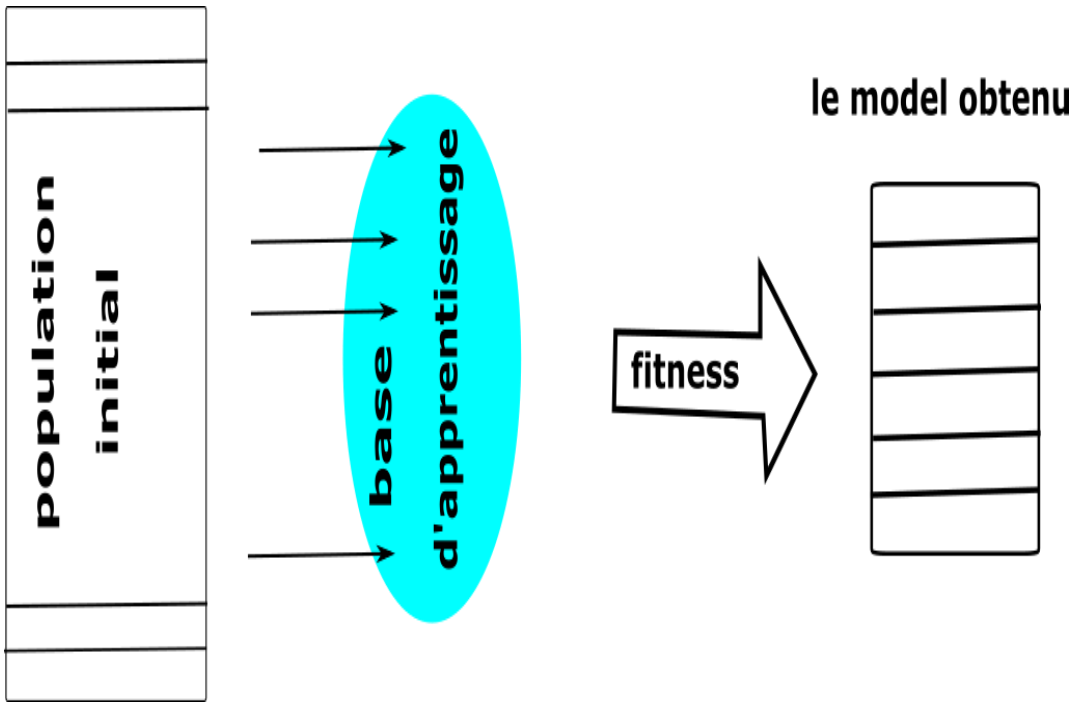


Figure 22: Fonction de fitness

Le but est d'obtenir y_j le plus proche possible de y_i, pour tout i = 1, .. ., N. Pour savoir si l'objectif est atteint, on mesure l'écart entre ces valeurs.

4.2.3 Générer de nouvelles populations :

Si la condition d'arrêt est n'est pas satisfaite par le modèle obtenu, on génère une nouvelle population en utilisent les mutations et les croisements à partir les règle du modèle

4.2.4 Méthode de croisement :

Le croisement a plusieurs méthodes (1 point ; 2 points ; plusieurs points).

```
Début
 -choisir un point de découpe de manier aléatoire X
Pour i=0 a X faire
  Fils1.add (indiv1.get (i));
  Fils2.add (indiv2.get (i));
Fin Pour
```

```
Pour i=X a indiv1.size () faire
  Fils1.add (indiv1.get (i));
  Fils2.add (indiv2.get (i));
Fin Pour
-Deux fils sont créés pour la mutation
Fin
```

4.2.5 Méthode de mutation :

La mutation consiste à choisir un attribut pour un individu choisi d'une manière aléatoire est changer sa valeur.

```
Début
-sélectionner les bits a muté x aléatoirement dans chaque individu enfant
-échanger ces bits de cet enfant par 0 ou 1
Si (fils.get(x).equal(0))
   Fils.set(x, 1) ;
Sinon
   Fils.set(x, 0) ;
Fin si ;
Fin ;
```

5 STRUCTURE DU RESEAU NEURONE :

Le réseau neurone proposé est une architecture à trois couches, nous considérons cette fois ci un cas plus général avec un réseau à *deux* entrées [x_1, x_2] et à une sortie *y*. En considérant la figure 23, explicitons le fonctionnement du réseau, couche par couche.

