République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mentouri de Constantine

Institut d'informatique

MEMOIRE

Présenté par

Bendana Rokia

Pour l'obtention du Diplôme Magister en informatique

Option intelligence artificielle et génie logiciel

Intitulé

Sélection d'Attributs Basée sur un Algorithme Génétique Neuronal : Application à la Reconnaissance des Caractères Manuscrits

Dirigé par :

Dr S. Meshoul

Président :	Dr A. Chaoui	MC U.CONSTANTINE
Rapporteur :	Dr MK. Kholadi	MC U.CONSTANTINE
Examinateur :	Dr S. Chikhi	MC U.CONSTANTINE
Examinateur :	Dr F. Hachouf	MC U.CONSTANTINE

-2007-

Remerciements

Je remercie Dieu le tout puissant qui m'a bénis et permis d'arriver là où je suis.

Je tiens à témoigner ma reconnaissance à ma famille et mes amis pour leurs soutiens indéfectibles durant mon cursus d'étude.

Je tiens à remercier Monsieur Baatouche pour être à l'origine de ce projet, et pour ses conseils avisés.

Je tiens également à exprimer mes sincères remerciements à M^me^ S. Meshoul pour sa direction et ses conseils.

Je remercie les membres du jury, Messieurs

Allaoua Chaoui, Salim Chikhi, ainsi que

Madame Fella hachoufe, pour avoir accepté d'évaluer le présent travail.

Merci …….

Dédicace

Je dédie ce travail à mes chers parents qui ont toujours été à mess cotés et contribués d'une façon ou d'une autre à ma réussite, que dieu me les garde ...

Je le dédie aussi à ma famille et à toutes les personnes qui de prés ou de loin m'ont soutenus, elles se reconnaîtront dans ce message.

Sommaire

CHAPITRE 3 RESEAUX DE NEURONES

CHAPITRE 4

Partie 1 Le système de Reconnaissance Mis en Place

Introduction

Contexte et Motivation

L'écriture est une "représentation de la parole et de la pensée par des signes" ou 'une représentation physique d'un contenu sémantique'. L'apparition des premières écritures, il y a près de 5500 ans, a révolutionné à ce point l'humanité, que celle-ci décida ultérieurement que son Histoire débuterait à ce moment. L'importance de l'écriture en tant que moyen de communication pour l'homme n'a fait que croître depuis. L'écriture est restée, jusqu'au siècle dernier, le seul moyen matériel de transmission des connaissances. De ce fait, l'homme a toujours développé des techniques visant sa pérennité et sa diffusion. Aujourd'hui, la prolifération des ordinateurs dans notre société conduit à une dématérialisation du support de l'écriture, au profit de sa forme numérique. Ainsi sont nés les concepts de société sans papier et de bibliothèque virtuelle. Paradoxalement, plusieurs sociétés produisent un nombre considérable de documents écrits : lettres, enveloppes, chèques, formulaires, mémentos, il n'est alors pas surprenant de constater le développement de nombreuses techniques visant à sa reconnaissance automatique ″conversion des images qui sont compréhensibles par l'homme, en un code interprétable par un ordinateur″, cette forme numérique peut alors être diffusé plus largement via Internet, cette dernière a une durée de vie plus longue que le papier et évidemment permet le traitement automatique de l'information par l'intermédiaire d'ordinateurs.

Les applications de la reconnaissance de l'écriture sont nombreuses ; nous pouvons citer entre autres la lecture automatique de bons de commande, le traitement automatique des chèques, la vérification des signatures ou encore le tri automatique du courrier.

Nous distinguons deux types de système de la reconnaissance automatique, selon l'outil utilisé pour l'acquisition de l'écriture : les systèmes de reconnaissance en ligne (qui utilisent des tablettes graphiques munie d'un stylo optique pour faire entrer l'écriture), les systèmes de reconnaissance hors ligne (l'image originale du texte à traiter est sur papier). Cette dernière se distingue par deux approches : l'approche globale ou la reconnaissance de mot, l'approche analytique ou la reconnaissance de caractères isolés. L'écriture même peut être imprimée ou manuscrite. Dans ce travail nous nous somme intéressé à la reconnaissance hors ligne de caractères manuscrits isolés. Un des problèmes de la reconnaissance de l'écriture manuscrite est la grande variabilité associée au signal de l'écriture. Les sources de la variabilité sont : l'outil d'écriture et le support utilisé. Les différents auteurs (différents styles d'écriture) L'humeur de la personne au moment de la réalisation peut également conduire à une

déformation du signal. C'est cette variabilité qui complique fortement la tache de la reconnaissance. Pour cela, on retrouve une étape d'extraction de caractéristiques dans les systèmes de reconnaissances, qui permet à ces derniers de travailler avec les caractéristiques extraites des caractères et non avec les images des caractères. Les attributs (ou caractéristiques) extraits doivent présenter une certaine invariance *géométrique* d'une part, et une certaine invariance *statistique* d'autre part. L'invariance géométrique signifie une tolérance vis-à-vis des opérateurs de translation, de rotation, et de changement d'échelle, tandis que l'invariance statistique signifie une tolérance vis-à-vis du bruit inévitablement présent sur chaque caractère. Mais le problème de cette étape, est que La dimension de l'espace de représentation des attributs comporte alors autant de dimensions que les données de départ ; et parmi l'ensemble des attributs, certains ne sont pas aussi pertinents. Il est possible que certains correspondent à du bruit ou qu'ils soient peu informants, corrélés ou même inutiles au système pour l'accomplissement de sa tâche. Ce qui va dégrader les performances et la qualité du système de reconnaissance, delà jaillit le besoin d'ajouter un processus qui permet de réduire l'ensemble des attributs, en gardant ceux qui sont pertinents et non redondants, ce processus est la sélection d'attributs. C'est cette problématique particulière qui est le sujet de ce mémoire.

Problématique et Objectifs

C'est cette dernière problématique qui est abordée dans ce travail. La sélection d'attributs est un processus qui permet la réduction de l'ensemble des attributs de taille N à un sous ensemble de taille M tel que M< N conduisant à une amélioration des performances et de la qualité du Système. Ce processus fait partie des problèmes NP difficile, puisque pour choisir le sous ensemble optimale (de taille M), on a parcours à 2^{N-1} sous ensembles possibles. Toute méthode de sélection d'attributs passe généralement par 4 étapes : 1 *génération de sous ensemble d'attribut. 2 Stratégie d'évaluation. 3 Critère d'arrêt. 4 Validation des résultats* [Liu 05]. La génération de sous ensemble est une procédure de recherche qui produit un sous ensemble d'attributs candidat pour évaluation, basée sur une stratégie de recherche [Lan 94]. Chaque sous-ensemble candidat est évalué et comparé avec le meilleur antérieur, selon un certain critère d'évaluation [Liu 05]. Si le nouveau sous-ensemble s'avère être meilleur, il remplace le meilleur sous-ensemble précèdent. Le processus de génération de sous-ensemble et évaluation est répété jusqu' un critère d'arrêt donné est satisfait. Ensuite, le meilleur sous-ensemble sélectionné a généralement besoin d'être validé par des connaissances a priori ou par différents tests via des données réel /synthétique.

Dans le cadre de ce travail, nous avons proposé deux nouvelles méthodes de sélection d'attributs qui permettent de trouver le sous ensemble (de taille réduite) nécessaire et suffisant pour l'amélioration de la qualité de notre système de reconnaissance hors ligne des caractères manuscrits isolés. Ces méthodes utilisent les algorithmes génétiques (SSGA, GGA) pour explorer l'espace des sous ensembles des attributs, où chaque individu représente un sous ensemble d'attributs, dont le gène détermine la présence ou l'absence de l'attribut dans le sous ensemble. En utilisant l'approche enveloppante, les individus sont évalués en entraînant les classifieurs sur le sous-ensemble d'attributs indiqué par le chromosome et nous optons pour le taux d'erreur comme valeur pour sa fitness. Le classifieur utilisé est un réseau de neurones multicouche de type ″ perceptron multicouches″ ou ″PMC″. Ce dernier est entraîné par l'algorithme de rétropropagation du gradient, qui permet de minimiser le critère d'erreur (en modifiant les poids synaptiques) entre l'entrée et la sortie obtenu pour aboutir à la sortie désirée. Il constitue l'algorithme d'apprentissage convaincant pour le modèle puissant PMC. Le critère d'arrêt ici est le nombre de générations maximales fixé par des expériences. Les deux méthodes proposées sont appliquées sur un échantillon de la base standard CEDAR de caractères alphanumériques. Les résultats expérimentaux obtenus dans ce cadre montre l'efficacité des méthodes proposées.

Organisation du mémoire

Le travail présenté dans cette thèse s'intègre dans le domaine de la sélection des attributs, il se focalise

Cette thèse est structurée en quatre chapitres ainsi :

Le premier chapitre présente un état de l'art de la sélection d'attributs. Dans un premier temps une revue du domaine a été présentée. Elle a permis de mettre en évidence l'activité de ce domaine de recherche, particulièrement dans les communautés de l'apprentissage automatique et de l'analyse exploratoire des données. Après avoir exposé les différentes composantes nécessaires à un algorithme de sélection des attributs, les alternatives possibles pour leurs mises en oeuvre ont été présentées. Un certain nombre d'algorithmes a alors été décrit en fonction de la combinaisons de composantes utilisées.

Dans *le deuxième chapitre,* Une revue de littérature des algorithmes génériques est élaborée. Le but de ce chapitre est de familiariser le lecteur avec les algorithmes génétiques, en

expliquant ses différentes étapes, ses opérateurs de reproduction (croisement et mutation) et le réglage de ses paramètres. Tout en faisant le point sur les insuffisances de chacun.

Le troisième chapitre est consacré à l'état de l'art des réseaux de neurones. Nous donnons, un rapide historique des réseaux de neurones. Nous présentons ensuite, un bref aperçu de quelques propriétés élémentaires de neurophysiologie qui permettent au lecteur de relier neurones réels et neurones artificiels. Enfin, nous donnons les différents types de réseau et leurs principales applications.
Le Quatrième chapitre est réparti en trois parties :

Partie 1 : La première partie traite la reconnaissance de caractères ″ORC″. Elle permet de présenter les différentes étapes de ce système. Ensuite, le système ORC mis en œuvre est décrit en détail.

Partie 2 : La deuxième partie est consacré pour les deux algorithmes de sélection d'attributs mis en place.

Partie 3 : présente les résultats des tests effectués dans le cadre de ce travail.

Finalement, *Une conclusion* quant au travail réalisé est présentée. Rien n'étant parfait, cette dernière section est consacrée aux améliorations possibles et donc aux perspectives envisagées de ce travail.

CHAPITRE I

Sélection d'Attributs

Les performances d'un classificateur dépendent fortement de la qualité de représentation des formes à reconnaître, ce qui implique généralement l'obligation de représenter ces dernières au moyen d'un nombre élevé d'attributs. Il est alors fréquent qu'une partie de celui-ci ne contienne que des informations redondantes ou inutiles à la classification, rendant ainsi l'apprentissage du système plus complexe.

Le processus connu sous le nom de « Sélection d'attributs » a pour but de filtrer le vecteur des attributs de base, de manière à en extraire l'information discriminante et à Présenter celle-ci au classificateur de manière pertinente.

1. Introduction

Les performances d'une tache de traitement d'images dépendent fortement de la qualité de représentation des formes à traiter, ce qui implique généralement l'obligation de représenter ces dernières au moyen d'un nombre élevé d'attributs représentatifs. Il est alors fréquent qu'une partie de celles-ci ne contienne que des informations non pertinentes, redondantes ou inutiles à la tache, rendant cette dernière plus complexe. Donc il est nécessaire, lors de la construction d'un système, de limiter le nombre d'attributs pris en compte, de manière à optimiser ses performances. C'est exactement ce que fait un processus connu sous le nom de *"sélection d'attributs²* , qui a pour but de filtrer le vecteur des attributs de base, de manière à en extraire l'information discriminante et pertinente améliorant la qualité du système.

2. Quelques définitions

Sélection d' attributs :

Dash propose de regrouper les techniques de sélection d'attributs en fonction de l'objectif visé. Il identifie alors quatre classes distinctes :

- "Classic" : La sélection d'attributs est un processus qui permet de réduire l'ensemble des attributs de N à M tel que M < N, et la fonction d'évaluation soit optimal. Cette réduction de la dimensionnalité mène à une accélération de la vitesse du traitement et augmentation et amélioration de la précision du traitement.
- *²Idealized"* : La sélection d'attributs est un processus qui permet de trouver le sous-ensemble de taille minimale qui est nécessaire et suffisant pour atteindre l'objectif fixé **[Gra 02]**.
- *²Improving prediction accuracy*" : La sélection d'attributs est un processus qui permet de choisir un sous-ensemble d'attributs afin d'améliorer la précision de la prédiction ou diminuer la taille de la structure sans diminution significative de la précision de prédiction du classificateur, construit en utilisant seulement les variables sélectionnées**[Gra 02]**.
- "*Approximating original class distribution*" : La sélection d'attributs est un processus qui permet de sélectionner un sous-ensemble de variables tel que la distribution des classes résultante soit aussi proche que possible de la distribution des classes étant donné l'ensemble des variables complet**[Gra 02]**.

Pertinence d'un attribut

Il existe dans la littérature plusieurs définitions de la pertinence d'un attribut. Celles-ci dépendent de la nature des données.

Ø Les définitions suivantes données par Almullin et Diettrich en 1991 en considère que tous les attributs sont des booléens.

- Un attribut A_i est dit pertinent pour un concept C si A_i apparaît dans chaque formule booléenne qui représente C. il est dit non pertinent sinon.
- Un attribut A_i est pertinent, s'il existe deux instances (X_1, Y_1) et (X_2, Y_2) appartenant à deus classes différentes($y_1 \neq y_2$) telles que : la valeur de l'attribut A_i est différente pour les deus instances et les valeurs des autres attributs sont identiques.

Ø Gennari, Langley et Fisher, ils définissent les attributs pertinents comme ceux dont les valeurs de l'attribut changent systématiquement en fonction de l'appartenance de la donnée à telle ou telle catégorie. On peut l'exprimer formellement comme suit :

- Un attribut A_i est pertinent si et seulement si il existe une valeur a de cet attribut et une valeur de la classe Y,y, vérifiant $p(A_i=a_i)>0$ et tels que : $P(Y = y \backslash A_i = a_i) \neq P(Y = y \backslash A_i=a_i)$.avec $p(A_i = a_i)$ qui exprime le fait que les données contiennent au moins une instance ayant la valeur a pour l'attributs A.

  Cette formule nous permet de dire que A_i est pertinent si la probabilité de prédire Y, en connaissant la valeur de A_i, est différente de la probabilité de prédire Y sans connaître la valeur de A_i.

Ø Dash et Liu définissent un attribut non pertinent comme celui n'affectant pas la structure fondamentale des données et un attribut redondant comme celui n'apportent de rien de nouveau pour décrire la structure fondamentale des données.

Ø Kohavi et John Définirent la Pertinence forte d'un attribut comme suit

- Un attribut A_i est fortement pertinent si et seulement si il existe a_i, y et s_i pour lesquels $p(A_i = a_i, S_i = s_i)>0$ tels que $p(Y=y \backslash A_i=a_i, S_i=s_i) \neq p(Y=y, \backslash S_i=s_i)$.

  Cette formule nous permet de dire que A est très pertinent si la probabilité de prédire Yen connaissant la valeur de A et les valeurs des autres attributs de S est différente de prédire Y en connaissant seulement des valeurs des autres attributs de S.

- Définition textuelle

Les attributs non pertinents sont les attributs qui sont incohérent avec l'objectif du traitement.

Les attributs *inutiles*, sont ceux qui n'apportent pas d'information supplémentaire, ralentissent un peu l'apprentissage, mais ne modifient pas la qualité des résultats.

3. Le domaine de la sélection d'attributs

La sélection d'attributs est une technique permettant de choisir les caractéristiques, variables ou mesures les plus intéressantes, pertinentes ou informantes, à un système donné, pour la réalisation de la tâche pour laquelle il a été conçu. Cette phase est généralement un module important d'un système complexe. Les domaines d'application des techniques de sélection de caractéristiques sont variés tels que la modélisation, la classification, l'apprentissage automatique (*Machine Learning*) et l'analyse exploratoire de données (*Data Mining*). Dans ce mémoire nous nous intéressons plus particulièrement à la sélection d'attributs pour la classification **[Gra 02]**.

4. Le cadre général d'un algorithme de sélection d'attributs

Toute méthode de sélection d'attributs passe généralement par quatre étapes : 1 génération de sous ensemble d'attribut. 2 Stratégie d'évaluation. 3 Critère d'arrêt. 4 Validation des résultats (voir figure1) **[liu 05]**.

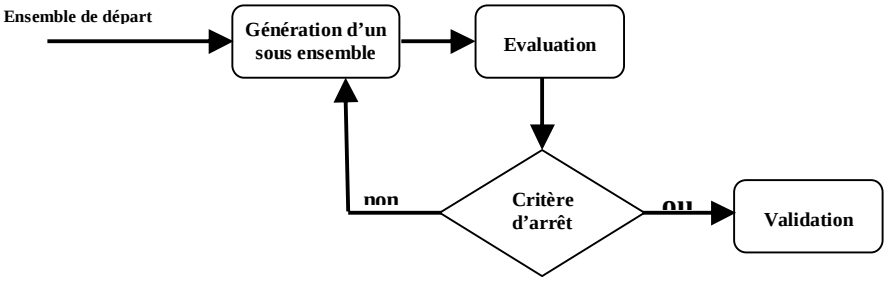


Figure 1. Processus de la sélection des attributs avec l'étape de validation.

4.1 Génération de sous ensemble

Est un processus qui produit un sous ensemble d'attributs candidat pour évaluation **[liu 05]**, la nature de celui-ci est déterminée par les deux étapes suivantes :

Point de départ

Est le point dans l'espace des sous ensembles d'attributs, à partir duquel on commence la recherche, et ce point même qui va affecter la direction de recherche **[Ala 05]**.

Pour N attributs, l'espace de recherche contient 2^N-1 sous ensembles possibles (illustré dans la figure 2).

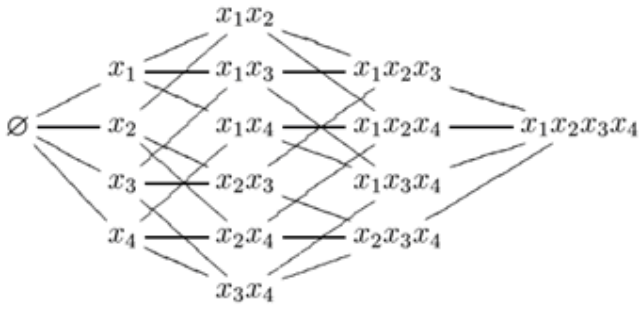


Figure 2 Graphe des sous ensembles de {x_1,x_2,x_3,x_4}.

- Si on commence avec zéro attribut, on doit faire des rajouts successifs d'attributs et c'est l'*Option forward* (figure 3).
- Si on commence avec l'ensemble de tous les attributs, on doit faire des suppressions successives et c'est *l'Option backward* (figure 4).
- Si on commence avec quelques attributs on doit faire des rajouts et des supressions et c'est l' *Option stepwise* .

Une stratégie de recherche : est une procédure qui permet d'explorer l'espace des combinaisons des attributs. Pour N attributs, l'espace contient 2^N-1 sous ensembles d'attributs possible. Cet espace de recherche est exponentiellement prohibitif pour la recherche exhaustive. Pour cela différentes stratégies de recherche sont à explorer, on peut les classifier en trois catégories **[Gra 02]**:

La recherche complète

Elle garantie le résultat optimal par rapport le critère d'évaluation utilisé. Une recherche exhaustive est complète mais la recherche ne doit pas être exhaustive pour qu'elle soit complète. Différents heuristiques et fonctions peuvent être utilisé pour réduire l'espace de recherche sans avoir le risque de perdre les résultats optimaux exemples **[liu 05]** : Branch and Bound (B&B) **[Cla 99]**,**[Ham 90]**,**[Gut 92]**, Best First Search (BFF), Beam Search (BS)**[Meu 04]**,**[Zho 05]**, Approximate Branch and Bound (AB&B), Minimum Description Length Method (MDLM).

4.1.2.2 La recherche séquentielle

Elle ne garantie pas le résultat optimal, elle consiste à rajouter ou éliminer itérativement des attributs. Il existe plusieurs variations de l'approche greedy hill-climbing comme :

Forward : cette approche part d'un ensemble d'attributs vide auquel, à chaque itération sont ajoutées un ou plusieurs attributs. Elle est également appelée approche ascendante **[Gra 02]**.

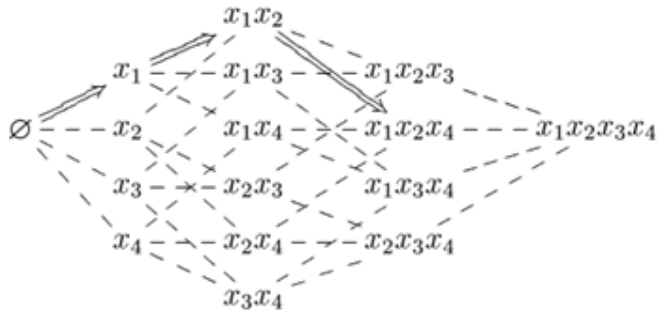


Figure 3 Sélection d'attributs Forward.

Backward : c'est l'approche inverse ; l'ensemble total des attributs est considéré au départ de la procédure itérative, chaque itération permet d'en supprimer. Une autre appellation est approche descendante **[Gra 02]**.

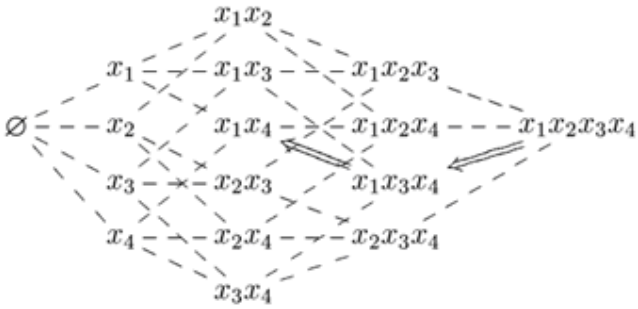


Figure 4 Sélection d'attributs Backrward.

Bidirectionnelle ,

Les méthodes SFS et SBS sont connus par leur simplicité de mise en oeuvre et leur rapidité. Cependant, comme elles n'explorent pas tous les sous-ensembles possibles d'attributs et ne permettent pas de retour arrière pendant la recherche; elles sont donc sous-optimales. Pour réduire cet effet, il y a d'autres alternatives qui permettent d'ajouter des attributs et d'en retirer d'autre c à d ajouter (ou retirer) P attributs en une étape et supprimer (ou ajouter) Q attributs à la prochaine étape et (P > Q). L'algorithme *"plus l- take away r"* est une généralisation de cette méthode. Il commence par élargir le sous-ensemble d'attributs en répétant *l* fois la procédure SFS, et puis d'en éliminer des attributs en répétant *r* fois la

procédure SBS. Le choix des paramètres *l* et *r* influe beaucoup sur la qualité des résultats ainsi que le temps de calcul.

Les méthodes flottantes SFFS et SFBS sont une extension de l'algorithme "*plus l-take away r*". Elles sont considérées comme les méthodes sous-optimales les plus efficace.

SFFS est l'acronyme de sequential forward floating search. L'algorithme SFFS consiste à appliquer après chaque étape *forward* autant d'étapes *backward* que le sous-ensemble d'attributs *F* correspondant améliore le critère d'évaluation *J*(*F*) à ce niveau de recherche.

SBFS est l'acronyme de sequential forward floating search. Le même principe de SFFS est appliqué sauf que les deux étapes sont inversées.

4.1.2.3 La recherche aléatoire : est la plus récente dans le domaine de la sélection d'attributs. Chaque sous ensemble d'attributs est généré d'une manière complètement aléatoire (c.-à-d., le sous-ensemble courant n'est pas issu d'une augmentation ou diminution d'attributs du sous-ensemble précédent). Cette catégorie nécessite le choix de différents paramètres. L'obtention de bons résultats à l'aide de ces techniques nécessite un choix judicieux de ces paramètres, comme les algorithmes génétiques (AG) **[Vfa 94]**,**[Pun 93]**, le recuit simulé (Simulated Annealing (SA)), random generation plus sequential selection (RGS).

La figure ci-dessous illustre la classification des méthodes de sélection d'attributs selon leur stratégie de recherche

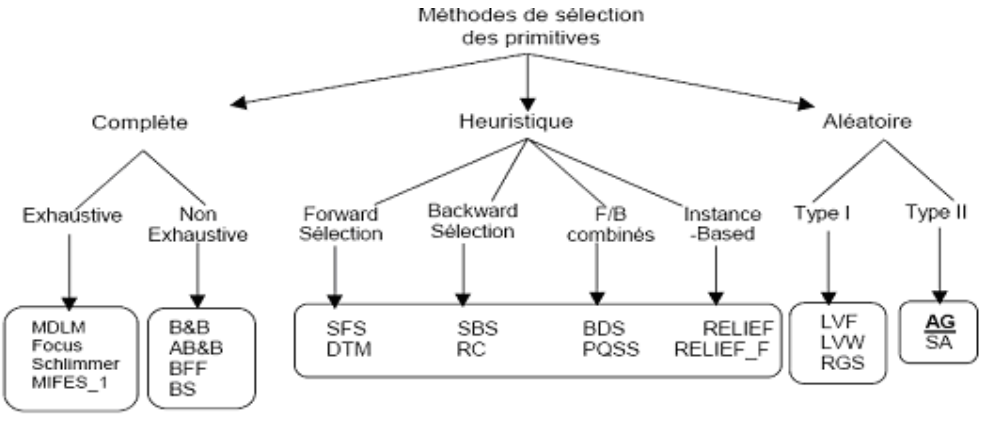


Figure 5 Les méthodes de sélection d'attributs classifiées selon la stratégie de recherche.

Le tableau ci-dessous présente une petite comparaison entre les méthodes de recherche :

	Rigueur (Exactitude)	Complexité	Avantages	Inconvénients
Exhaustive	Trouve toujours la solution optimale	Exponentielle	Grande exactitude	Complexité très élevé
Séquentielle	Bonne s'il n'y a pas besoin de retour arrière	Quadratique O(NEX2)	Simple et rapide	Pas de retour arrière
Aléatoire	Bonne avec un bon contrôle de paramètres	Généralement basse	Désigné pour échapper des optima locaux	Difficile de régler les paramètres

Tableau 1 Comparaison entre méthodes de recherches [Das 97].

4.2 L'évaluation du sous ensemble

Chaque nouveau sous-ensemble produit doit être évalué par un **certain** critère d'évaluation. Sa qualité est toujours déterminée par rapport ce critère (c.-à-d., un sous-ensemble choisi est optimale par rapport un critère bien déterminé et peut ne pas être optimal selon un autre critère). Il existe deux grandes classes d'algorithmes celles fondées uniquement sur les données et leur caractéristiques Filtrante (*filter*) et celles qui utilisent l'algorithme d'apprentissage même pour évaluer les sous-ensembles générés enveloppante (*wrapper*)[].

4.2.1 Filtrante (filter) *:* elle réalise l'évaluation sans tenir compte du classificateur, elle est fondée uniquement sur les données et leurs caractéristiques, elle utilise un critère indépendant du traitement **[Liu 05]**,**[Gra 02]**. Les critères indépendants les plus célèbres sont classifiés en 4 catégories :

Les mesures de distance sont également connues comme mesures de séparabilité, ou divergence, ou de discrimination. Elles permettent de mesurer les capacités de discrimination. Pour un problème de deux classe, un attribut X est préféré à un autre attribut Y si X induit la plus grande différence dans la distribution de probabilité conditionnelle des deux classes que Y, parce que nous essayons de trouver l'attribut qui peut séparer les deux classes aussi loin que possible. X et Y sont indistinct si la différence est zéro. . Un exemple de ce critère est la distance euclidienne, la distance de Mahalanobis, la distance de Bhattacharya, le critère de Fisher (voir Annexe).

Les mesures d'information déterminent typiquement le gain d'information apporté par l'attribut. Le gain d'information de l'attribut X est défini comme la différence entre l'incertitude antérieure et incertitude postérieure prévue en utilisant X.

L'attribut X est préféré que l'attribut Y si le gain d'information de X est plus grand que celui de Y. (ex : entropie de Shannon ou dérivées) [Liu 05].

Les mesures de dépendance sont également connues comme mesures de corrélation ou mesures de similitude. Elles mesurent la capacité pour prévoir la valeur d'une variable des valeurs des autres. Dans la sélection d'attributs pour classification, nous recherchons comment un attribut est fortement associé à la classe.
Un attribut X est préféré à un autre attribut Y si l'association entre l'attribut X et la classe C est plus grande que l'association entre Y et C. Dans la sélection d'attribut pour clustering, l'association entre deux attributs aléatoires mesure la similarité entre les deux. (Ex : coefficient de corrélation, information mutuelle).

Les mesures de consistance caractéristiquement différentes des autres méthodes à cause de leurs fortes relations avec la classe d'information dans la sélection du sous ensemble d'attributs. Ces mesures essayent de trouver le nombre minimal d'attributs qui séparent les classes comme consistance de l'ensemble complet d'attributs. L'inconsistance est définit comme deux instances ayant les mêmes valeurs d'attributs mais différentes étiquettes de classes voir [Das 03].

Le tableau suivant montre une comparaison des différentes fonctions d'évaluation indépendamment de la procédure de génération. Les différents critères d'évaluation sont : Généralité, Complexité, Exactitude.

A Comparison of Evaluation Functions

Evaluation Function	Generality	Time Complexity	Accuracy
Distance Measure	Yes	Low	–
Information Measure	Yes	Low	–
Dependence Measure	Yes	Low	–
Consistency Measure	Yes	Moderate	–
Classifier Error Rate	No	High	Very High

Tableau 2 Comparaison des fonctions d'évaluation.

4.2.2 Les méthodes enveloppantes [2]Wrappers[2]

Les wrappers ont été introduits plus récemment par John, Kohavi et Peger en 1994 **[Jkp94]**. Leur principe est de générer des sous-ensembles candidats et de les évaluer grâce à l'algorithme d'induction.

Dans la tache du clustering par exemple, le modèle Wrapper de la sélection d'attributs essaye d'évaluer la qualité du sous ensemble d'attributs par la qualité du cluster résultante de l'application de l'algorithme clustering sur le sous ensemble sélectionné. Il existe un certain nombre de critères pour estimer la qualité des résultats du clustering, comme cluster compactness, scatter separability, and maximum likelihood.

4.2.3 Les méthodes hybrides :

C'est pour avoir les avantages des deux modèles et éviter la pré-spécification du critère d'arrêt. Le modèle hybride est proposé récemment pour traiter les ensembles de large donnée. L'algorithme hybride typique utilise la mesure indépendante et l'algorithme d'induction (mining algorithm) pour évaluer les sous ensembles d'attributs. Il utilise la mesure indépendante pour décider le meilleur sous ensemble pour une cardinalité donnée et utilise l'algorithme d'induction pour sélectionner le sous ensemble final qui est le meilleure parmi les meilleures sous ensembles à travers les différentes cardinalités.

Les avantages et les inconvénients des deux méthodes

<table>
<tr><th colspan="2">Les méthodes filtrantes</th></tr>
<tr><th>Les avantages</th><th>Les inconvénients</th></tr>
<tr><td>• Exécution rapide, parce qu'elles impliquent un calcul non itératif. Pour cela elle s'exécute plus rapidement.
• Généralité Puisque elles évaluent les propriétés intrinsèques des données, plutôt que leurs interactions avec un classificateur particulier, leurs résultats montrent plus de généralité: la solution sera " bonne " pour un plus grand famille des classificateurs.</td><td>• Les filtrantes ont tendances à sélectionner un grand ensemble d'attributs.</td></tr>
<tr><th colspan="2">Les méthodes Wrappers</th></tr>
<tr><th>Les avantages</th><th>Les inconvénients</th></tr>
</table>

• Exactitude, les Wrappers réalisent généralement les meilleurs taux de réussites que les filtrantes.	• Exécution lente puisque les wrappers doivent appliquer le classificateur pour évaluer chaque sous-ensemble d'attributs. • Manque de généralité la solution manque de la généralité puisqu'elle est attachée au classificateur utilisé dans fonction d'évaluation. Le sous-ensemble " optimal " d'attributs sera spécifique au classificateur utilisé • Elles n'apportent pas vraiment de justifications théoriques à la sélection. • Elles ne permettent pas de comprendre les relations de dépendances entre les attributs.

Tableau 3 Avantages et inconvénients des méthodes enveloppantes et filtrantes.

4.3 Critère d'arrêt :

Le critère d'arrêt doit déterminer quand est ce que le processus de sélection d'attributs doit s'arrêter. Les critères d'arrêt les plus fréquents sont :

1. La recherche est accomplit.
2. Un certain seuil est atteint ou le seuil peut être un nombre spécifique (comme par exemple un nombre minimum d'attributs ou un nombre maximale d'itérations).
3. addition subséquente (ou suppression) de n'importe quel attribut ne produit pas un sous ensemble meilleure.
4. Un sous ensemble sélectionné est suffisamment bon (exemple un sous-ensemble peut être suffisamment bon si son taux d'erreur de la classification est moins que le taux d'erreur admissible pour une tâche donné).

4.4 Procédure de validation :

Une manière de faire la validation des résultats est de mesurer directement ces derniers en utilisant des connaissances a priori sur les données. Si nous connaissons les attributs pertinents à l'avance comme dans le cas des données synthétiques, nous pouvons comparer cet ensemble d'attributs connu avec les attributs sélectionnés (Les connaissances sur les attributs non pertinents ou redondants peuvent également aider). Dans les applications du monde réel, habituellement nous n'avons pas de connaissances a priori. Par conséquent, nous devons compter sur quelques méthodes indirectes en surveillant le changement des performances par rapport les changements des attributs. Par exemple, si nous employons le taux d'erreur de

classification comme un indicateur de performance pour le traitement, pour un sous-ensemble d'attributs sélectionné, nous pouvons simplement suivre l'expérience "avant et après" pour comparer le taux d'erreur de classificateur sur l'ensemble complet d'attributs et sur le sous-ensemble sélectionné.

5. Le Cadre général des méthodes filtrantes (Illustré dans le tableau)

Algorithme Filtre

Entrées : $D(F_0, F_1, \ldots, F_{n-1})$ // données d'entraînement avec N attributs

S_0 // un sous ensemble, à partir duquel on commence la recherche

δ // Critère d'arrêt

Output: S_{best} // sous ensemble optimale

```
01  Begin
02      Initialize: S_best = S_0;
03      γ_best = eval (S_0, D, M); //évaluer S_0 par le critère indépendant M
04      Do begin
05          S = générer (D); // générer un sous ensemble pour évaluation
06          γ = eval (S, D, M); // évaluer le sous ensemble courant S par M
07          If (γ est meilleur que γ_best)
08                  γ_best = γ;
09                  S_best = S;
10      End until (δ est satisfait);
11      Return S_best;
12  End;
```

Pour l'ensemble des données D, l'algorithme commence la recherche à partir du sous ensemble S_0 (un ensemble vide, un ensemble complet, ou n'importe sous ensemble sélectionné aléatoirement) et recherche dans l'espace des attributs selon une stratégie de recherche particulière. Chaque sous ensemble généré S est évalué par un critère indépendant M et comparé avec le meilleur précédent. S'il est le meilleur alors il sera considéré comme le meilleur sous ensemble courant. La recherche itère jusqu'à satisfaction du critère d'arrêt δ. L'algorithme retourne le meilleure sous ensemble courant S_{best} comme résultat finale [Liu 05].

6. Le cadre général des méthodes enveloppantes (Illustré dans le tableau)

Wrapper Algorithm

input: $D(F_0, F_1, ..., F_{n-1})$ // a training data set with N features
S_0 // a subset from which to start the search
δ // a stopping criterion

output: S_{best} // an optimal subset

```
01 begin
02     initialize: S_best = S_0;
03     γ_best = eval(S_0, D, A); // evaluate S_0 by a mining algorithm A
04     do begin
05         S = generate(D); // generate a subset for evaluation
06         γ = eval(S, D, A); // evaluate the current subset S by A
07         if (γ is better than γ_best)
08             γ_best = γ;
09             S_best = S;
10     end until (δ is reached);
11     return S_best;
12 end;
```

L'algorithme général des méthodes enveloppantes (wrappers) est très similaire à celui des méthodes filtrantes sauf qu'il utilise l'algorithme d'induction A à la place de la mesure indépendante M pour l'évaluation des sous ensembles S. Pour chaque sous ensemble généré S, il évalue sa qualité en appliquant l'algorithme d'induction sur les données avec le sous ensemble S et évalue la qualité des résultats. Différents algorithmes d'inductions produit différents résultats de sélection d'attributs [Liu 05].

7. Le cadre de catégorisation :

Il existe une gamme des algorithmes disponibles de sélection d'attributs. Afin de mieux comprendre le principe de chaque algorithme et les points commun et différences entre eux, Huan Liu, Lei Yu ont développé un cadre de catégorisation de trois dimensions.

Comme les stratégies de recherche et les critères d'évaluation sont les deux facteurs dominants distinguant les algorithmes de sélection d'attributs, pour cela ces deux facteurs sont choisis comme étant les deux dimensions du cadre. Dans le Tableau dessous, sous les stratégies de recherche, les algorithmes sont classés par catégories en complet, Séquentiel, et aléatoire. Sous les critères d'évaluation, les algorithmes sont classés par catégorie en filtre, Wrappers, et hybride. Ils considèrent que les algorithmes d'inductions comme une troisième dimension parce que la disponibilité d'information de classe dans la classification ou Tâche

de clustering affecte les critères d'évaluation utilisés dans les algorithmes de sélection d'attributs. En plus de ces trois dimensions de base, les algorithmes dans la catégorie de filtre sont encore distingués par des critères d'évaluation spécifiques comprenant distance, information, dépendance, et consistance. Dans la catégorie Wrapper, l'exactitude prédictive est employée pour la classification, et la qualité du cluster pour le clustering.

			Search Strategies					
			Complete		*Sequential*		*Random*	
Evaluation Criteria	*Filter*	*Distance*	B&B [67] BFF [92]		Relief [45] ReliefF [47] ReliefS [57] SFS [73] Segen's [79]			
		Information	MDLM [80]		DTM [12] Koller's [46] SFG [53] FCBF [95]	Dash's [17] SBUD [22]		
		Dependency	Bobrowski's [8]		CFS [34] RRESET [64] POE+ACC [66] DVMM [83]	Mitra's[63]		
		Consistency	Focus [2] ABB [56] MIFES1 [71] Schlimmer's [78]		Set Cover [16]		LVI [55] QBB [53] LVF [59]	
	Wrapper	*Predictive Accuracy or Cluster Goodness*	BS [25] AMB&B [31] FSLC [38] FSBC [39]		SBS-SLASH [13] WSFG [24] WSBG [24] BDS [25] PQSS [25] RC [26] SS [65] Queiros' [75]	AICC [23] FSSEM [27] ELSA [42]	SA [25] RGSS [25] LVW [58] RMHC-PF [82] GA [88] [93] RVE [85]	
	Hybrid	*Filter+Wrapper*			BBHFS [15] Xing's [91]	Dash-Liu's [20]		
			Classification	*Clustering*	*Classification*	*Clustering*	*Classification*	*Clustering*
			Data Mining Tasks					

Cadre de catégorisation des algorithmes de sélection d'attributs en trois dimensions [Liu 05].

En plus,ce cadre de catégorisation aide à trouver les combinaisons inexplorées des procédures de génération et évaluation.

8. Revue des différentes méthodes

Un algorithme de sélection d'attributs est composé de plusieurs modules. Étant donné que plusieurs approches sont possibles pour leur développement, il en résulte qu'un grand nombre d'algorithmes est disponible.

- Un des algorithmes les plus anciens en sélection de caractéristiques est l'algorithme *Branch and Bound* (*B&B*). Il fournit une solution optimale mais à une condition : le critère d'évaluation des sous-ensembles doit être monotone. Cette contrainte limite bien sûr le choix de la mesure de distance. Généralement celles qui sont utilisées sont la

distance de Mahalanobis, la distance de Bhattacharya, le critère de Fisher, la fonction discriminante et la divergence. Cette approche couvrant une grande partie de l'espace des solutions, elle nécessite un grand nombre d'opérations. Elle apporte tout de même une réduction par rapport à une approche exhaustive. Cependant, lorsque le nombre de variables devient important (> 30), il n'est tout de même pas raisonnable d'utiliser cette approche (voir Annexe).