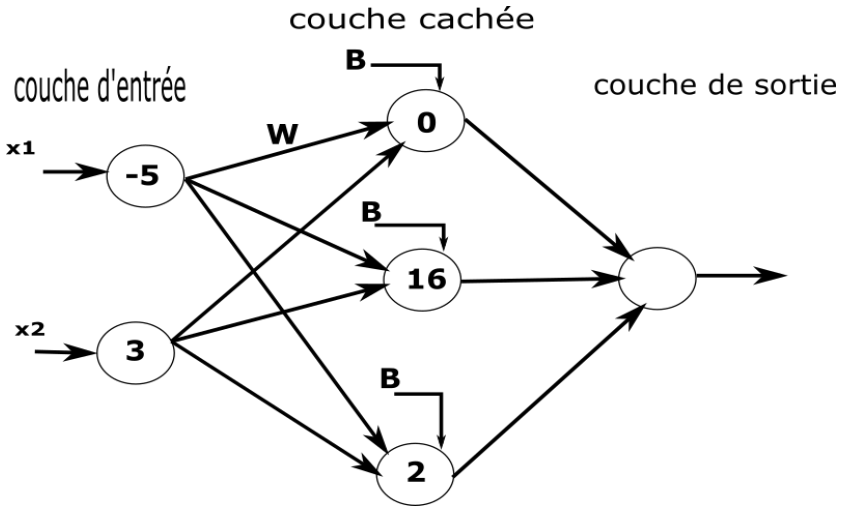


Figure 23: Exemple d'un réseau à une seule couche caché

Couche d'entrée **:** Aucune fonction n'est réalisée avec les neurones de cette couche. Chaque neurone transmet le signal d'entrée vers la deuxième couche. Avec 2 entrées noté X

Couche cachée : une seule couche cachée est composé de 3 neurones.

La couche de sortie **:** la sortie est composée d'un seul neurone.

- Codage des chromosomes (Représentation des solutions)

Soit w^k_{ij} le poids d'une connexion du neurone *i* vers le neurone *j* de la couche *k.* sa représentation est :

$$C^k_j = \sum_{i=j=1}^{N} x_i w_{ij} + b_j$$

Où

w_{ij} est le poids de connexion entre le neurone « *i* » de la couche d'entrée et le neurone « *j* » de la couche cachée.

b est le biais.

k spécifie la couche cachée.

Chaque couche est la concaténation des neurones de la couche correspondante. Chaque neurone est représenté par la concaténation des poids de ses entrées. Le réseau est alors entièrement codé dans les chromosomes

- calcul la couche de sortie, la sortie du neurone est :

Notre réseau définit une fonction par exemple X→ f(x)= x^2. L'entrée correspond donc à un réel et la sortie également

On a y=f(c)=f(x)

Avec $c^{\Theta}= \sqrt{\sum_{j=1}^{N} f(c)}$

Où « Θ » spécifie la couche de sortie

- calcul la Fonction de fitness (**fonctions d'erreurs**)

Il existe différentes formules pour calculer l'erreur entre Sortie désirée y_i et sortie obtenue y_j=*F* (*xi*).
On considère une série de valeurs *yi*, *i* = 1, ., *N* (fournie par observations ou expérimentations) qui sont approchées par des valeurs y_j produites par une formule issue d'un réseau de neurones.

L'algorithme consiste à ajuster les poids de connexion en minimisant la fonction coût suivant :

$$Err = \sqrt{\sum_{i=1}^{m} (y_i - y_j)^2}$$

Où Err l'erreur quadratique moyenne du réseau

6 IMPLEMENTATION DE L'ALGORITHME :

Après initialisation des poids et des biais :

1) Appliquer le vecteur d'entrée $X = (x1,...,xn)$ aux neurones d'entrée.

2) Calculer les valeurs de *c* pour les neurones de la couche cachée à par de l'équation

$$C^k_j = \sum_{i=j=1}^{N} x_i w_{ij} + b_j$$

3) Calculer les valeurs de v pour la couche de sortie à par de l'équation

$$c^o = \sqrt{\sum_{j=1}^{N} f(c)}$$

4) Calculer les sorties à par de l'équation

$$y=f(c)=f(x)$$

5) Calculer les termes d'erreur à par de l'équation

$$Err = \sqrt{\sum_{i=1}^{m} (y_i - y_j)^2}$$

A chaque fois, on présente au réseau un vecteur d'entraînement d'entrée et on refait les étapes de **2** jusqu'à **5**.

6) On refait le calcul jusqu'à ce que la fonction coût Err soit plus petite que possible.

7 LES STRUCTURES DES DONNEES UTILISEES

Pour la manipulation des données, nous avons choisi une structure qui convient mieux au langage de programmation et la nature des données utilisées.

Le genom (individu**) *est*** représenté par de déférents types de champs :

- *inputsize* contient le size de couche d'entrée
- Le champ *hiddensize* contient le size de couche caché.
- Le champ *outputsize* contient le size de couche sortie.

La population des genoms est représentée par de déférents types de champs :

- Le champ *pop_size* contient le size de population.
- Le champ *fitness*

Les poids est représenté par un tableau d'une seule dimension contient :

- Le champ *Linearweights*

8 LES ALGORITHMES

La réalisation de ce travail est faite à l'aide de l'évolution de l'algorithme génétique (AG)

8.1 Algorithmes génétique(AG)

L'implémentation de cet algorithme nécessite un ensemble d'étapes.

Algorithme 1 : Algorithme génétique ()

Début

```
Population (mutationRate, rnd, pop_size)
tournament_selection (k)
Crossover (rate)
Mutation ()
```

Fin

8.1.1 Génération d'une population initiale

La première étape de cet algorithme est générer aléatoirement une population initiale.

Algorithme 2 :

Population (mutationRate,rnd,pop_size)
mutationrate: taux de mutation

Pop_size :taille de population

Début
Pour i=0, **pop_size** ***faire***

Générer genom aléatoire

P(i) ← genom

Fin pour.

Fin

8.1.2 Calcule de fitness

La fitness se calcule à partir de la matrice des entrées nommée *'sample'* .Dans la figure 1, (une partie de la matrice des entrées)

```
{0.020115902388317508,0.38800240581774736}
{0.9799941939226933,0.35715272800707243},
{0.6509294213884272,0.09213154422652192},
{0.025111196340401265,0.8423915834708964},
{0.6653501758343596,0.995961158103732l},
{0.6403787791900267,0.622480883067237},
{0.23151772717659624,0.6752618369988852},
{0.649944665119086,0.9166152469482595},
{0.49143015377422894,0.16182033894177106},
{0.08083954420450246,0.21900735777797808},
{0.6486977564891027,0.2445056130492156},
{0.4415556972516581,0.14854028389529916},
{0.22979395201854858,0.14037283862249028},
{0.15198261613938224,0.6640176912178855},
{0.6632429819911366,0.4675989253365985},
{0.3536110852884319,0.6964338594684572},
{0.21669217801746643,0.9803624878597298},
```

Figure 24: Matrice des entrées ('sample')

Le calcul de la fitness est fait selon l'algorithme suivant

Algorithme 1 : Evaluate ()

Mutationrate : taux de mutation

Pop_size : taille de population

Début

Fitness ← 0;

Pour i=1, 100 ***faire***

Fitness ← fitness+ Math.*pow* (Calculate (input) – function (input [0], input [1]), 2)

Fin pour.

Fitness=Math.*sqrt* (fiss)

Fin

Exemple :

La figure suivante présente une population

```
----------------------------------
num    fitness           size
1:  2.7882996626029493 : 200
----------------------------------
----------------------------------
num    fitness           size
2:  2.661433757063705 : 200
----------------------------------
----------------------------------
num    fitness           size
3:  2.439862000229935 : 200
----------------------------------
----------------------------------
num    fitness           size
4:  2.127961751186039 : 200
----------------------------------
----------------------------------
num    fitness           size
5:  1.9822188139570702 : 200
----------------------------------
----------------------------------
num    fitness           size
6:  1.6965941158169715 : 200
----------------------------------
```

Figure 25: Exemple illustratif d'une population

8.1.3 Sélection

La sélection est réalisée selon l'algorithme de sélection par tournoi

Pour choisir les parents de reproduction.

8.1.4 Croisement

L'opérateur de croisement est montré dans l'algorithme suivant :

Algorithme 3 : Crossover (index1, index2)

newweights1, newweights2 : deux table de poids avec taille = 12

Début

Pour i=0, 12 ***faire***

Si (rnd.nextBoolean ()***)***

newweights1 (i) = genomes.get (index2).Linearweights (i);
newweights2 (i) = genomes.get (index1).Linearweights (i);

Sinon

newweights1 (i) = genomes.get (index1).Linearweights (i);
newweights2 (i) = genomes.get (index2).Linearweights (i);

Fin pour.