- L'algorithme *FOCUS* fait également partie des méthodes appelées complètes. Contrairement à *B&B* qui utilise une mesure de distance, ce dernier algorithme utilise un critère d'évaluation des variables basé sur la cohérence. L'inconvénient de cet algorithme est qu'il ne fonctionne pas correctement lorsque les données sont bruitées. De plus il ne peut être utilisé que pour des problèmes de classification à deux classes. Il existe des variantes de cet algorithme ne permettant pas cependant d'abolir ces restrictions.
- L'algorithme *Relief :* La procédure de sélection est réalisée par l'intermédiaire d'une pondération des différents attributs. Premièrement l'utilisateur doit fixer le nombre d'échantillons que l'algorithme choisira aléatoirement dans le corpus d'apprentissage. Pour chacun d'eux il recherche au sein de ce sous-ensemble l'échantillon de la même classe le plus proche et celui d'une classe différente le plus proche également. Les poids des attributs sont mis à jour en fonction de ces valeurs. À la fin du processus les attributs sélectionnés sont ceux qui ont une pondération supérieure à un seuil donné. Ce dernier peut être déterminé automatiquement. L'algorithme *Relief* permet une sélection efficace lors de la présence de variables corrélées. Cependant il ne détecte pas la présence d'attributs redondants. Une restriction importante est qu'il ne fonctionne que dans le cas de la présence de deux classes. Konnenko propose une extension aux problèmes multi-classes. Cette version de l'algorithme est appelée *Relief-F []*.

Voila l'algorithme de Relief présenté en détail

Relief(D, S, $NoSample$, $Threshold$)
(1) $T = \phi$
(2) Initialize all weights, W_i, to zero.
(3) For $i = 1$ to $NoSample$/* Arbitrarily chosen */
Randomly choose an instance x in D
Finds its *nearHit* and *nearMiss*
For $j = 1$ to N
$W_j = W_j - diff(x_j, nearHit_j)^2 + diff(x_j, nearMiss_j)^2$
(4) For $j = 1$ to N
If $W_j \geqslant Threshold$
Append feature f_j to T
(5) Return T

- **SBS**

en 1971, proposé par Whitney.

1. Point de départ : ensemble complet.

Parcours de l'espace : Depth-First search.

2. Evaluation des sous-ensembles : J(**x**).

3. critère d'évaluation de corrélation : .

Partant de l'ensemble complet des caractéristiques on élimine à chaque itération la moins bonne caractéristique au sens de J(**x**).

• Inconvénient: L'algorithme SFS ne permet pas de réévaluer l'utilité d'une caractéristique une fois éliminée (=> sous-optimal).

• Efficace lorsque le nombre de caractéristiques optimal est grand.

- **Plus-L Minus-R Selection (LRS)**

• La méthode LRS permet de réduire les problèmes des méthodes sous-optimales SFS et SBS en permettant une certaine forme de « backtracking ».

• On utilise un processus d'alternance d'ajout de L caractéristiques et de retrait de R caractéristiques.

• Le nombre de caractéristique change de L-R.

• Si L > R alors LRS débute avec un ensemble **vide**, puis ajoute L et retire R caractéristiques en alternance.

• Si L < R alors LRS débute avec l'ensemble **complet**, puis retire R et ajoute L caractéristiques en alternance.

SFFS

• On part de l'ensemble vide et on ajoute 1 caractéristique à la fois.

• Toutefois à chaque inclusion d'une caractéristique on vérifie si on ne pourrait pas en enlever une autre selon J(x). Sinon on continue, si oui on enlève la caractéristique et on vérifie si on ne pourrait pas en enlever encore une autre.

- **SFBS**

• Similaire en partant de l'ensemble complet.

- **Min-Max**

• Supposons que nous avons obtenu un certain nombre de caractéristiques x_k parmi les *d* possibles

• Le ΔJ ajoutée par une **nouvelle** caractéristique notée y_k à l'une des caractéristiques courante x_k est:

$$\Delta J(y_j, x_k) = J(y_j, x_k) - J(x_k)$$

• Bien sûr, on cherche une caractéristique yj qui donnera un ΔJ grand pour toutes les caractéristiques courantes.

• L'algorithme Min-Max cherche y_k qui maximise ΔJ mais pour le x_k qui le minimise (pire cas).

9. Conclusions

Comme nous venons de le voir dans cette section, la sélection de caractéristiques est un domaine de recherche très actif, proposant un grand nombre d'algorithmes satisfaisant un grand nombre de configurations. Au cours de cette analyse du domaine, nous avons remarqué que la plupart des algorithmes développés le sont par des scientifiques travaillant dans les domaines de l'apprentissage automatique (*Machine Learning*) et de l'analyse exploratoire de données (*Data Mining*), plus qu'en reconnaissance de formes. Cette constatation nous a menés à énoncer que pour le domaine de la reconnaissance de formes, les travaux concernant l'extraction d'information pertinente ne sont pas effectués dans la même optique. Les applications de la reconnaissance de formes étant généralement soumises à des considérations de temps réel, l'étape d'extraction doit être la plus rapide possible. D'un autre coté les applications de l'apprentissage automatique et de l'analyse exploratoire de données n'ayant pas les mêmes contraintes, la quantité d'information extraite peut être beaucoup plus importante. Il est alors naturel de développer des algorithmes permettant de sélectionner les plus pertinentes.

Une autre remarque est que pour la plupart des algorithmes de sélection d'attributs étudiés, les données à traiter doivent être étiquetées. Pour une application de classification, cela revient à

considérer un apprentissage supervisé du système. Finalement ces algorithmes sont généralement appliqués à des problèmes de taille réduite ou moyenne, c'est-à-dire que le nombre de variables de départ est inférieur à 50. Pour un nombre supérieur il semble que peu d'algorithmes soient efficaces.

1. Introduction

Un algorithme génétique est utilisé lors des problèmes d'optimisation dès lors qu'il s'avère inconcevable d'imaginer une méthode exacte pour les résoudre. Les algorithmes génétiques appartiennent à une famille d'algorithmes appelée métaheuristiques dont le but est d'obtenir une solution approchée, en un temps correct, à un problème d'optimisation. Ils sont des techniques de recherche stochastiques inspirées de l'évolution des espèces naturelles. Nous pouvons dire de manière générale, qu'ils se réfèrent à la théorie selon laquelle les organismes vivants s'adaptent à leurs environnements et deviennent meilleurs (mieux adaptés). Cette adaptation se manifeste par le fait qu'un organisme devient capable de transmettre ses gènes à ses descendants, ce que Darwin a appelé "the survival of the fittest". La sélection naturelle montre son aptitude à résoudre un problème qui consiste à produire des organismes de mieux en mieux adaptés. Les AG tentent de simuler, de façon certes sommaire, les systèmes d'évolution naturelle sur une machine et ainsi d'utiliser leur capacité de résolution sur leurs propres problèmes de recherche et d'optimisation.

1.1 Origines

Les premiers travaux sur les algorithmes génétiques ont commencé dans les années cinquante, lorsque plusieurs biologistes américains ont simulé des structures biologiques sur ordinateur. Puis entre 1960 et 1970, John Holland de l'Université du Michigan, sur la base des travaux précédents, commença à s'intéresser à ce qui allait devenir les algorithmes génétiques. Ses travaux ont trouvé un premier aboutissement en 1975 avec la publication de *"Adaptation in Natural and Artificial System*[2]. Holland poursuivait un double objectif : améliorer la compréhension des processus naturels d'adaptation, et concevoir des systèmes artificiels possédant des propriétés similaires aux systèmes naturels.

L'idée fondamentale est la suivante : le pool génétique d'une population donnée contient potentiellement la solution, ou plutôt une meilleure solution à un problème adaptatif donné. Cette solution n'est pas exprimée car la combinaison génétique sur laquelle elle repose est dispersée chez plusieurs individus. Ce n'est que par l'association de ces combinaisons génétiques au cours de la reproduction que la solution pourra s'exprimer. L'originalité des travaux de Holland repose en particulier sur le fait qu'il n'a pas considéré les seules mutations comme source d'évolution, mais aussi et surtout les phénomènes de croisement: c'est en croisant les solutions potentielles existantes au sein du pool génétique que l'on peut se rapprocher de l'optimum **[Ren 00]**.

Domaine d'application

Les algorithmes génétiques classiques, tels que nous venons de les décrire, ont été étendus d'une part par imitation des phénomènes d'évolution naturelle comme la création de niches écologiques (pour l'optimisation de fonctions multimodales) ou la co-évolution de différentes populations (où l'on recherche un état d'équilibre). D'autre part, l'emploi d'autres types de codes que les codes binaires, ont permis des applications orientées Intelligence Artificielle comme les systèmes de classeurs, qui manipulent des chromosomes représentant des règles, ou la programmation génétique, où les chromosomes représentent directement des programmes arborescents.

Les algorithmes génétiques et les algorithmes évolutifs en général intéressent des chercheurs et des ingénieurs de disciplines très diverses, par exemple :

-- en optimisation : lorsque les fonctions à optimiser sont complexes, de forte dimensionnalité, irrégulières, mal connues.

-- en intelligence artificielle et sciences cognitives : où l'on exploite plutôt les capacités adap-tatives des algorithmes génétiques, et les techniques fondées sur les systèmes de classeurs, (réseaux de neurones, évolution de langages, grammaires).

-- en robotique : où l'on s'intéresse aux MOBOTS (MObile roBOTS) qui doivent pouvoir se mouvoir et agir dans des environnement inconnus, variables (programmation génétique, systèmes de classeurs) ;

-- en physique et en ingénierie : en tant que méthode d'optimisation pour les problèmes réels complexes (pour l'optimisation de structures par exemple) ;

-- en économie : pour la modélisation de comportements d'agents par exemple ;

-- en traitement d'images, du signal, pour détecter des formes caractéristiques, problème que l'on peut soit comprendre comme une optimisation, soit comme une application de règles de décision (SC) ;

-- en théorie des graphes et théorie des jeux : le problème du voyageur de commerce, notamment a beaucoup intéressé les chercheurs [Lut 94].

1.2 Terminologie

Les algorithmes génétiques étant basés sur des phénomènes biologiques, il convient de rappeler au préalable quelques termes de génétique.

Le chromosome

Les organismes vivants sont constitués de cellules, dont les noyaux comportent des chromosomes qui sont des chaînes d'ADN. L'élément de base de ces *chromosomes* (le

caractère de la chaîne d'ADN) est un *gène.* Sur chacun de ces chromosomes, une suite de gènes constitue une chaîne qui code les traits de l'organisme (la couleur des yeux ...). Différentes valeurs d'un même gène (trait) sont appelées *allèles*. La position d'un gène sur le chromosome est appelée *locus*. L'ensemble des gènes d'un individu est son *génotype* et l'ensemble du patrimoine génétique d'une espèce est le *génome* **[Wik 06]**, **[Gol 01]**.

La reproduction

Pendant la reproduction, la recombinaison (ou croisement) se produit en premier lieu. Les gènes des parents se combinent pour former un nouveau chromosome. La progéniture récemment créée (les enfants) peut ensuite subir une mutation. La mutation veut dire qu'un bit des éléments d'ADN est changé. Ce changement est dû principalement à des erreurs dans le copiage des gènes à partir des parents.

La Fitness

La Fitness d'un organisme est mesurée par le succès de l'organisme dans sa vie.

Evolution

L'évolution est un processus que chaque espèce dans la nature est soumise. Dans l'évolution, chaque espèce fait face aux problèmes de recherche d'une adaptation bénéfique, pour s'adapter aux changements rapides de l'environnement autour d'elle.

2. Principe de fonctionnement des algorithmes génétiques

Le principe général du fonctionnement d'un algorithme génétique est représenté dans la figure suivante

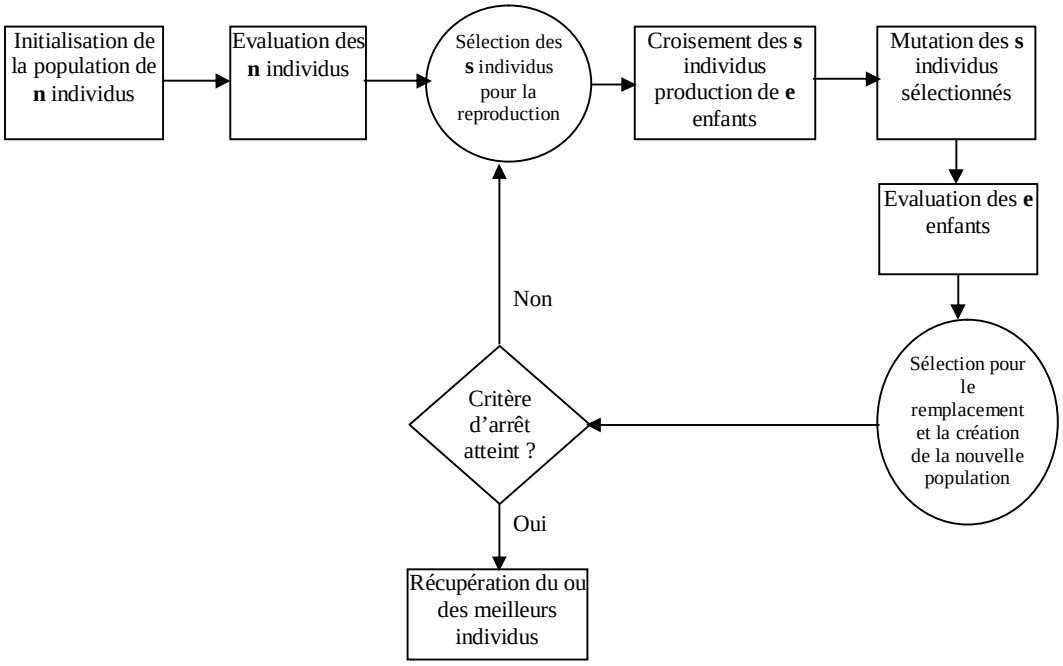


Figure 6 Principe du fonctionnement d'un algorithme génétique.

A présent, nous allons expliquer rapidement le principe de fonctionnement des algorithmes génétiques. Nous reviendrons plus en détail sur les différentes étapes par la suite :

1. **[Début]** Générer une population de n individus (ensemble de solutions potentielles) de façon aléatoire.
2. **[fitness]** Evaluer la fitness de chaque chromosome x de la population.
3. **[nouvelle population]** Créer la nouvelle population, cette dernière évolue en générations successives. Les individus les plus forts[*] survivent et se reproduisent entre eux pour créer de nouveaux individus, tandis que les plus faibles disparaissent petit à petit. Cela conduit à définir les trois opérateurs génétiques de base que sont la sélection, la recombinaison ou croisement (crossover) et la mutation.
   3.1. [sélection] Des couples de parents sont sélectionnés en fonction de leurs fitness.
   3.2. [croisement] L'opérateur de croisement est appliqué sur les couples parents avec une probabilité P_c et génère des couples d'enfants.
   3.3. L'opérateur de mutation leur est appliqué avec la probabilité P_m (P_m est généralement très inférieur à P_c) et génère des individus mutés P'. Le niveau

[*] La notion de fort ou faible c'est par rapport à la valeur de fitness de l'individu.

d'adaptation des enfants et des individus mutés sont ensuite évalués avant insertion dans la nouvelle population.

On réitère les opérations précédentes (3.1 3.2 3.3) jusqu'à ce qu'un critère d'arrêt soit satisfait. Différents critères d'arrêt de l'algorithme peuvent être choisis : nombre de générations fixé, limite de convergence de la population, population qui n'évolue plus suffisamment.

Comme nous voyons le fonctionnement d'un algorithme génétique est trop général. Il y a beaucoup de choses qui peuvent être implémentées différemment dans divers problèmes. Les points qui différent un algorithme d'un autre sont les suivants

Que représente le chromosome ? Quel type de codage utilise t-on ? Quel type de croisement et de mutation à utiliser? Comment sélectionner des parents pour croisement ? Une chose qui peut être faite de différente manière, mais l'idée principale est de choisir les meilleurs parents (dans l'espérance que les meilleurs parents vont produire les meilleurs enfants) **[Obi 98]**, toutes ces questions vont êtres discutés dans ce qui va suivre.

2.1 Codage d'un algorithme génétique

Pour les algorithmes génétiques, un des facteurs les plus importants, si ce n'est le plus important, est la façon dont sont codées les solutions (ce que l'on a nommé ici les chromosomes), c'est-à-dire les structures de données qui coderont les gènes.

2.1.1 Codage binaire

Ce type de codage est certainement le plus utilisé car il présente plusieurs avantages. Son principe est de coder la solution selon une chaîne de bits (qui peuvent prendre les valeurs 0 ou 1). Les raisons pour lesquelles ce type de codage est le plus utilisé sont tout d'abords historiques. En effet, lors des premiers travaux de Holland, les théories ont été élaborées en se basant sur ce type de codage. Et même si la plupart de ces théories peuvent être étendues à des données autres que des chaînes de bits, elles n'ont pas été autant étudiées dans ces contextes. Cependant, l'avantage de ce type de codage sur ses concurrents a tendance à être remis en question par les chercheurs actuels qui estiment que les démonstrations d'Holland sur les avantages supposés de ce codage ne sont pas révélatrices.

Ce type de codage n'est pas toujours bon comme le montrent les deux exemples suivants :

- deux éléments voisins en terme de distance de Hamming ne codent pas nécessairement deux éléments proches dans l'espace de recherche. Cet inconvénient peut être évité en utilisant un codage de Gray[†].
- Pour des problèmes d'optimisation dans des espaces de grande dimension, le codage binaire peut rapidement devenir mauvais. Généralement, chaque variable est représentée par une partie de la chaîne de bits et la structure du problème n'est pas bien reflétée, l'ordre des variables ayant une importance dans la structure du chromosome alors qu'il n'en a pas forcément dans la structure du problème **[All 97]**.

2.1.2 Codage de Gray : est venu pour éviter le premier inconvénient du codage binaire cité auparavant. Le codage de Gray est un codage qui a comme propriété que entre un élément n et un élément n + 1, donc voisin dans l'espace de recherche, un seul bit diffère.

2.1.3 Codage réel : est le plus récent où le gène est représenté par un nombre réel. L'opérateur de sélection reste identique à celui du codage binaire. En revanche, il existe d'autres opérateurs de croisement et de mutation. Ce type de codage apporte la simplicité et la flexibilité (ainsi que la vitesse) dans les AGs **[Val 01]**, **[Bab 07]**.

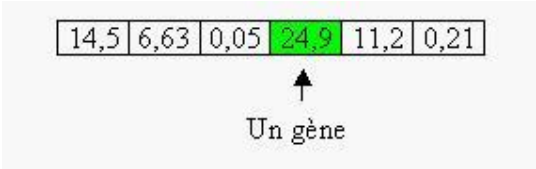


Figure 7 : représentation d'un gène en codage réel

2.1.4Codage sous forme d'arbre

Ce codage utilise une structure arborescente avec 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 où 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.

Le problème de ce type de codage est que les arbres résultants sont souvent difficiles à analyser et que l'on peut se retrouver avec des arbres « solutions » dont la taille sera importante alors qu'il existe des solutions plus simples et plus structurées à côté desquelles sera passé l'algorithme. De plus, les performances de ce type de codage par rapport à des codages en chaînes n'ont pas encore été comparées ou très peu. En effet, ce type d'expérience ne fait que commencer et les informations sont trop faibles pour se prononcer **[Wik 06]**.

[†]

Finalement, le choix du type de codage ne peut pas être effectué de manière sûre dans l'état actuel des connaissances. Selon les chercheurs dans ce domaine, la méthode actuelle à appliquer dans le choix du codage consiste à choisir celui qui peut répondre à ces questions :
– Le codage utilisé couvre-t-il toutes solutions envisageables avec la même probabilité ?
– Est-il possible de mettre au point un opérateur de croisement de telle sorte que les enfants aient de bonnes chances de combiner les avantages des parents?

- Comment définir un opérateur de mutation permettant une « bonne » exploration de l'espace de recherche ? **[Mou 05]**.