Genom gnm1 = **new** Genom (2, 4, 1, newweights1, rnd);

Genom gnm2 = **new** Genom (2, 4, 1, newweights2, rnd);

Fin

8.1.5 Mutation

Nous avant appliqués la mutation, qui se base sur l'algorithme suivant :

Algorithme 3 : Mutate (mutationRate)

Début

Pour index=0, 12 ***faire***

Si (rnd.nextBoolean ()< mutationRate***)***

Si (rnd.nextBoolean ()***)***

Linearweights (index) = Linearweights (index) + rnd.nextDouble ()
Sinon
Linearweights (index) = Linearweights (index) - rnd.nextDouble ()

Fin pour

Fin

9 RESULTATS

L'évolution de la performance de notre réseau neurone, se base sur une fonction d'évolution (fitness) pour évoluer la performance des genomes. , les AG font évoluer un ensemble de solutions candidates, appelé une « population de genom ». Chaque genom de cette population se voit attribuer une fonction appelée fonction d'adaptation (fitness) qui permet de mesurer sa poids cette fonction d'adaptation représenter la fonction objectif à optimiser. Ensuite, les meilleurs genomes de cette population sont sélectionnés, subissent des croisements et des mutations et une nouvelle population de solutions est produite pour la génération suivante. Enfin on choisi la meilleure poid à partir d'un meilleure fitness.

Exemple

La figure suivantes présentent le résultat d'un meilleure poids, avec nombre de génération =9 et size population=200.

```
--------------------------------
num    fitness          size
1:  6.601284339421626 : 200
--------------------------------
--------------------------------
num    fitness          size
2:  6.35815284794724 : 200
--------------------------------
--------------------------------
num    fitness          size
3:  6.22895705715038 : 200
--------------------------------
--------------------------------
num    fitness          size
4:  5.904312482867444 : 200
--------------------------------
--------------------------------
num    fitness          size
5:  5.759660402593108 : 200
--------------------------------
--------------------------------
num    fitness          size
6:  5.524244908521504 : 200
--------------------------------
--------------------------------
num    fitness          size
7:  5.405039545958749 : 200
--------------------------------
--------------------------------
num    fitness          size
8:  5.226836861516526 : 200
--------------------------------
--------------------------------
num    fitness          size
9:  5.067464437927224 : 200
---------------*
fitnesse: 5.067464437927224
0.8593186221222772,0.7809909606256609,0.8220329911029707,0.02711019053605168,0.83940557892591,0.6954117969247898,0.5834797310703045,0.03495263304163532,
```

Figure 26: Résultat d'un meilleure fitness avec sa poid

10 COURBE FITNESS

Dans cette figure est pour dessiner la courbe de meilleure fitness de chaque génération

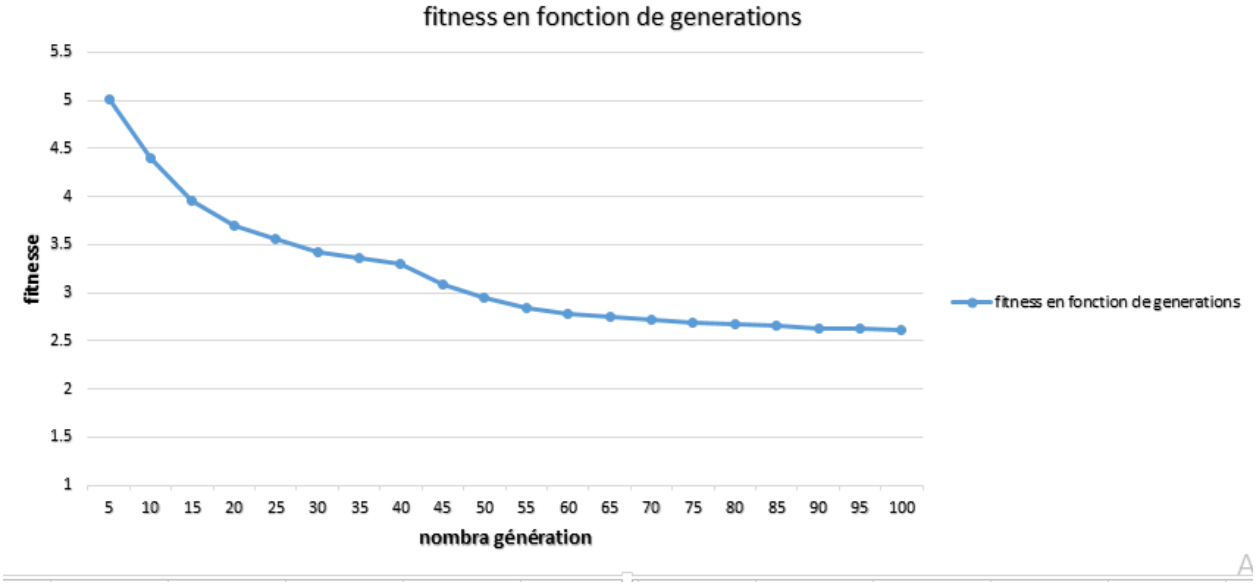


Figure 27: Graphe représenté les meilleures finesses de chaque génération.