Finalement, le choix du type de codage ne peut pas être effectué de manière sûre dans l'état actuel des connaissances. Selon les chercheurs dans ce domaine, la méthode actuelle à appliquer dans le choix du codage consiste à choisir celui qui semble le plus naturel en fonction du problème à traiter et développer ensuite l'algorithme de traitement

2.2 Sélection

Comme son nom l'indique, la sélection vise à sélectionner les individus qui participeront à la création de la population suivante. C'est un opérateur clé sur lequel repose en partie la qualité d'un algorithme évolutionnaire. L'une des principales caractéristiques de tout opérateur de sélection est la pression sélective exercée sur la population. C'est un paramètre d'un maniement délicat, car opter pour une pression sélective trop forte revient à écraser dans l'œuf toute forme de diversité, pour ne garder que le groupuscule d'individus qui le haut du paé. Mais si l'on verse dans l'autre extrême, à savoir une pression sélective trop faible, l'algorithme tend à rapprocher d'une simple recherche aléatoire **[Tel 05]**, **[Bab 07]**. Il existe différentes méthodes de sélection. En voici les plus célèbres :

2.2.1 Roue de la fortune (Roulette wheel selection)

Le principe de la *Roue de la fortune* consiste à associer à chaque individu θ_i une probabilité P_i proportionnelle à sa fitness $f(\theta_i)$ dans la population. Cette probabilité pourra être calculée comme :

$$P_i = \frac{S(f(q_i))}{\sum_{i=1}^{N} S(f(q_i))}$$

Où S est une fonction régulière et croissante, au sens large. Chaque individu est alors

reproduit avec la probabilité P_i, certains individus (les ``bons") seront alors ``plus" reproduits et d'autres (les``mauvais") éliminés.

Remarque : Dans la pratique, il est aisé de tirer N individus, affectés de la probabilité P_i, parmi N avec remise. Pour cela, on associe à chaque individu un segment dont la longueur est proportionnelle à sa fitness (ou à $S(f(\theta_i))$, plus exactement).Ces segments sont ensuite concaténés sur un axe que l'on normalise entre 0 et 1 (voir figure). On tire alors un nombre aléatoire de distribution uniforme entre 0 et 1, puis on ``regarde" quel est le segment sélectionné, et on reproduit l'individu correspondant. Avec ce système, les grands segments, c'est-à-dire les bons individus, seront plus souvent adressés que les petits, on privilégie ainsi les individus ayant une forte fitness au détriment des individus moins forts, et un même individu pourra avec cette méthode être sélectionné plusieurs fois tout en gardant une structure aléatoire . Néanmoins, sur des populations de petite taille, il est difficile d'obtenir exactement l'espérance mathématique de sélection à cause du faible nombre de tirages (le cas idéal d'application de cette méthode est bien évidemment celui où la population est de taille infinie). On aura donc un biais de sélection plus ou moins fort suivant la dimension de la population **[Ber 03], [Mat 02]**.

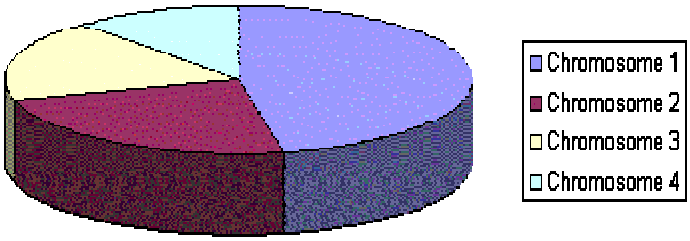


Figure 8 Exemple de sélection par roulette

2.2.2 La sélection par rang

La sélection précédente rencontre des problèmes lorsque la valeur d'adaptation des chromosomes varie énormément. Si la meilleure fonction d'évaluation d'un chromosome représente ***90%*** de la roulette alors les autres chromosomes auront très peu de chance d'être sélectionnés et on arriverait à une stagnation de l'évolution.

La sélection par rang trie d'abord la population par fitness. Ensuite, chaque chromosome se voit associé un rang en fonction de sa position. Ainsi 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). La sélection par rang d'un chromosome est la même que

par roulette, mais les proportions sont en relation avec le rang plutôt qu'avec la valeur de l'évaluation. Avec cette méthode de sélection, 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 **[Cou 03]**, **[Obi 98]**.

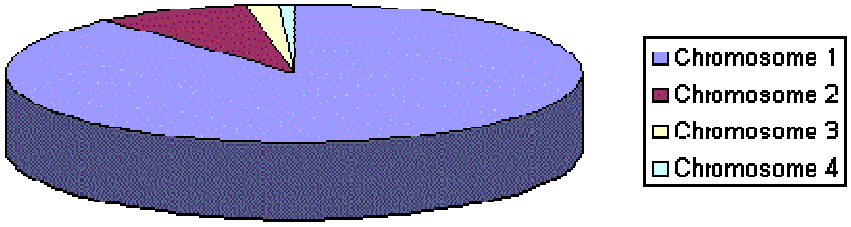


Figure 9 Situation before ranking (graph of fitnesses)

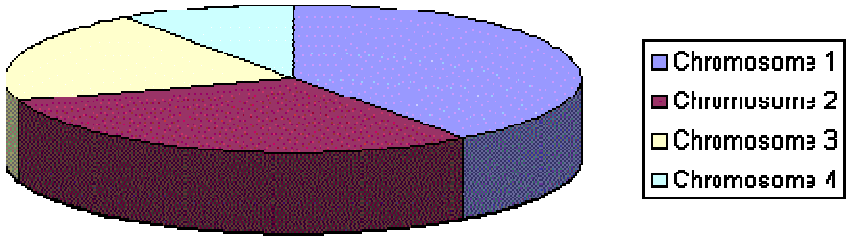


Figure 10 Situation after ranking (graph of order numbers)

2.2.3 La sélection par tournoi n'utilise aussi que des comparaisons entre individus -- et ne nécessite même pas de tri de la population. Elle possède un paramètre *T*, taille du tournoi. Pour sélectionner un individu, on en tire *T* uniformément dans la population, et on sélectionne le meilleur de ces *T* individus. Le choix de *T* permet de faire varier la *pression sélective*, c'est-à-dire les chances de sélection des plus performants par rapport aux plus faibles. A noter que le cas *T*=2 correspond, en espérance et au premier ordre en fonction de *P*, à la sélection par le rang linéaire **[Gou 06]**.

Le principe de la sélection par tournoi augmente les chances pour les individus de piètre qualité de participer à l'amélioration de la population. Le principe est très simple. Un tournoi consiste en une rencontre entre **T** individus (taille du tournoi) pris au hasard dans la population. Le vainqueur du tournoi est l'individu de meilleure qualité. Nous pouvons choisir

de ne conserver que le vainqueur comme nous pouvons choisir de conserver les 2 meilleurs individus ou les 3 meilleurs. A nous de voir, selon que nous souhaitons créer beaucoup de tournois, ou bien créer des tournois avec beaucoup de participants ou bien mettre en avant ceux qui gagnent les tournois haut la main. Nous pouvons faire participer un même individu à plusieurs tournois. Une fois de plus, Nous sommes totalement libre quant à la manière d'implémenter cette technique de sélection.

Remarque Le choix de T permet de faire varier la *pression sélective*, c'est-à-dire les chances de sélection des plus performants par rapport aux plus faibles. A noter que le cas T=2 correspond, en espérance et au premier ordre en fonction de P, à la sélection par le rang linéaire **[Gou 06]**, **[Kha 05]**.

2.2.4 Reste stochastique sans remplacement (La Stochastic remainder without replacement)

La sélection stochastique avec reste s'inspire très fortement de la sélection par roulette, sauf qu'elle y ajoute un léger aspect déterministe. On commence par déterminer la probabilité $Pi = fi/F$ qu'a chaque individu d'être sélectionné. Soit N la taille de la population. On pose alors $Mi = N. Pi$, où Mi correspond au nombre moyen de copies qu'un individu peut espérer recevoir par une sélection de type roulette. On définit ensuite :

$n_i = E(M_i)$ où E désigne la partie entière.

$r_i = M_i - E(M_i)$ les r_i sont les restes .

Le caractère déterministe de la sélection stochastique avec reste intervient alors, car on affecte automatiquement à chaque individu n_i copies dans la génération suivante.

Comme : $\sum_{i=1}^{N} n_i \leq N$

Il reste en générale un nombre $d = N - \sum_{i=1}^{N} n_i$ de copies à affecter pour compléter la population de la génération suivante. Pour attribuer ces copies restantes, le processus utilisé est une roulette classique, qui s'appuie non pas sur les valeurs Pi mais sur les restes ri **[Tel 05]**.

2.2.5 Sélection uniforme

La sélection se fait aléatoirement, uniformément et sans intervention de la valeur d'adaptation. Chaque individu a donc une probabilité 1/P d'être sélectionné, où P est le nombre total d'individus dans la population **[Wik 06]**.

2.2.6 Elitisme

A la création d'une nouvelle population, il y a de grandes chances que les meilleurs chromosomes soient perdus après les opérations d'hybridation et de mutation. Pour éviter cela, on utilise la méthode d'élitisme. Elle consiste à copier un ou plusieurs des meilleurs chromosomes dans la nouvelle génération. Ensuite, on génère le reste de la population selon l'algorithme de reproduction usuel. Cette méthode améliore considérablement les algorithmes génétiques, car elle permet de ne pas perdre les meilleures solutions **[Cou 03]**, **[GOL 01]**. Elle a l'avantage de permettre une convergence (plus) rapide des solutions, mais au détriment de la diversité des individus. On prend en effet le risque d'écarter des individus de piètre qualité, mais qui auraient pu apporter de quoi créer de très bonnes solutions dans les générations suivantes. Cette méthode est extrêmement sensible à la présence d'optimas locaux, c'est à dire à la présence de solutions paraissant optimales tant que l'on ne s'en éloigne pas trop - le véritable optimum pouvant alors être situé dans une toute autre partie du domaine de recherche **[Kha 05]**.

2.2.7 Sélection par troncature

Il existe également des algorithmes évolutionnaires qui considèrent que la sélection ne doit pas être laissée au hasard. Ainsi, les *breeding genetic algorithms,* développés par (schlierkam-pvoosen, 1993) s'inspirent des méthodes de sélection utilisées par les éleveurs. L'une des plus classique étant la sélection par troncature *(truncation selection)* où on choisit de manière déterministe les *T%* meilleurs individus d'une génération pour générer la suivante. Cela revient en quelque sorte à généraliser le modèle élitiste à tout le processus de sélection, qui s'appuie non pas sur les valeurs Pi mais sur les restes *ri* **[Tel 05]**.

2.3 Le croisement

Le croisement a pour but d'enrichir la diversité de la population en manipulant la structure des chromosomes **[Sys89]**. Classiquement, les croisements sont envisagés avec deux parents et

génèrent deux enfants. Il consiste à échanger les gènes des parents afin de donner des enfants qui portent des propriétés combinées. Bien qu'il soit aléatoire, cet échange d'informations offre aux algorithmes génétiques une part de leur puissance : quelque fois, de " bons " gènes d'un parent viennent remplacer les " mauvais " gènes d'un autre et créent des fils mieux adaptés aux parents **[Mat 02]**.

2.3.1 Croisement simple à un point

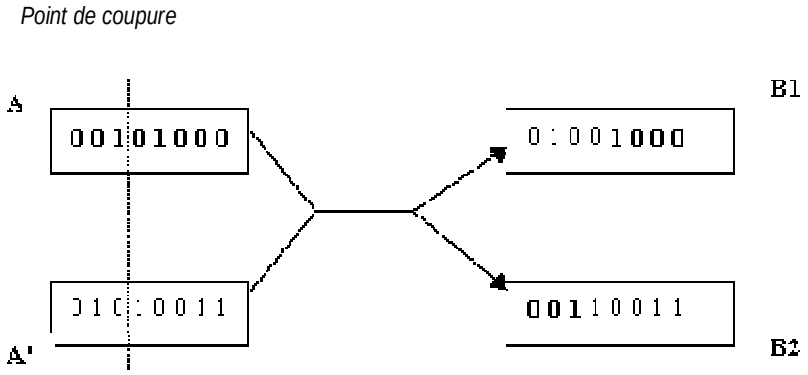


Figure 11 opérateur de croisement, un seul point de coupure

Le croisement à un point est l'opérateur de croisement le plus simple. Il consiste à sélectionner un point de coupure, puis à subdiviser le génotype de chacun des parents en deux parties de part et d'autre part de ce point. Les fragments obtenus sont alors échangés pour créer les génotypes des enfants. Ainsi, la figure 2.1 ésente le cas où le point de coupure tombe après le troisième locus ; les parents A et A' produisent alors les enfants B1 et B2 **[TEL 05]**.

2.3.2 Croisement en deux points:

On choisit au hasard deux points de croisement et on échange les fragments situés entre ces deux points **[Ber 03]**.

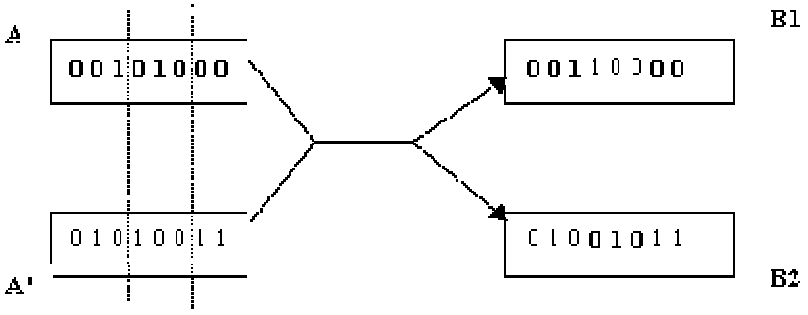


Figure 12 opérateur de croisement à deux points de coupures

2.3.3 Le *croisement multipoints*,

Le croisement multipoints est une généralisation du croisement à un point puisque qu'au lieu de choisir un seul point de coupure, on en sélectionne n. L'intérêt d'une telle approche apparaît rapidement, car il devient possible en un seul croisement d'échanger des blocs qui sont situes au milieu du chromosome sans avoir a procéder par étape. Bien sûr la contrepartie est que c'est une technique qui est plus destructrice par rapport à ces même blocs, puisqu'ils peuvent se retrouver scindes en plusieurs fragments sans intérêt **[Bar 03]**.

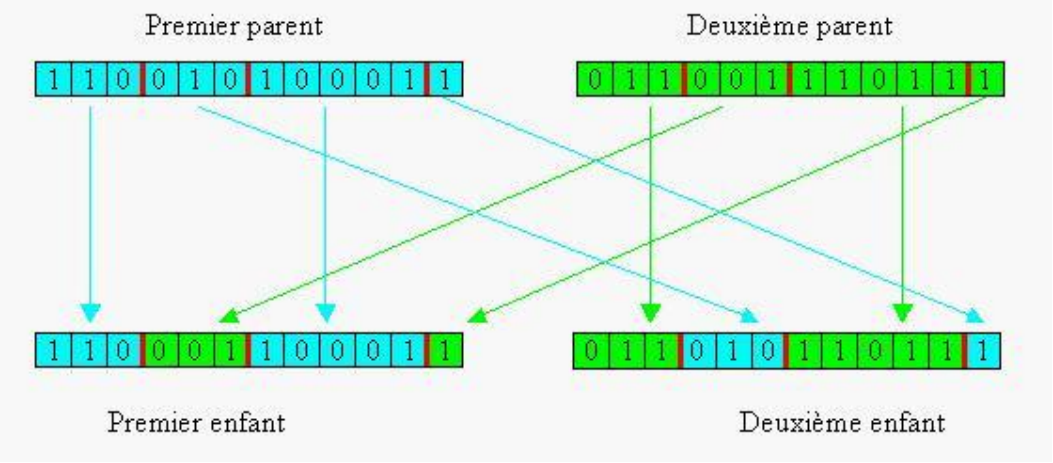


Figure 13 : exemple d'un croisement trois points [Bab 07]

2.3.4 Le croisement uniforme

Il consiste à définir de manière aléatoire un "masque", c'est-à-dire une chaîne de bits de même longueur que les chromosomes des parents sur lesquels il sera appliqué. Ce masque est destiné à savoir, pour chaque locus, de quel parent le premier fils devra hériter du gène s'y trouvant; si face à un locus le masque présente un 0, le fils héritera le gène s'y trouvant du parent n° 1, si il présente un 1 il en héritera du parent n° 2. La création du fils n° 2 se fait de manière symétrique : si pour un gène donné le masque indique que le fils n° 1 devra recevoir celui-ci du parent n° 1 alors le fils n° 2 le recevra du parent n°2, et si le fils n° 1 le reçoit du parent n° 2 alors le fils 2 le recevra du parent n° 1 **[Sou 04]**.

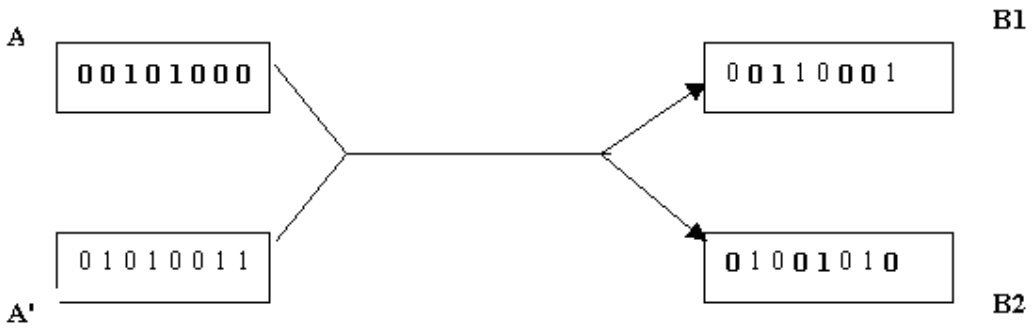


Figure 14 Exemple d'un croisement uniforme.

2.4 La mutation

L'opérateur de mutation apporte aux algorithmes génétiques l'aléa nécessaire à une exploration efficace de l'espace. Cet opérateur nous garantit que l'algorithme génétique sera susceptible d'atteindre tous les points de l'espace d'état, sans pour autant les parcourir tous dans le processus de résolution. Ainsi, en toute rigueur, l'algorithme génétique peut converger sans croisement, et certaines implantations fonctionnent de cette manière (Fogel et al.[*FOGEL*]) et sont conformes aux résultas théoriques de R. Cerf [*11*]. Les propriétés de convergence des algorithmes génétiques sont donc fortement dépendantes de cet opérateur **[Bon 95]**

La mutation est une modification aléatoire de la valeur d'un gène qui se produit avec une probabilité fixée. Cette dernière est généralement comprise entre 0,001 et 0,01. Il est nécessaire de choisir pour cette probabilité une valeur relativement faible de manière à ne pas tomber dans une recherche aléatoire et conserver le principe de sélection et d'évolution (Si elle est trop élevé, on risque de ne pas voir une bonne convergence, et s'il est trop faible, ça réduit d'autant son effet).

2.4.1 Mutation binaire :

La mutation binaire s'applique à un seul chromosome. Un bit du chromosome est tiré au hasard. Sa valeur est alors inversée.

Il existe une variante où plusieurs bits peuvent muter au sein d'un même chromosome. Un test sous le taux de mutation[‡] est effectué non plus pour le chromosome mais pour chacun de ses bits : en cas de succès, un nouveau bit tiré au hasard remplace l'ancien [**Tel 05]**.

[‡] Probabilité de mutation

Allèle à muter

Figure 15 Exemple de mutation binaire.

2.4.2 / Mutation réelle :

La mutation réelle ne se différencie de la mutation binaire que par la nature de l'élément qu'elle altère : ce n'est plus un bit qui est inversé, mais une variable réelle qui est de nouveau tirée au hasard sur son intervalle de définition **[Tel 05]**.

2.4.3 / Mutation non uniforme :

La mutation non uniforme possède la particularité de retirer les éléments qu'elle altère dans un intervalle de définition variable et de plus en plus petit. Plus nous avançons dans les générations, moins la mutation n'écarte les éléments de la zone de convergence. Cette mutation adaptative offre un bon équilibre entre l'exploration du domaine de recherche et un affinement des individus. Le coefficient d'atténuation de l'intervalle est un paramètre de cet opérateur **[Tel 05]**.

Cet opérateur dispose de 4 grands avantages :

• Il garantit la diversité de la population, ce qui est primordial pour les algorithmes génétiques.

• Il permet d'éviter un phénomène connu sous le nom de ***dérive génétique***. On parle de dérive génétique quand certains gènes favorisés par le hasard se répandent au détriment des autres et sont ainsi présents au même endroit sur tous les chromosomes. Le fait que l'opérateur de mutation puisse entraîner de manière aléatoire des changements au niveau de n'importe quel locus permet d'éviter l'installation de cette situation défavorable.

• Il permet de limiter les risques d'une convergence prématurée causée par exemple par une méthode de sélection élitiste imposant à la population une pression sélective trop forte. En effet, dans le cas d'une convergence prématurée on se retrouve avec une population dont tous les individus sont identiques mais ne sont que des optimums locaux. Tous les individus étant identiques, le croisement ne changera rien à la situation. En effet, l'échange d'informations par croisement entre des individus strictement identiques est bien sûr totalement sans conséquences; on aura beau choisir la méthode de croisement qu'on veut on se retrouvera

toujours à échanger des portions de chromosomes identiques et la population n'évoluera pas. L'évolution se retrouvant bloquée on n'attendra jamais l'optimum global.

La mutation entraînant des inversions de bits de manière aléatoire permet de réintroduire des différences entre les individus et donc de nous extirper de cette situation.

Il est quand même utile de garder à l'esprit que ceci n'est pas une solution "miracle" et qu'il est bien entendu plus intelligent de ne pas utiliser de méthodes de sélection connues pour entraîner ce type de problème.

• La mutation permet d'atteindre la propriété d' ***ergodicité***. L'ergodicité est une propriété garantissant que chaque point de l'espace de recherche puisse être atteint.

En effet, une mutation pouvant intervenir de manière aléatoire au niveau de n'importe quel locus, on a la certitude mathématique que n'importe quel permutation de notre chaîne de bits peut apparaître au sein de la population et donc que tout point de l'espace de recherche peut être atteint.

Grâce à cette propriété on est donc sûr de pouvoir atteindre l'optimum global.

A chaque étape de l'algorithme, il faut effectuer le compromis entre **explorer** l'espace de recherche, afin d'éviter de stagner dans un optimum local, et **exploiter** les meilleurs individus obtenus, afin d'atteindre de meilleurs valeurs aux alentours. Trop d'exploitation entraîne une convergence vers un optimum local, alors que trop d'exploration entraîne la non convergence de l'algorithme.

Typiquement, les opérations de sélection et de croisement sont des étapes d'exploitation, alors que l'initialisation et la mutation sont des étapes d'exploitation. On peut ainsi régler les parts respectives d'exploration et d'exploitation en jouant sur les divers paramètres de l'algorithme (probabilités d'application des opérateurs, pression de sélection, ...). Malheureusement, il n'existe pas de règles universelles de réglages et seuls des résultats expérimentaux donnent une idée du comportement des divers composantes des algorithmes **[Gou 06]**.

3. Le réglage des paramètres d'un algorithme génétique :

Par sa difficulté, le réglage des paramètres d'un AG, est souvent assimilé à de la magie noire (la littérature des AG utilise le terme de *Black Art*). A chaque étape de l'algorithme, il faut effectuer le compromis entre ***explorer*** l'espace de recherche, afin d'éviter de stagner dans un optimum local, et exploiter les meilleurs individus obtenus, afin d'atteindre de meilleurs valeurs aux alentours. Cette balance entre l'exploration (par mutation, initialisation) et l'exploitation (par croisement, sélection) doit être soigneusement respectée. Car un ***taux de***

mutation[§] trop élevé entraîne la destruction de gènes avant qu'ils n'aient eu la chance d'être assemblés par croisement afin de former des structures valables, ce qui entraîne la non convergence de l'algorithme. De l'autre côté, si le ***taux de croisement***[**] est excessif, la population sera uniformisée trop rapidement et la population convergera alors prématurément vers un type d'individu probablement sous-optimal. On ne parle alors plus de la survivance du plus apte, mais de la survivance du plus médiocre.

Il y a aussi d'autres paramètres qui influencent sur le comportement de l'algorithme génétique comme la ***taille de la population*** [††]**,** Car s'il y a peu de chromosomes, AG a peu de possibilités d'exécuter le croisement et seulement une petite partie de l'espace de recherche est à explorer (évolution vers un optimum local). De l'autre côté, s'il y a trop de chromosomes, AG ralentit. Il y a beaucoup de chercheurs qui 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 à 50, autres entre 20 et 100 de manière empirique, et d'autres ont mentionné qu'il peut exister une relation entre la taille de la population et le nombre de gènes.

Le nombre de génération, pour le fixer, les chercheurs préfèrent qu'il soit assez grand pour mieux visualiser la convergence de la solution.

Malheureusement, il n'existe pas de règles universelles de réglages et seuls des résultats expérimentaux donnent une idée du comportement des algorithmes génétiques par rapport ces paramètres **[Obi 98]**, **[Gou 06]**.

4. Les avantages et les inconvénients des algorithmes génétiques :

4.1 Les avantages

- Les algorithmes génétiques permettent de résoudre des problèmes d'optimisation. Leurs avantages sont
- il n'y a pas de contraintes sur les fonctions à optimiser (dérivabilité, continuité).
- ils peuvent donner plusieurs solutions (par exemple une seule pour le recuit simulé) **[Bab 07]**.
- Ils peuvent parcourir rapidement un grand ensemble de solutions.
- La nature inductive des AG signifie qu'il ne doit connaître aucune règle du problème. Ils travaillent avec leurs propres règles internes.

[§] Le taux de mutation indique combien de fois les parties de chromosomes subissent une mutation.

[**] Le taux de croisement indique combien de fois le croisement sera exécuté.

[††] Ça indique le nombre de chromosomes dans chaque population (pour une seule génération).

4.2 Les inconvénients

- c’est difficile de trouver un bon codage adapté à la structure du problème.
- L’application de la fonction de décodage lords l’évaluation de la fitness est coûteuse en temps de calcul.
- Le temps de calcul : par rapport à d'autres métaheuristiques, ils nécessitent de nombreux calculs (surtout au niveau de la fonction d'évaluation).
- Ils sont de plus souvent difficiles à mettre en oeuvre: des paramètres comme la taille de la population, le nombre de génération ou le taux de mutation sont parfois difficile à déterminer. Or le succès de l'évolution en dépend et plusieurs essais sont donc nécessaires, ce qui limite encore l'efficacité de l'algorithme. En outre, choisir une bonne fonction d'évaluation est aussi critique. Celle-ci doit prendre en compte les bons paramètres du problème. Elle doit donc être choisie avec soin.
- Les outils théoriques disponibles pour effectuer les choix optimaux (sélection des opérateurs génétiques) sont quasi-inexistants. Au final, il est nécessaire d’expérimenter de nombreuses combinaisons pour obtenir de bons résultats et il est souvent difficile de justifier les choix effectués.
-
- Il faut aussi noter l'impossibilité d'être assuré, même après un nombre important de générations, que la solution trouvée est la meilleure. On peut seulement être sûr que l'on s'est approché de la solution optimale (pour les paramètres et la fonction d'évaluation choisie), sans la certitude de l'avoir atteinte.
- Un autre problème important est celui des optimums locaux. En effet, lorsqu'une population évolue, il se peut que certains individus qui à un instant occupent une place importante au sein de cette population deviennent majoritaires. À ce moment, il se peut que la population converge vers cet individu et s'écarte ainsi d'individus plus intéressants mais trop éloignés de l'individu vers lequel on converge. Pour vaincre ce problème, il existe différentes méthodes comme l'ajout de quelques individus générés aléatoirement à chaque génération, des méthodes de sélection différentes de la méthode classique.

Conclusion

Ce chapitre nous a permis d’avoir une vue générale sur le principe des algorithmes génétiques. Nous pouvons conclure que les AG sont efficace pour traiter des problèmes assez

complexes. La résolution de ces problèmes est obtenue grâce des mécanismes de sélection, d'hybridation et de mutation. Elles présentent certains limites et difficultés. Ces difficultés reposent sur le bon réglage des paramètres tels que : la taille de la population, le nombre de génération, les taux de croisement et de mutation et le choix des opérateurs de reproduction. Seuls les résultats expérimentaux qui peuvent nous donner une idée sur les valeurs des paramètres.

CHAPITRE

Réseaux de Neurones III

L'idée des réseaux de neurones est de s'inspirer de l'organisation du cerveau pour traiter les types de problèmes que les humains savent résoudre. Bien que les réseaux de neurones sont simulés la plus part du temps sur ordinateur séquentiel, ils sont théoriquement composés de cellules élémentaires ²neurones² interconnectées fonctionnant en parallèle et simulant l'activité d'une partie d'un cerveau humain.

1. Introduction

Comment l'homme fait-il pour raisonner, parler, calculer, apprendre, ...? Comment s'y prendre pour créer une ou de l'intelligence artificielle ? Deux types d'approches ont été essentiellement explorés :

- procéder d'abord à l'analyse logique des tâches relevant de la cognition humaine et tenter de les reconstituer par programme. C'est cette approche qui a été privilégiée par l'Intelligence Artificielle et la psychologie cognitive classique. Cette démarche est étiquetée sous le nom de *cognitivisme.*
- puisque la pensée est produite par le cerveau où en est une propriété, commencer par étudier comment celui-ci fonctionne. C'est cette approche qui a conduit à l'étude de réseaux de neurones formels. On désigne par *connexionnisme* la démarche consistante à vouloir rendre compte de la cognition humaine par des réseaux de neurones.

La seconde approche a donc menée à la définition et l'étude de réseaux de neurones formels qui sont des réseaux complexes d'unités de calcul élémentaire interconnectées. Il existe deux courants de recherche sur les réseaux de neurones : un premier motivé par l'étude et la modélisation des phénomènes naturels d'apprentissage à l'aide de réseaux de neurones; un second motivé par l'obtention d'algorithmes efficaces s'inspirant du fonctionnement biologique du cerveau. Nous nous plaçons du point de vue du second groupe. En effet, bien que les réseaux de neurones formels aient été définis à partir de considérations biologiques, pour la plupart d'entre eux, de nombreuses caractéristiques biologiques ne sont pas prises en compte dans ce chapitre. Toutefois, nous donnons, dans la suite de cette introduction, un rapide historique des réseaux de neurones. Nous présentons ensuite, un bref aperçu de quelques propriétés élémentaires de neurophysiologie qui permettent au lecteur de relier neurones réels et neurones artificiels. Enfin, nous donnons les différents types de réseau et les principales applications.

2. Historique

Des modèles de réseaux de neurones existent depuis longtemps. En 1890 : W. James, célèbre psychologue américain introduit le concept de mémoire associative, et propose ce qui deviendra une loi de fonctionnement pour l'apprentissage sur les réseaux de neurones connue plus tard sous le nom de loi de Hebb. Ainsi, le physiologiste viennois Sigmund Exner

proposait en 1894 un modèle neuronal de la détection du mouvement par l'œil de la mouche. Cependant, notre histoire commence vraiment dans les années quarante. En 1943, W. S. McCulloc et W. Pitts définissent le neurone formel comme élément binaire : le neurone est soit actif (état 1) soit inactif (état 0). Ils démontrent que tous ce qui est calculable par un homme ou une machine peut l'être en principe par une suite de manipulation de 0 et de 1, à l'aide d'un assemblage de neurones formels. Ce résultat a été largement utilisé, puisque tous nos ordinateurs utilisent le codage binaire.

En 1949, Donald Hebb est le premier à avoir clairement envisagé la possibilité de comprendre les comportements physiologiques à partir d'une théorie au niveau cellulaire. Il tentait d'expliquer les effets d'apprentissage de mémoire et de conditionnement à partir d'un groupe de cellules. Pour expliquer les effets d'apprentissage en fonction de l'expérience, il propose que les cellules apprennent à modifier l'intensité des connexions qui les relient en fonction de leur activité simultanée. Ils proposent deux idées fondamentales :

- Tous percept ou concept est physiquement représenté dans le cerveau par l'entrée en activité d'un ensemble de neurones (nous parlons de l'assemblé de Hebb).
- Deux neurones ou deux ensembles de neurones qui sont activés en même temps, vont finir par être « associés», de sorte que l'entrée en activité de l'un facilitera l'autre. Ainsi, la mise en mémoire s'effectue aux lieux d'interaction entre neurones, c'est-à-dire aux synapses (on parle de la synapse de Hebb). Le premier succès de Hebb est expérimentale : des études effectués sur l'Aplysie[1], ont démontré l'existence des modifications propriétés de synapses en fonction de l'activité neuronale. Il y a eu beaucoup d'autres expériences qui ont confirmé les deux mécanismes (renforcement ou affaiblissement des connexions suivant leurs utilités). la formalisation mathématique confirme ces principes, et les premières applications à l'informatique datent des années soixante **[Sen 05]**, **[Tou 92]**, **[Cor 03]**.

3. Fondement biologique

3.1 Le neurone biologique possède trois principales composantes : les dendrites, le corps cellulaire et l'axone (figure 16):

[1] Une Limace de mer (un petit animal).

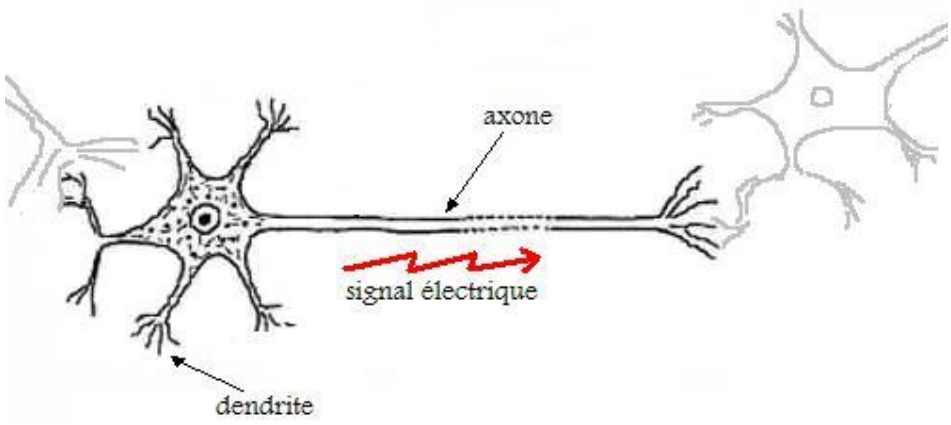


Figure 16 Le neurone biologique.

Les dendrites : Ce sont de fines extensions tubulaires qui se ramifient autour du neurone. Celles-ci sont parfois si nombreuses que l'on parle alors de chevelure dendritique ou d'arborisation dendritique. C'est par les dendrites que l'information est acheminée de l'extérieur vers le soma, corps du neurone. Leur taille est de quelques dizaines de microns de longueur **[Sen 05], [Tou 92], [Sci 07]**.

L'axone : L'information traitée par le neurone chemine ensuite le long de l'axone (unique) pour être transmise aux autres neurones. L'axone est plus long que les dendrites (sa taille peut varier entre quelques millimètres à plusieurs mètres) et se ramifient à son extrémité où il se connecte aux dendrites des autres neurones. En réalité, La transmission entre deux neurones n'est pas directe. En fait, il existe un espace intercellulaire de quelques dizaines d'Angstroms (10-9 m) entre l'axone du neurone afférent et les dendrites (on dit *une* dendrite) du neurone efférent. La jonction entre deux neurones est appelée la synapse **[Sen 05], [Tou 92]**.

Le corps cellulaire : est autour du noyau de la cellule (neurone), il se prolonge par un axone unique et comporte de nombreuse dendrites qui constituent son organe d'entrée **[Sen 05]**.

Les dendrites forment un maillage de récepteurs nerveux qui permettent d'acheminer vers le corps du neurone des signaux électriques en provenance d'autres neurones. Celui-ci agit comme une espèce d'intégrateur en accumulant des charges électriques. Lorsque le neurone devient suffisamment excité (lorsque la charge accumulée dépasse un certain seuil), par un processus électrochimique, il engendre un potentiel électrique qui se propage à travers son axone pour éventuellement venir exciter d'autres neurones. Il semble que c'est l'arrangement spatial des neurones et leur axone, ainsi que la qualité des connexions synaptiques individuelles qui déterminent la fonction précise d'un réseau de neurones biologique. C'est en se basant sur ces connaissances que le modèle mathématique décrit ci-dessus a été défini.

3.2 Le cerveau

Le cerveau se compose d'environ 10^{12} neurones (mille milliards), avec 1000 à 10000 synapses (connexions) par neurone. On ne dénombre que quelques dizaines de catégories distinctes de neurones. La structure du cerveau provient en partie des contacts avec l'environnement. L'apprentissage est donc indispensable à son développement **[Tou 92]**.

4. Le neurone formel

La figure ci dessous montre la structure d'un neurone artificiel. Chaque neurone artificiel est un processeur élémentaire. Il reçoit un nombre variable d'entrées en provenance de neurones amonts. A chacune de ces entrées est associée un poids w abréviation de weight (poids en anglais) représentatif de la force de la connexion. Chaque processeur élémentaire est doté d'une sortie unique, qui se ramifie ensuite pour alimenter un nombre variable de neurones avals. A chaque connexion est associée un poids **[Tou 92]**.

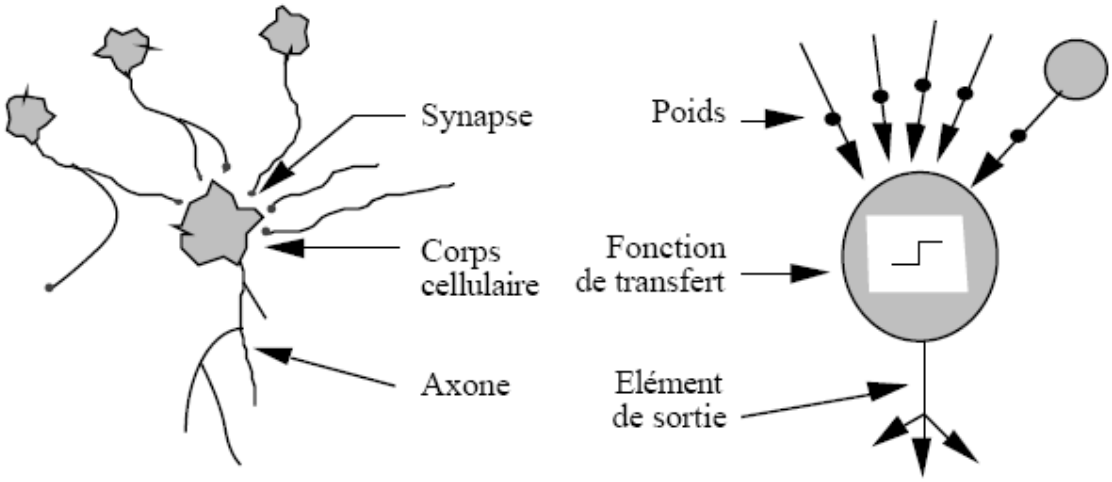


Figure 17 Structure d'un neurone biologique \artificiel.

4.1 Fonctionnement du neurone

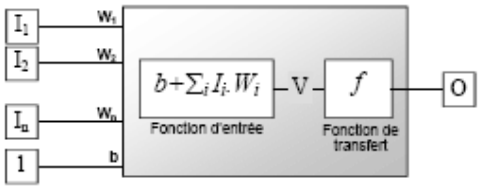


Figure 18 Fonctionnement d'un neurone artificiel.

Le neurone se compose de deux parties

- évaluation de la stimulation reçue (fonction E)
- évaluation de son activation (fonction f)

Il est caractérisé par :

- son état X (binaire, discret, continu)
- le niveau d'activation reçu en entrée U (continu)
- le poids des connections en entrée

La fonction d'entrée :

♣ Il Calcule la somme pondérée des signaux d'entrée selon l'expression suivante :

$$S = \sum I_i W_i$$

Où I_1, I_2, ..., I_N sont les entrées qui proviennent de l'environnement externe au réseau.

S est la sortie.

w_1, w_2, ..., w_N sont les poids associés à chaque connexion.

X est le vecteur d'entrée, *W'* est le vecteur poids.

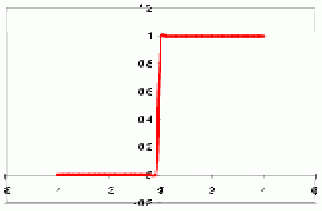
♣ Le biais d'entrée (bias input):

Unité fictive dont le poids permet de régler le seuil de déclenchement du neurone.

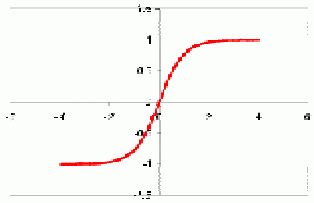
$$S = \sum I_i W_i + q$$

La fonction d'activation (ou **fonction de seuillage**, ou encore **fonction de transfert**)

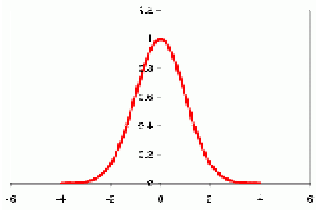
A partir de la valeur précédente (la somme pondéré), une fonction de transfert calcule la valeur de l'état du neurone (pour limiter l'amplitude de la valeur de sortie). C'est cette valeur qui sera transmise aux neurones avals. Il y a 3 types de fonctions d'activation **[SZC 06]**, **[Pel 05]** :



Fonction à seuil



Tangente hyperbolique



Fonction Gaussienne

Figure 19 les différents types de la fonction d'activation.