11 CONCLUSION

Nous pouvons conclure que les AG 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.

Conclusion générale

Conclusion général

Les réseaux de neurones sont largement utilisés pour contrôler des systèmes complexes (notamment dans la robotique et les automates industrielles). Dans ce cas-là, l'entrainement du réseau de neurone devient une tâche difficile.

Dans notre travail on utilise un mécanisme évolutif pour surmonter la difficulté de l'entrainement des réseaux de neurones.

Nous avons résolu, ce problème, par une évolution de réseau neurone avec l'algorithme génétique (AG) pour augmenter la robustesse du processus de recherche des bonnes solutions et améliorer le processus d'entrainement.

Pour traiter ce thème, nous avons fait une étude détaillée sur les réseaux de neurones et l'algorithme génétique qui ont été l'objet de deux premiers chapitres. Nous avons présenté la conception et l'implémentation de l'algorithme génétique dans le troisième chapitre.

Une étude expérimentale présentée, dans le troisième chapitre, a montré que évolution de réseau neurone avec l'algorithme génétique a donné de bons résultats.

Comme perspectives, nous proposons consiste à élaborer d'autres techniques de codage de la structure du RN sous forme de chromosome qui tiennent compte de tous les paramètres qui caractérisent une architecture donnée. Cette partie constitue la pierre d'angle de la réussite de l'utilisation des algorithmes génétiques dans la résolution d'un problème d'optimisation

Bibliographies

Bibliographies

[1]: MC Culloch, W., Pitis, W. - « *A Logical Calculus for the Ideas Immanent in Nervous Activity* » - Bulletin Mathematics and Biophysics - 1943, N°. 5 - p. 115-133.

[2] : M. ZEMOURI «*Contribution à la surveillance des systèmes de production à l'aide des réseaux de neurones dynamiques: Application à la e-maintenance*» thèse de doctorat, université de France-comté, 2003.

[3]: B. Kosko. « *Unsupervised Learning in Noise* », IEEE Trans.Neural Net, Vol.1, N°1, pp. 44-57, Mars 1990.

[4]: A.A. Zhigljavsky. « *Theory of Global Random Search Mathematics and its Applications* », Kluwer Academic PubLishers. 1991.

[5] : A. Traynard. « *D'une méthode de gradient conjugué préconditionné élément par élément* » Institut National de Recherche en Informatique et en Automatique, Le Chesnay, France. 1990.

[6]: M. Wall, Galib. « *A C++ library of genetic algorithms components* » Massachusetts Institute of Technology MIT press, 1996, disponible.

[7]: **F.Z. DAIKH. «** *Contribution des approches de l'intelligence artificielle pour la stabilisation robuste des systèmes non linéaires* **»** thèse de doctorat, université de oran 2015.

[8]: **G. Dreyfus, J. Martinez, M. Samuelides, M. Gordon, F. Badran, S. thiria et L.Herault**, **«** *réseaux de neurones, méthodologies et application* **»**; EditionEyrolles.2002.

[9] : G. Dreyfur. « *Réseaux de neurones* : *Méthodologie et application* », édition Eyrolles, 2004

[10]: P.K. Simpson. « *Artificial Neural Systems* », Pergmon Press Elmsford, New York, 1989.

[11]: A. Blum. « *Neural Networks in C ++* », Wiley & Sons Edit, New York, 1992.

[12]: B. Kosko. « *Unsupervised Learning in Noise* », IEEE Trans.Neural Net, Vol.1, N°1, pp. 44-57, Mars 1990.

[13] : Souquet Amedee Radet Francois-Gerard, « *ALGORITHMES GENETIQUES », 2004.*

[14] : Mohamed Rida ABDESSEMED« Conception par Emergence Inversée d'Agents Autonomes dans le Cadre de Systèmes Complexes Adaptatifs, Université de Batna Faculté des Sciences, thèse Doctorat »,2013.

[15] : Terki Amel," Analyse des performances des algorithmes génétiques utilisant différentes techniques d'évolution de la population", mémoire de Magister, Université Mentouri Constantine

[16] : Lotfi AMIAR,"Un système hybride AG/PMC la reconnaissance de la parole arabe", mémoire de MAGISTER. UNIVERSITE Badji Mokhtar Annaba, 2005. Un système hybride AG/PMC la reconnaissance de la parole arabe