La sortie y du neurone devient $y = F(S)$.

Des exemples classiques de fonctions d'activation sont

Pas unitaire		$f(x) = \begin{cases} 0 \text{ if } 0 > x \\ 1 \text{ if } x \geq 0 \end{cases}$
Sigmoïde		$f(x) = \frac{1}{1+e^{-\beta x}}$
Linéaire Seuillée		$f(x) = \begin{cases} 0 & \text{if } x \leq x_{min} \\ mx+b & \text{if } x_{max} > x > x_{min} \\ 1 & \text{if } x \geq x_{max} \end{cases}$
Gaussienne		$f(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$
Identité		f (x) = x

Figure 20 des exemples de la fonction d'activation.

♣ Remarque : Le choix de cette fonction f se révèle être un élément constitutif important des réseaux de neurones. Ainsi, l'identité est rarement suffisante et des fonctions non linéaires et plus évoluées sont nécessaires.

4.2 Structure général d'un réseau de neurones formel

Par définition, un réseau de neurones est un maillage de plusieurs neurones, généralement organisés en couches. Il n'y a pas de connexion entre neurones d'une même couche et les connexions ne se font qu'avec les neurones des couches avales. Habituellement, chaque neurone d'une couche est connecté à tous les neurones de la couche suivante et celle-ci seulement. Nous distinguons dans un réseau de neurones une couche d'entrée contenant l'ensemble des neurones d'entrée, couche de sortie contenant l'ensemble des neurones de sortie, et des couches intermédiaires dites couches cachées n'ayant aucun contact avec l'extérieur (figure 21) **[Sci07], [Sen 05]**, **[Tou 92]**.

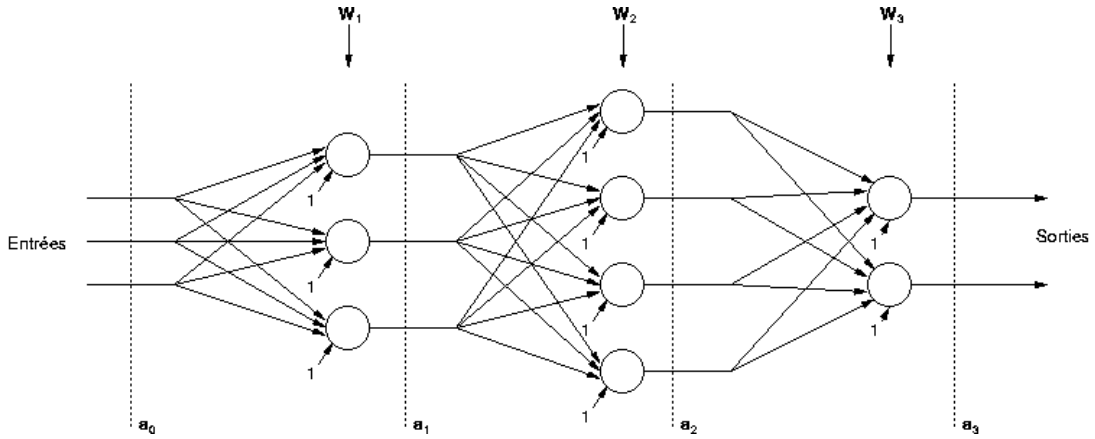


Figure 21 Structure de réseau de neurones.

4.3 Architecture des réseaux de neurones

Architecture est le terme le plus général pour désigner la façon dont sont disposés et connectés les différents neurones qui composent un réseau. On parle également de topologie (terme emprunté de la théorie des graphes). La topologie peut être quelconque, mais le plus souvent il est possible de distinguer une certaine régularité. La différence majeure porte sur la possibilité d'avoir des boucles dans le réseau (cycles ou circuits), ce qui permet au système d'avoir accès au passé. On notera la possibilité d'avoir des couches, ce qui permet le transfert des données dans le réseau de manière séquentielle (entre les couches) et parallèle (entre neurones). La figure **ci dessous** présente une taxonomie possible.

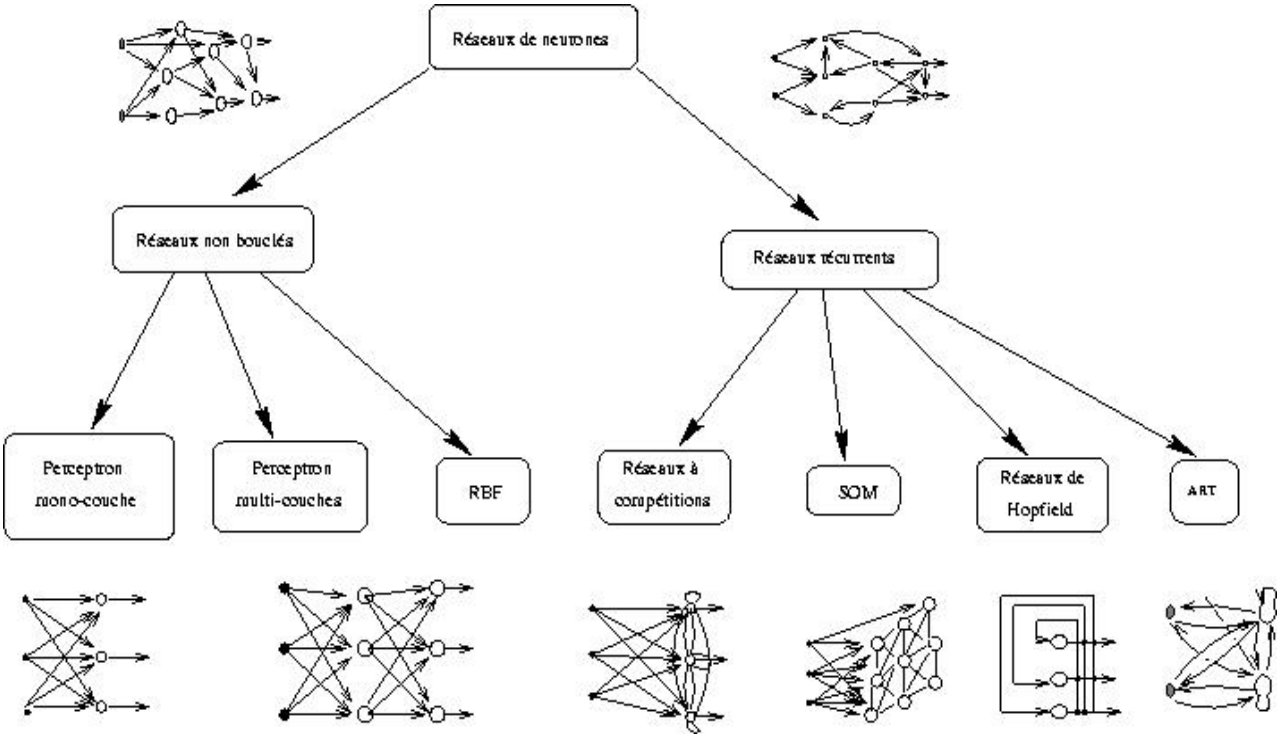


Figure 22 Une taxonomie possible.

4.4 Domaine d'application des réseaux de neurones (Les applications des réseaux de neurones)

Se trouvant à l'intersection de différents domaines (informatique, électronique, science cognitive, neurobiologie et même philosophie), l'étude des réseaux de neurones est une voie prometteuse de l'intelligence artificielle, qui a des applications dans de nombreux domaines :

- industrie : contrôle qualité, diagnostic de panne, corrélations entre les données fournies par différents capteurs, analyse de signature ou d'écriture manuscrite

Informatique : élimination du bruit, reconnaissance de formes (bruits, images, paroles), traitement du signal, le traitement de la vision et de la parole, compression de données, la robotique.

- Classification : diagnostic médical, vérification des signatures, reconnaissance d'image, reconnaissance de parole, l'évaluation du risque de l'emprunt (loan risk evaluation), data mining.

Modélisation et contrôle : contrôle des systèmes, composition de la musique,…

- finance : prévision et modélisation du marché (cours de monnaies...), sélection d'investissements, attribution de crédits.

Automobile : système de guidage automatique,…

Aérospatial : pilotage automatique, simulation du vol…

Défense : guidage de missile, suivi de cible, reconnaissance du visage, radar, sonar, lidar, traitement du signal, compression de données, suppression du bruit…

Electronique : prédiction de la séquence d'un code, vision machine, synthétiseur vocal, modèle non linéaire,…

- environnement : évaluation des risques, analyse chimique, prévisions et modélisation météorologiques, gestion des ressources **[Yze 05] [Din 06] [Szc 06].**

5 Apprentissage

Une caractéristique des réseaux de neurones est leur capacité à apprendre (par exemple à reconnaître une lettre, un son...). Mais cette connaissance n'est pas acquise dès le départ. La plupart des réseaux de neurones apprennent par des exemples en suivant un algorithme d'apprentissage.

Définition :

- *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 neuronal fait appel à des exemples de comportement.*
- L'apprentissage est la modification des poids des connexions du réseau dans l'optique d'accorder la réponse du réseau aux exemples et à l'expérience **[Dre 98]**, **[Tou 92]**. On distingue trois paradigmes d'apprentissage : supervisé, non supervisé et hybride.

5.1 Apprentissage Supervisé

Dans ce cas, on fournit au réseau de neurone la donnée à traiter mais aussi la réponse attendue. Le réseau effectue une évaluation de la donnée, puis compare la valeur obtenue avec la valeur désirée, il va ensuite modifier ses paramètres internes (poids et biais) afin de minimiser l'erreur constatée. Le réseau va se modifier jusqu'à ce qu'il trouve la bonne sortie, c'est-à-dire celle attendue.

5.2 Apprentissage Non Supervisé

On dispose d'un ensemble de données représentées par des vecteurs de grande dimension, l'apprentissage consiste à les regrouper selon des critères de ressemblance qui sont inconnus à priori. Dans ce type d'apprentissage, on ne connaît pas la catégorie de chaque forme fournie à l'entrée pour apprendre le réseau.

5.3 Apprentissage Hybride

Plus rare (et mal explorée) cette approche combine méthodes numériques (réseaux de neurones, algorithmes génétiques) et des méthodes symboliques. Certains auteurs utilisent le terme d'apprentissage hybride pour parler d'un couplage supervisé, non supervisé : dans ce cas, il s'agit d'un réseau qui met en parallèle ou en série un réseau entraîné en mode supervisé et un autre en mode non supervisé.

6. Les différents types des réseaux de neurones

6.1 Le perceptron monocouche

C'est historiquement le premier RNA, conçu en 1958 par Rosenblatt. C'est un réseau simple, puisque il ne se compose que d'une couche d'entrée et d'une couche de sortie. Il est calqué, à la base, sur le système visuel. 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 **[Cou 02] [Sen 05] [And 06].** Cette réponse suit la formule suivante :

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

6.1.1 Structure

Il ne dispose que de deux couches :

- une couche d'entrée qui s'appelle la rétine.
- une couche de sortie qui donne la réponse correspondante à la simulation présentée à l'entrée.

La figure suivante montre la structure de Perceptron monocouche, avec une sortie *y* et deux entrées *x1* et *x2* qui forment la rétine du réseau.

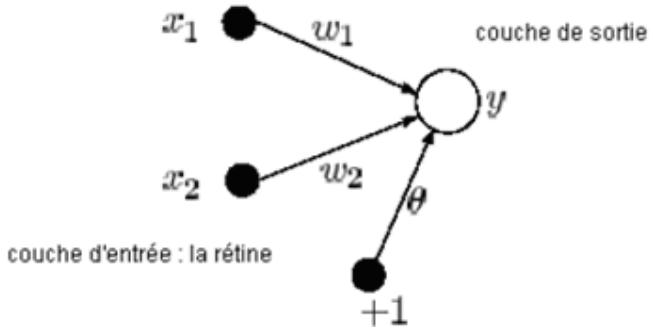


Figure 23 Structure de perceptron monocouche [And 06].

Fonction d'activation

Elle est linéaire

6.1.2 Apprentissage

Type d'apprentissage : Supervisé

Algorithme d'apprentissage

Il suit généralement un apprentissage supervisé selon règle de correction de l'erreur ou selon la règle de Hebb.

6.1.3 Domaines d'application

Connu en premier lieu dans la reconnaissance des formes, il peut être utilisé pour faire de la classification et pour résoudre des opérations logiques simples (telle "ET" ou "OU", "NAND", "NOR").

Limites

Il ne peut pas résoudre des problèmes non linéairement séparables.

6.2 Le perceptron multicouches

Le perceptron multicouches (noté le PMC) est une extension du précédent, il est sans doute le plus simple et le plus connu des réseaux de neurones. En général, la connectivité de ce type de réseau est totale intercouches adjacente, nulle sinon.

6.2.1 Structure

La structure est relativement simple : une couche d'entrée, une couche de sortie et une ou plusieurs couches cachées. Chaque neurone n'est relié qu'aux neurones de la couche précédente et de la couche suivante (excepté pour les couches d'entrée et de sortie) et il n'y a pas de connexions entre les cellules d'une même couche **[Cou 02].**

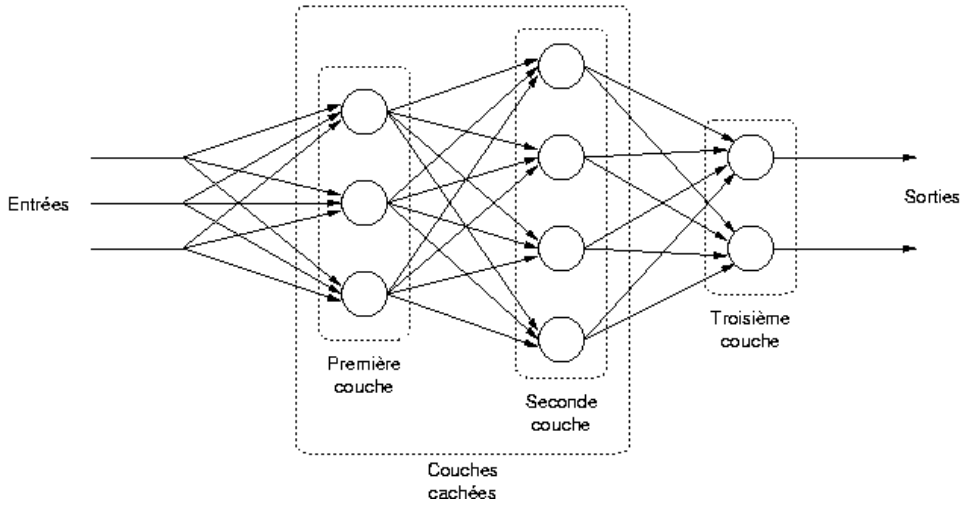


Figure 24 Structure de perceptron multicouches [Bec 02].

Types d'entrées

Les entrées du PMC peuvent être binaires, Entières ou réelles.

Fonction d'activation

Les fonctions d'activation utilisées dans ce type de réseaux sont principalement les fonctions à seuil ou sigmoïdes.

6.2.2 Apprentissage

Type d'apprentissage : L'apprentissage utilisé pour ce type de réseau est supervisé.

Algorithme d'apprentissage

Les PMC utilisent, pour apprentissage (modification de leurs poids), un algorithme de rétropropagation du gradient, qui est une généralisation de la règle de Widrow-Hoff (voir annexe). Il s'agit toujours de minimiser l'erreur quadratique. On propage la modification des poids de la couche de sortie jusqu'à la couche d'entrée (l'algorithme est décrit en détail dans le prochain chapitre, partie 2).

6.2.3 Domaines d'application

Les PMC agissent comme un séparateur non linéaire et peuvent être utilisés pour la classification, le traitement de l'image ou l'aide à la décision.

Avantages et inconvénients :

- meilleure capacité de généralisation, notamment sur des données bruitées.
- Risque de convergence vers un optimum local.
- Paramètres plus difficiles à régler **[tuf 04].**

6.3 Réseau à base radiale

L'architecture des RBF (pour Radial Basic Function) est la même que pour les PMC. Un neurone à base radiale est construit à partir d'une fonction du même nom. Au lieu de réaliser une somme pondérée de ses entrées, un tel neurone compare chaque entrée à une valeur de référence et produit une sortie d'autant plus grande (proche de 1) que les entrées sont proches des valeurs de références. Chaque entrée x_i est donc associée à une valeur c_i. La comparaison entre les deux jeux de valeurs se fait généralement au sens de la *norme euclidienne.* Plus précisément, le neurone commence par calculer la grandeur suivante

$$\|x - c\| = \sqrt{\sum_{j=1}^{m} (x_i - c_i)^2}$$

Il est parfaitement possible d'utiliser une autre norme que la norme euclidienne, et plus généralement une distance quelconque, pour comparer les vecteurs $x = (x_1, ..., x_m)$ et $c = (c_1, ..., c_m)$.

Le neurone transforme ensuite la valeur obtenue grâce à une fonction d'activation. Sa sortie est donc finalement donnée par :

$$j\ (\|x - c\|).$$

La fonction calculée par le neurone est dite à base radiale car elle possède une symétrie radiale autour du point de référence **C** : si on réalise une rotation quelconque autour de ce point, la sortie du neurone reste inchangée **[Tal 04].**

6.3.1 Structure

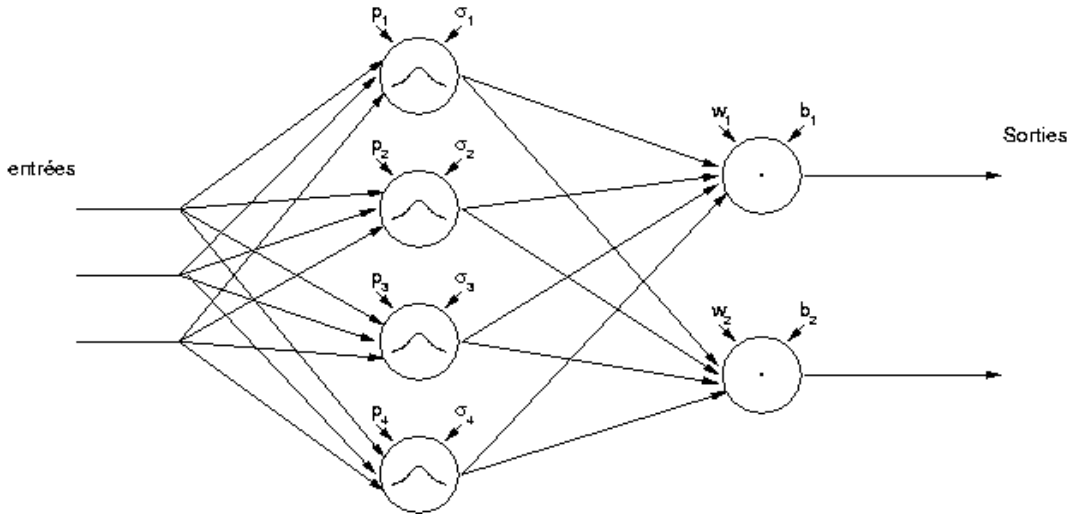


Figure 25 Structure de Réseau à base radiale.

6.3.2 La fonction d'activation

En pratique, il est très courant d'utiliser une fonction d'activation gaussienne définie par

$$\varphi(u) = \exp\left(-\beta u^2\right).$$

Le paramètre β peut être interprété comme l'inverse de la sensibilité du neurone : plus il est grand, plus il faut que les entrées soient proches des valeurs de références pour que la sortie du neurone soit proche de 1**[wik 06]**.

6.3.3 Apprentissage

Type d'apprentissage : L'apprentissage le plus utilisé est le mode hybride.

Algorithme d'apprentissage

Pour apprentissage, ils utilisent soit la règle de correction de l'erreur (voir annexe) soit, la règle d'apprentissage par compétition (voir annexe).

6.3.4 Domaine d'application

Les RBF sont utilisés dans les mêmes types de problèmes que les PMC, en classification et en approximation de fonction.

Avantages et inconvénients :

- Apprentissage plus rapide.
- Plus facile de trouver les bons paramètres.
- Moins bonne capacité de généralisation, surtout si l'échantillon d'apprentissage ne configure pas toutes les configurations possibles **[Tuf 04].**

6.4 Réseau de Kohonen

Ce modèle appartient à la classe des réseaux à compétition. Les neurones de la couche de sortie entrent en compétition, de telle façon qu'habituellement, un seul neurone de sortie est activé pour une entrée donnée. Cette compétition entre les neurones est réalisée grâce à des connexions latérales inhibitrices.

6.4.1 Structure

La structure du réseau de Kohonen est constituée de deux couches. La première, appelée couche d'entrée, la deuxième est la couche de sortie, ou la couche compétitive, elle est composée d'un certain nombre de neurones régulièrement répartis sur une carte, où chaque neurone de la couche compétitive est connecté à toutes les entrées. Chaque connexion d'un

neurone d'entrée **j** vers un neurone de sortie **i** a le vecteur poids $\mathbf{W_{i,j}}$. Ainsi, chaque neurone de sortie i a le vecteur poids $W = [W_{m,1}, W_{m,2}, \ldots, W_{m,N}]^T$.

Chaque neurone de la couche de sortie est donc caractérisé par sa position relative sur la carte et par son vecteur poids dans l'espace de représentation ″carte″ (figure 26) **[Lem 01].**

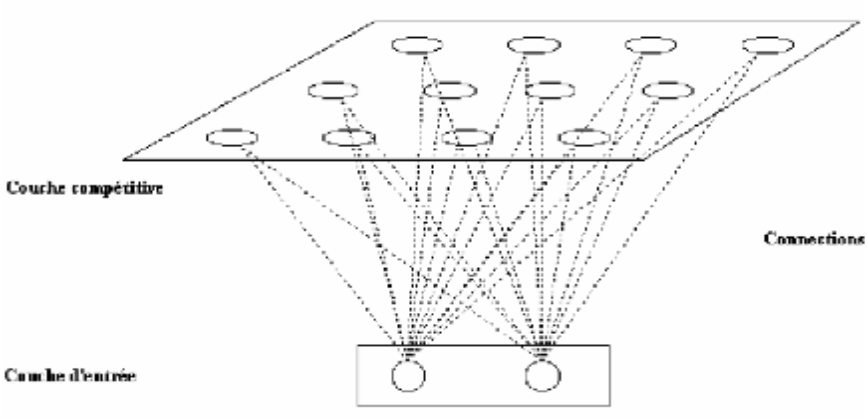
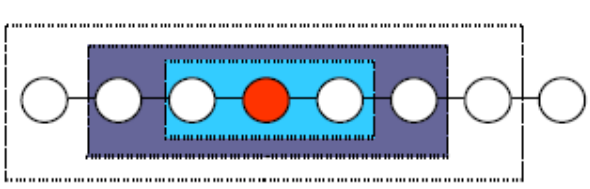
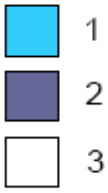


Figure 26 Structure de Kohonen.

La structure topologique de la carte introduit la notion de voisinage et de distance entre les neurones de la carte. Cette dernière a différentes topologies et voisinage illustrés dans les figures ci-dessous

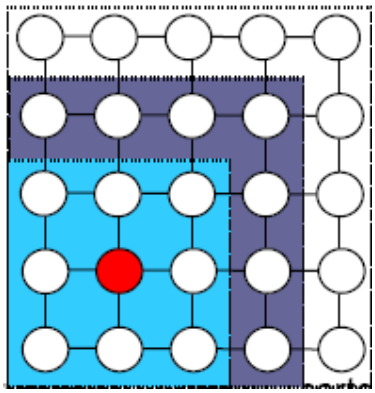
- Carte unidimensionnelle ou fil ou ficelle



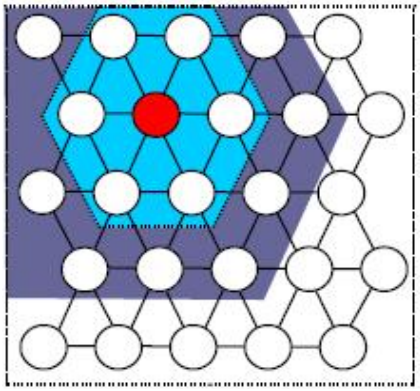


Rayon de voisinage

- Carte bidimensionnelle ou grille (carrée, rectangle)



Voisinage rectangulaire



Voisinage hexagonal

- Autres structures, cylindre, carte 3D.

Figure 27 les différentes topologies et voisinage de la carte [Lem 01].

La fonction d'activation

6.4.2 Apprentissage

Type d'apprentissage : Apprentissage non supervisé.

Algorithme d'apprentissage

L'algorithme utilisé est l'apprentissage auto organisation, il met en correspondance l'espace des entrées et la carte, en adaptant des poids *W* de telle manière que des exemples proches dans l'espace d'entrée sont associés au même neurone ou à des neurones proches dans la carte.

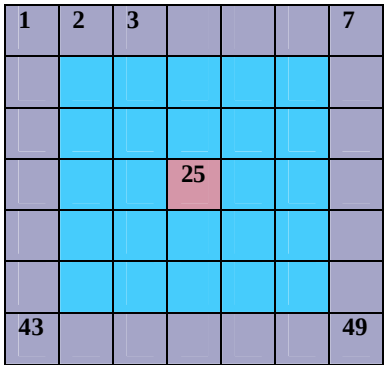
Matérialisation du voisinage : On définit le voisinage de rayon **r** d'un neurone $\mathbf{n_0}$, noté $\mathbf{V_r(n_0)}$, comme l'ensemble des neurone **n** situé sur le réseau à une distance inférieure ou égal à **r**. En utilisant les coordonnées cartésiennes, on peut définir la distance d par :

$d(n, n_0) = Max(|i - i_0|, |j - j_0|)$ pour une grille.

$d(n, n_0) = |i - i_0| = |n - n_0|$ pour une ficelle. Où d désigne la distance euclidienne.

Dans les deux cas, $V_r(n_0) = \{ n \in \{1, ..., N\} / d(n, n_0) \leq r \}$.

Exemple voila deux types de voisinage



Par rapport la grille :

$V_2(36)$ ={9 ,10 ,11 ,12,13,16,17,18,19 ,20,23,24,25,26,27,30,31,32,33,34,37,38,39,40,41}.

Par rapport la ficelle :

$V_1(5) = \{4,5,6\}$.

Le vecteur poids **W** (initialisé aléatoirement) et le vecteur d'entrée **x** doivent être normalisés.

Pour chaque neurone **i** (de la carte), on calcule son activation $\mathbf{a_i}$ telle que :

$$a_i = \sum_j W_{ij}\, x_j$$

On sélectionne le neurone k qui a l'activation maximale (appelé neurone gagnant) :

$\forall i \neq k$: $\quad a_i \leq a_k$

Les nouvelles valeurs des activations deviennent comme suit :

$a_k = 1$ et $a_i = 0$, $i \neq k$.

On procède à la modification du vecteur des poids du neurone gagnant **k** et ses voisins :

$$W_{i(nouveau)} = \frac{W_i + h\,(x - W_i)}{\left\| W_i + h\,(x - W_i) \right\|} \qquad \forall\, i \in V_r(k)$$

Où le diviseur assure la normalisation du nouveau vecteur W_i. Tandis que les vecteurs poids des autres neurones restent inchangés.

Si les vecteurs d'entrée et de poids ne sont pas normalisés, l'algorithme d'apprentissage devient comme suit :

Initialisation aléatoire des vecteurs des poids.

A chaque étape t, t=1,…, T (T étant le nombre d'itérations, tel que T=P*N avec un P un entier très grand et N étant le nombre de neurones) .

On présente le vecteur d'entrée x(t) extrait au hasard de l'ensemble des exemples.

Pour chaque neurone i, on calcule on calcule la distance euclidienne entre $W_i(t)$ et x(t) :

$$D_i = \sqrt{\sum \left(x_i(t) - W_{ij}(t)\right)^2}$$

On détermine le neurone gagnant (k) qui a la distance minimale : D_k=min (Di) i=1…N

Les nouvelles valeurs d'activation deviennent comme suit : a_k =1 et a_i =0, i≠k.

On détermine les voisins du neurone k suivant la règle de voisinage à l'instant t

On modifie le vecteur des poids du neurone k et de ses voisins par les transformations suivantes $W_i(t+1) = W_i(t) + h(t) + (x(t) - W_i(t)) \quad \forall i \in V_r(t)(k)$

$W_i(t+1) = W_i(t)$ pour les autres neurones.

Où η est le pas d'apprentissage qui décroît après chaque itération (on peut choisir par exemple $\eta = \frac{1}{t}$) .

6.4.3 Domaine d'application

Classification des modèles et optimisation des problèmes, simulation.

6.4.4 Inconvénient des cartes auto adaptatives

Le voisinage dans les cartes auto adaptatives est malheureusement fixe, et une liaison entre neurones ne peut être cassée même pour mieux représenter des données discontinues. Les *Growing Cell Structure*, ou *Growing Neural Gas* sont la solution à ce problème. Les neurones et leurs liaisons peuvent être supprimées ou ajoutées quand le besoin s'en fait sentir **[wik 06]**.

6.4 Le réseaux de Hopfield

Le modèle de Hopfield fut présenté en 1982. Ce modèle est très simple basé sur le principe des mémoires associatives. C'est d'ailleurs, la raison pour laquelle ce type de réseau est dit associatif. Outre un intérêt pratique, ce réseau admet une analyse théorique précise et complète **[Sen 05].**

6.4.1 Structure

Les réseaux de Hopfield sont des réseaux récurrents et entièrement connectés (voir figure 28). 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.

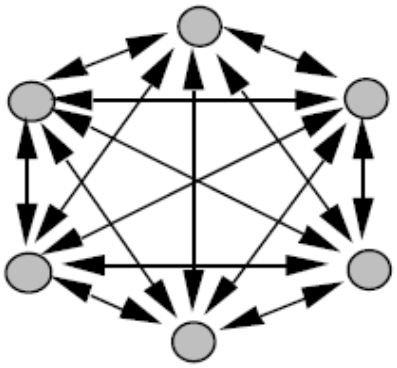


Figure 28 Structure des réseaux Hopfield.

Description du fonctionnement du réseau Hopfield

L'architecture du réseau Hopfield est semblable à celle d'une mémoire auto-associative, il se différencie de cette dernière par : la réponse est binaire, et la réponse des cellules est asynchrone.

Les valeurs d'entrée sont binaire (-1, 1) (le modele Ising) mais peuvent être aisément remplacées par les valeurs binaires usuelles (0, 1) en utilisant une simple transformation $A_{(-1, 1)} = 2.A_{(0, 1)}-1$.

L'activation a_j du neurone j se calcule comme suit : $a_j = \sum_i^N x_i W_{ij}$ avec :

x_i : la valeur de sortie du neurone i.

W_{ij} : poids de la connexion entre les neurones i et j.

On calcule la valeur de sortie du neurone j en bridant son activation à l'aide de la fonction signe où $x_j = \mathrm{sgn}(a_j) = \begin{cases} -1 \ pour \ a_j < 0 \\ +1 \ pour \ a_j >= 0 \end{cases}$

On peut utiliser un seuil $x_j = \mathrm{sgn}(a_j) = \begin{cases} -1 \ pour \ a_j < q \\ +1 \ pour \ a_j >= q \end{cases}$

6.4.2 Apprentissage

Type d'apprentissage : L'apprentissage utilisé est non supervisé.

Algorithme d'apprentissage

La règle d'apprentissage proposée par Hopfield est basée sur la règle de Hebb. La règle de Hebb consiste à forcer les poids des liaisons entre les neurones actifs au même moment. Par contre, les poids seront diminués si les neurones sont dans des états contraires. Dans le cas de Hopfield, cette règle est légèrement étendue si l'on considère que deux neurones dans l'état -1 sont actifs. La règle de modification des poids devient donc :

$W_{ij} = \sum V_i^p \ V_j^p$ avec $W_{ii = 0}, W_{ij} = W_{ji}$

W_{ij} : le poids entre les neurones i et j

P : le nombre d'exemples présenté au réseau

V_i^p : le i^{eme} élément du p^{eme} exemple (vecteur)

On remarque que la phase d'apprentissage est immédiate en calculant directement les poids à l'aide de cette fonction. C'est d'ailleurs, l'un des réseaux qui permet un calcul aussi simple et analytique des poids.

6.4.3 Domaine d'application

Il est utilisé généralement pour les problèmes d'optimisation, association des modeles.

7. Les avantages et les inconvénients des réseaux de neurones :

Avantages

- Les réseaux neuronaux ont montré leurs efficacités dans le domaine de la reconnaissance. ils ont à l'heure actuelle été capables de faire aussi bien que les programmes classiques dans ce domaine, et pour certains problèmes spécifiques peuvent aborder des reconnaissances de formes pour lesquelles les approches classiques n'ont pas de solution.
- Un autre avantage des réseaux neuronaux est leur flexibilité face aux variations de l'environnement à analyser, en d'autres termes leur capacité à s'adapter à la nouveauté. Ils peuvent travailler sur des signaux extrêmement dégradés.
- De plus, cette robustesse s'applique aussi à la dégradation interne du fonctionnement du réseau neuronal. A l'opposé d'un programme classique où la moindre virgule manquante peut faire tout arrêter, un réseau neuronal peut encore donner des résultats valables quoique avec une certaine perte de précision.
- Simple à manier, beaucoup moins de travail personnel à fournir que dans l'analyse statistique classique. Aucune compétence en math, informatique ou statistiques requise.
- Comportement moins mauvais en cas de faible quantité de données.
- Aptitude à modéliser des structures complexes et des données irrégulières.
- Aptitude à modéliser des problèmes très variés **[Tuf 04], [Sen 05].**

Inconvénients

- Bien sûre, le RN ne dispense pas de bien connaître son problème, de définir ses classes avec pertinence, de ne pas oublier des variables importantes, etc. Surtout le RN est une boite noir qui n'explique pas ses décisions.
- Résultats totalement non explicites.
- Convergence vers la bonne solution pas toujours assurée.
- Difficulté d'utilisation correcte « paramètres nombreux et délicat à régler » **[Tuf 04]**, **[Sen 05].**

8. Conclusion

Dans ce chapitre, nous avons présenté un aperçu général sur les réseaux de neurones en commençant par une description biologique qui a permis de souligner l'écart entre le modèle et la réalité, par exemple au niveau du nombre d'éléments impliqués ou de leur complexité. Ensuite nous avons présenté des architectures classiques de réseaux de neurones, PMC, RGB, Kohonen, Hopfield. Où nous nous sommes intéressés à leurs structures et leur principe de fonctionnement, et nous avons expliqué leurs modes d'apprentissage et avons cité aussi les cas d'utilisation de chacune. Enfin nous décrivons les apports non négligeables des réseaux de neurones qui ont marqué les travaux menés dans différents domaines sans oublier bien sure leurs inconvénients.

Comment nous proposons d'utiliser les réseaux de neurones pour résoudre le problème étudié dans notre thèse ? Ça va être abordé dans le chapitre suivant.

Dans cette partie, nous allons présenter notre domaine d'intérêt (application envisagé), à savoir celui de la reconnaissance de l'écriture manuscrite (REM). Le but de notre projet étant la sélection d'attributs pour améliorer les performances du système de reconnaissance de caractères manuscrits. Nous terminerons en exposant la direction choisie pour REM.

1. Introduction

De nos jours, l'écriture est toujours le moyen de communication visuelle le plus utilisé par l'homme. Il n'est donc pas surprenant de voir que de nombreux travaux scientifiques portent sur sa reconnaissance automatique. L'écriture est en fait la réalisation d'un message à transmettre, c'est-à-dire la représentation physique d'un contenu sémantique. Le média ou support généralement utilisé est le papier. Le but de la reconnaissance de l'écriture est de prendre une décision quant au contenu sémantique du message transmis à partir de sa représentation physique. Les applications de systèmes capables de remplir cette tâche sont nombreuses ; nous pouvons citer entre autres la lecture automatique de bons de commande, le traitement automatique des chèques, la vérification de signatures ou encore le tri automatique du courrier, la récupération du vieux manuscrit pour enregistrement sur des bases de données que tout le monde puisse interroger.

On peut distinguer deux axes de recherche : la *reconnaissance de l'écriture imprimée* et la *reconnaissance de l'écriture manuscrite*. Dans le cas de la reconnaissance de l'écriture manuscrite, il existe deux champs d'applications suivant le mode de saisie. Si le mode de saisie est dynamique, on parle de reconnaissance en temps réel appelée *en ligne* ou *on-line* (Les caractères sont reconnus au moment même où ils sont écrits. L'information est dynamique et monodimensionnelle, et consiste en une séquence de traits qui suivent une échelle temporelle **[Gos 96]**). Dans le cas de la saisie statique on parle alors de reconnaissance en temps différé appelée *hors-ligne* ou *offline*. La reconnaissance hors-ligne s'applique une fois que l'écriture est sur un support papier qui est numérisé puis enregistré sur un format d'image (l'information ici est bidimensionnel). Cette image contient des pixels soit de type binaire (pixels noirs et blancs) soit de type entier (image niveaux de gris). Cette reconnaissance hors-ligne regroupe deux thèmes : *la reconnaissance de caractères manuscrits isolés* (numériques ou alphanumériques) et *la reconnaissance de caractères continus* ou *reconnaissance de mots* (elle nécessite une phase de segmentation préliminaire à la classification). La figure ci dessous présente une classification des différents types d'écriture faite selon le média, le support de saisi et l'application considérée.

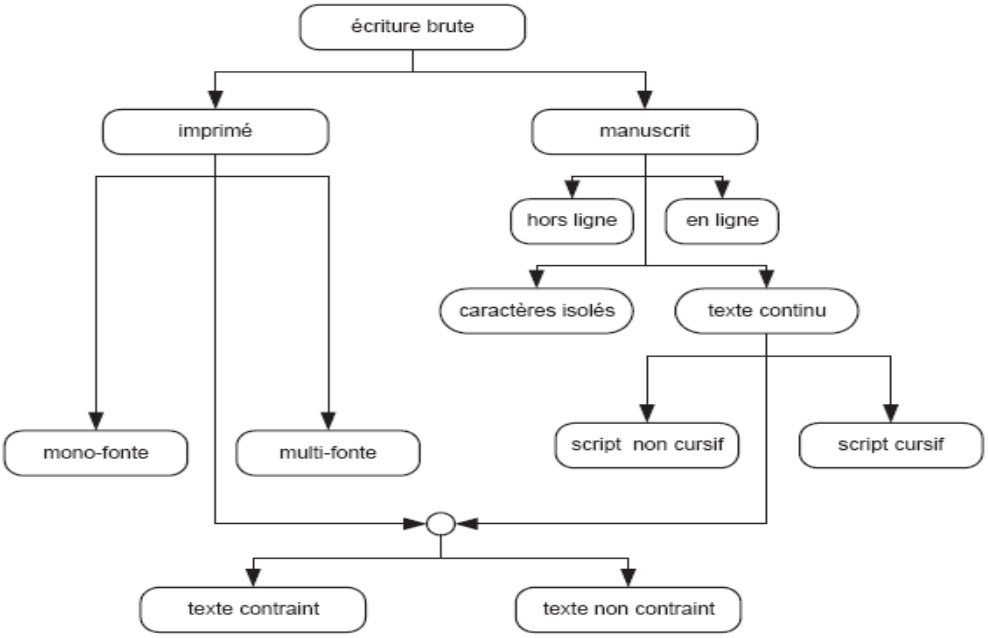


Figure 29 Dichotomie de types d'écritures [aya 04]

Nous nous intéressons dans notre étude à la reconnaissance hors ligne de caractères manuscrits isolés. Pour cela nous devons voir les caractéristiques d'un système de reconnaissance de caractères (OCR).

2. Système de reconnaissance optique de caractères

La reconnaissance optique de caractères (***OCR***) est un procédé permettant d'effectuer la transformation d'un texte sous forme image en un texte sous forme d'un fichier informatique de façon rapide et automatique (la transcription textuelle de l'image de caractère) **[Hal 04]**. Elle consiste en général en un ensemble de modules d'acquisition d'image, prétraitements, extraction de caractéristiques (ou attributs), classification et post-traitement, voir figure ci dessous.

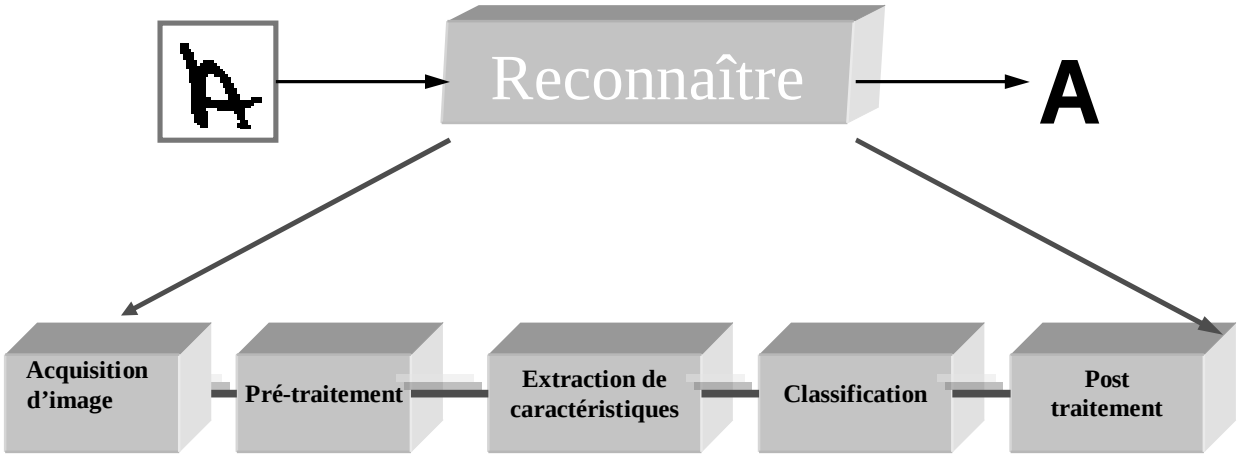


Figure 30 Le Schéma du système de reconnaissance de caractères.

2.1 Acquisition d'images

L'écriture est digitalisée par un scanner et elle est transformée en une image. Cette dernière est l'entrée du système de reconnaissance.

L'acquisition d'images est assez simple mais très importante car elle influence sérieusement le reste des étapes. Il y a deux paramètres importants :

- Résolution : la résolution normale est 300 dpi. Pourtant, quand la taille de l'écriture est petite, il faut augmenter la résolution.
- Niveau d'éclairage : si on ajuste le scanneur pour que l'image soit plus claire, le bruit est réduit mais des traits minces disparaissent aussi **[Ngh 05]**.

2.2 Prétraitement

L'étape de prétraitement est une des premières taches d'un système de reconnaissance après l'étape d'acquisition. Cette étape n'est pas spécifique à la reconnaissance du manuscrits mais fait partie de tout système de reconnaissance de forme, visant à améliorer la qualité de l'information pour les phases de traitement qui vont suivre.

Plusieurs prétraitements préliminaires sont nécessaires à la reconnaissance de caractère, parmi lesquelles on peut noter : binarisation, élimination du bruit, normalisation.

- ***Binarisation***

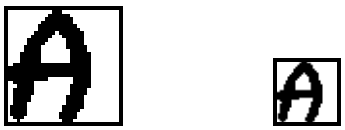
Cette étape de prétraitement consiste à convertir les images numérisées de la base de données en des images binaires. En général, on utilise un seuil de binarisation approprié qui traduit la limite des contrastes fort et faible dans l'image. Mais pour des images peu contrastées ou à contraste variable, il est difficile de fixer ce seuil à une valeur précise.

- ***Elimination du bruit***

Consiste à éliminer ou tout au moins réduire au maximum le bruit (salissures, froissure, trous).

- ***Normalisation***

La taille des caractères varie énormément selon le style d'écriture de chaque personne. Pour faciliter la reconnaissance, l'algorithme de normalisation doit intervenir, dans le but de ramener toutes les images à la même taille (taille standard) tout en conservant leurs particularités géométriques. Cette phase est indispensable surtout pour les réseaux de neurones **[Bel 02]** (voir figure).



(50×50) (27×27)

Figure 31 la normalisation d'une image caractère.

L'algorithme de normalisation **[Sen 05]** est décrit dans comme suit

1. On calcul le centre de gravité (ou la médiane) (x_m, y_m) du caractère.

$$x = \frac{x_{\max} + x_{\min}}{2}$$

$$y = \frac{y_{\max} + y_{\min}}{2}$$

2. La largeur sur les deux dimensions est calculée :

$$d_x = \frac{x_{\max} - x_{\min}}{2}$$

$$d_y = \frac{y_{\max} - y_{\min}}{2}$$

Le maximum de ces deux valeurs est choisi comme facteur d'échelle :

$$d = \max(d_x - d_y)$$

3. On calcul les nouvelles coordonnées de chaque pixel :

$$x_t = \frac{k}{2} + \frac{k}{2}\left(\frac{x_t - x_m}{d}\right)$$

$$y_t = \frac{k}{2} + \frac{k}{2}\left(\frac{y_t - y_m}{d}\right)$$

Où k est la taille standard

Remarque : Si la taille fixée est très petite, on peut perdre de l'information, si elle est très grande, l'étape de reconnaissance va opérer lentement **[Ngh 05]**.

- ***Squelettisation***

La procédure de squelettisation s'effectue sur une image binaire, et a pour but de réduire l'épaisseur du tracé d'un caractère à un pixel seulement, tout en conservant la continuité, topologie de celui-ci. Le principe de cette procédure est d'effectuer une succession d'opérations d'érosion conditionnelle, jusqu'à ce que le but recherché soit atteint. Une opération d'érosion consiste à éliminer dans une image tous les pixels d'intensité non nulle qui sont adjacents aux pixels de l'arrière-plan (élimination des pixels appartenant au corps même du caractère de manière itérative, tant que cela ne rompt pas la continuité du tracé.). L'érosion conditionnelle signifie qu'un pixel d'intensité non nulle ne peut être éliminé que si, une fonction logique déterminée des valeurs des pixels qui l'entourent, est vérifiée. Cette fonction logique a pour rôle de vérifier la participation ou non du pixel central au maintien de la continuité du tracé.

La procédure de squelettisation requiert un balayage de l'image en deux passes alternatives, de gauche à droite et de haut en bas d'abord, et de haut en bas et de gauche à droite ensuite (*figure B.1*), au moyen d'une fenêtre 3×3 . Le pixel du centre de cette fenêtre prend alors une valeur déterminée par une fonction logique de l'ensemble des 9 points de la fenêtre. C'est cette fonction qui assure le maintien de la continuité du tracé (voir figure) **[Gos 96]**.

Figure 32 Chiffre 2 et son squelette.

2.3 Extraction de caractéristiques (attributs)

L'objectif commun de tout les attributs ou caractéristiques est de caractériser au mieux la forme des caractères afin de pouvoir distinguer si deux images appartiennent à deux classes différentes ou à la même classe, c'est–à–dire qu'elles doivent diminuer la variabilité intra-

classe et augmenter la variabilité inter-classe **[Che 06]**. L'extraction de caractéristiques en reconnaissance de l'écriture est confrontée au grand problème de la variabilité intra-classe. En effet, d'un point de vue visuel, un caractère peut prendre différentes formes, en fonction de sa position dans le mot. Cependant, les plus grandes variations sont introduites par le scripteur. L'écriture étant propre à chaque individu, le tracé résultant de l'écriture d'un même caractère par deux personnes peut être bien différent. De plus, pour un même scripteur, un certain nombre de contraintes influencent la réalisation du tracé de son écriture. Nous pouvons citer entre autre l'outil, le support et même l'humeur de l'individu. Dans la littérature, il existe un grand nombre de travaux concernant l'extraction de caractéristiques pour la reconnaissance de l'écriture, et qui sont classées en deux catégories par rapport les attributs extraits : attributs structurelles (ou attributs locales) basées sur une représentation linéaire du caractère et les Attributs statistiques (ou attributs global).

2.3.1 Attributs structurels ou locales

Ces primitives sont des objets de la forme tel que les extrémités, les croisements de traits, les boucles et les courbes. L'inconvénient de ces primitives est que leur extraction nécessite une squelettisation préalable du caractère, puisque l'épaisseur du trait ne contient pas d'information. Néanmoins ce sont des primitives très robustes vis à vis de la rotation, translation, homothétie. Dans cette catégorie, il existe 4 familles de caractéristiques : intersections avec des droites, arcs concaves et occlusions, extrema et jonctions **[Ben 02]**.

2.3.2 Attributs statiques

Ces primitives sont dérivées de la distribution des pixels. Heutte et al suggèrent 3 familles de caractéristiques telles que : les moments invariants, les projections, et les profils, zonage. Elles sont extraites en considérant la distribution des pixels noirs de l'objet (caractère, mot, chiffres) **[Ben 02]**.

2.3.3 Quelques exemples de méthodes d'extraction de caractéristiques

Parmi les Méthodes d'extraction de caractéristiques nous citons :

2.3.3.1 Projection

La projection sur un axe revient à compter en chaque point de projection le nombre de pixels 'noirs' dans la ligne ou la colonne qui lui fait face.

Projection verticale la projection verticale d'une image I représentant un caractère donné est dénoté par **[Sar 99]** :

$$P_y^v = \frac{1}{N}\sum_{i=1}^{N} I_{ij} \quad \textbf{\textit{où}} \quad P^v = \left[P_1^v \;\; P_2^v, ..., \;\; P_N^v\right]$$

Projection horizontale La projection horizontale d'une image I représentant un caractère donné est dénotée par **[Sar 99]** :

$P_x^h = \frac{1}{N}\sum_{j=1}^{N} I_{ij}$ ***où*** $P^h = [P_1^h \ P_2^h, ... , \ P_N^h]$ où (N×N) est la taille de l'image de caractère.

Projection sur la diagonale (45°)

La projection sur la diagonale 45° d'une image I = (i_{xy}) de dimension (N×N) représentant un caractère C est dénotée par **[Sen 05]**:

$$P^{d1} = [P_1^{d1}, P_2^{d1}, ..., P_{2N-1}^{d1}] \qquad \text{Où :}$$

$$P_m^{d1} = \begin{cases} \sum_{l=N-m+1}^{N} \sum_{k=1}^{m} i_{lk} & 1 \le m \le N \quad et \quad l = k + N - m \\ \sum_{l=1}^{2N-m} \sum_{k=m-N+1}^{N} i_{lk} & N+1 \le m \le 2N-1 \quad et \quad l = k + N - m \end{cases}$$

Projection sur le diagonale 135°

La projection sur la diagonale 135° d'une image I = (i_{xy}) de dimension (N×N) représentant un caractère C est dénotée par **[Sen 05]**:

$$P^{d2} = [P_1^{d2}, P_2^{d2}, ..., P_{2N-1}^{d2}] \qquad \text{Où :}$$

$$P_m^{d1} = \begin{cases} \sum_{l=1}^{m} \sum_{k=1}^{m} i_{lk} & 1 \le m \le N \quad et \quad k = m - l + 1 \\ \sum_{l=m-N+11}^{N} \sum_{k=m-N+1}^{N} i_{lk} & N+1 \le m \le 2N-1 \quad et \quad k = m - l + 1 \end{cases}$$

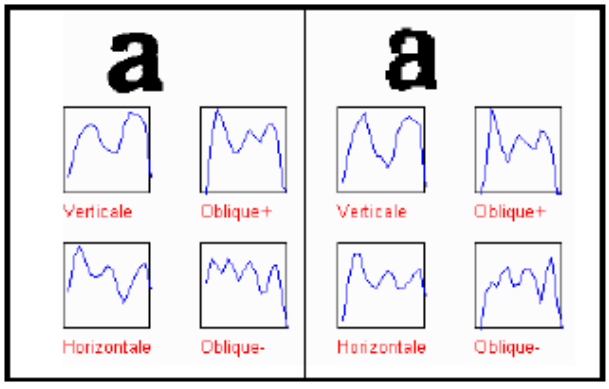


Figure 33 Projections sur 4 directions 0°, 45°, 90°, 135° [Hen 99].

Inconvénients

Des ambiguïtés peuvent être présentés, si on ne considère qu'une direction (("u","n") en projection verticale ou ("p","q","F") en projection horizontale). Les ambiguïtés en utilisant les deux directions principales horizontale et verticale sont plus rares, mais elles concernent les caractères qui possèdent majoritairement des traits obliques ("S","5" ; "2","Z" ; "X","K","Y" ; ...) ; d'où la nécessité d'utiliser les projections obliques à 45° et 135° même si leur utilité ne peut être prouvée de façon formelle **[Hen 99]**.

Les substitutions possibles comme ("h","b") ("s","8","a") démontrent les limites de cette mesure. En effet, les projections ne donnent que très peu d'informations sur le nombre d'intersections et sur la position relative des traits alignés côte à côte. Ceci induit un nombre élevé de substitutions du type "h" et "b" ou "8" et "a" qui présentent les mêmes projections horizontales et verticales. Pour compléter les informations apportées par les projections, il faut réintroduire la notion d'intersection ; ce qui est fait avec les projections partielles **[Hen 99]**.

2.3.3.2 Le contour

Le contour est extrait en parcourant la frontière intérieure de la forme du caractère. Il contient une information structurelle suffisamment complète pour décrire un caractère. Cette information est très sensible à des déformations et à des bruits dus à la numérisation,

Jean-Luc Henry qui a fait une étude sur le choix des caractéristiques pour la représentation de caractères remarque que pour le contour, les couples de caractères classiques comme ("5","S") et ("2","Z") présentent les taux de substitution les plus élevés, des erreurs très grossières subsistent comme ("e","c"), ("l","j") et même sur des formes totalement différentes ("m","h","k") par exemple.

Le contour est un très bon opérateur pour différencier des formes, mais les substitutions sont très nombreuses et proviennent essentiellement de la rupture des tracés ou de la jointure des traits (Un même caractère Peut présenter des contours radicalement différents si la structure est modifiée.) (Figure 32). De même, les formes composées nécessitent un prétraitement pour repérer la forme principale. Un même caractère Peut présenter des contours radicalement différents si la structure est modifiée **[Hen 99]**.

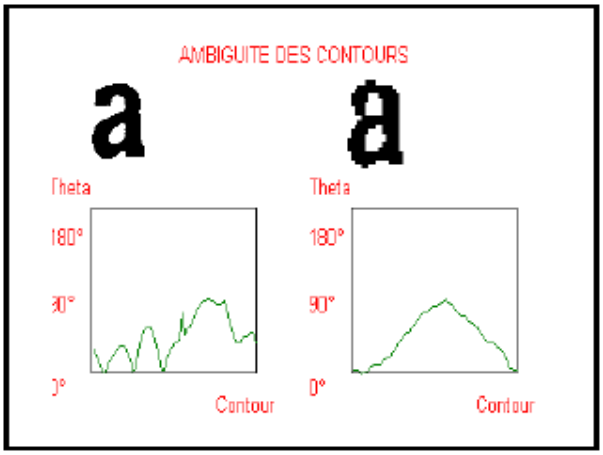


Figure 34

2.3.3.3 Profils

Ce type de caractéristiques fait ressortir l'allure générale selon l'angle de vision prise. Un point du profil est défini comme étant le premier point du caractère rencontré dans la ligne de vue **[Sar 99]**. Les profils externes d'une forme, très utilisés pour le manuscrit et les caractères chinois, sont sûrement des mesures plus robustes que le contour, mais on doit s'attendre à un pouvoir discriminant moindre puisque les cas d'ambiguïtés sont nombreux.

On obtient les profils haut, bas, gauche et droit du caractère en mesurant la distance séparant chaque pixel du caractère au bord du cadre (figure).

Le profil est dit consolidé dans la mesure où l'on effectue un calcul d'histogrammes sur 16 valeurs, de chaque coté, de façon à obtenir un vecteur de caractéristiques de 64 composantes.

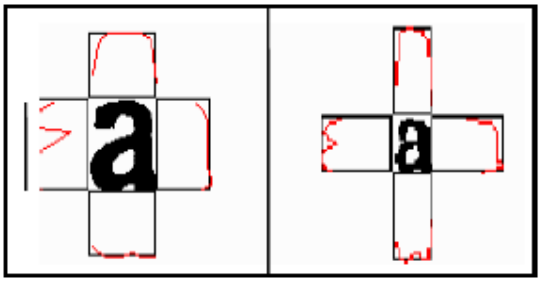


Figure 35 Profils nord, sud, est et ouest des caractères "a". Les profils ont peu varié bien que le deuxième "a" présente une liaison.

Inconvénient

Jean-Luc Henry a trouvé que différents caractères possèdent des profils presque équivalents. Par exemple, les caractères "e" et "c" substituables par leurs contours externes restent encore ambigus (Figure).

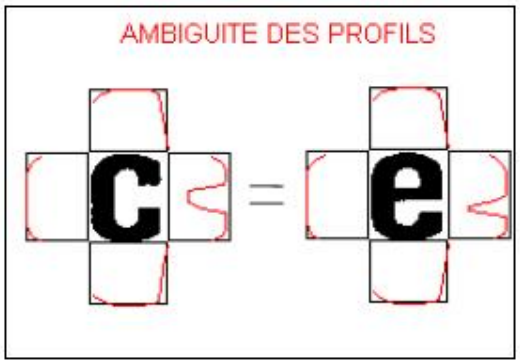


Figure 36 Caractères faiblement discriminés par la seule comparaison des profils externes.

les caractères ("5","S") et ("2","Z") sont mieux reconnus par les profils, beaucoup de substitutions sont obtenues pour le caractère "o" substituable un grand nombre de fois avec les caractères "G","Q","6","8","9". Cela tient du fait que l'information permettant de distinguer ces caractères provient des boucles intérieures fermées et non du contour externe. De même, les nombreuses substitutions entre "h" et "b" proviennent des effets néfastes du renforcement du profil sur les caractères à forts empattements **[Hen 99]**.

2.3.3.4 ***La Méthode des Pixels Moyennés***

Dans cette méthode-là, la matrice de l'image est divisée en *p* x *q* zones de même taille. Pour chaque zone, on calcule la valeur moyenne de niveaux de gris de tous les pixels dans cette zone et le résultat obtenu sera utilisé comme un élément du vecteur de caractéristique. Donc le vecteur de caractéristique est de taille *p* x *q* **[Pha 05]**. Cette méthode est utilisée, afin de minimiser la perte d'information due à la normalisation.

2.3.3.5 ***Le zonage*** Une grille 4*4 est superposée sur l'image du caractère et pour chacune des 4*4 régions résultantes, la moyenne, l'écart type, pourcentage des points noirs est calculé. Nous obtenons un vecteur contenant 4*4*3 attributs.

2.3.3.6 ***La Surface*** : elle consiste à calculer le nombre de points noire dans chaque zone et\ou rapport hauteur largeur.

Les inconvénients des méthodes d'extraction cités précédemment montrent qu'un attribut seul ne peut caractériser de façon non ambiguë un caractère. La combinaison de plusieurs types d'attributs pour représenter un caractère apparaît comme étant la solution incontournable.

2.4 Classification

La classification est l'élaboration d'une règle de décision qui transforme les attributs caractérisant les formes en appartenance à une classe (passage de l'espace de codage vers

l'espace de décision) **[Ben 02]**. Pour construire un classifieur il existe trois approches principales

2.4.1 Approche structurelle

Cette approche est basée sue l'extraction de caractéristiques (ou attributs) qui prennent en considération l'information structurelle de la forme, c à d les attributs qui peuvent décrire la structure de la forme à partir de ses composantes élémentaires.

Cette approche nécessite une mesure de la similarité entre deux représentations structurelles. On distingue plusieurs techniques telles que les structures de graphes et les structures syntaxiques.

2.4.2 Approche statistique

Cette approche est l'étude statistique de mesures effectuées (qui font la description statistique des classes des formes) sur les formes à reconnaître. L'étude de leur répartition dans un espace métrique et la caractérisation statistique des classes permet de prendre une décision du type : plus forte probabilité d'appartenance à une classe **[Gra 02]**.

Elle a besoin d'un nombre élevé d'exemples afin de réaliser un apprentissage correct des lois de probabilité des différentes classes **[Ben 02]**. Les deux principales familles de cette approche sont les méthodes *paramétriques* et les méthodes *non paramétriques*.

Les méthodes paramétriques

Elles opèrent sur l'hypothèse que les clases étudiés suivent une distribution de probabilités de formes bien connu à priori. La prise de décision consiste à déterminer à quelle classe appartient une forme inconnue. L'approche englobe le classificateur euclidien, Mahalanobis, quadratique, gaussien, la règle de Bayes, les chaînes de Markov **[Ben 02]**. Pour plus d'information voir **[Gos 96]**.

Les méthodes non paramétriques

Dans le cas des méthodes non paramétriques, les lois de probabilité sont inconnues pour une des classes. Le problème revient à établir des frontières de décision entre les classes. Les techniques les plus utilisées en reconnaissance de formes sont : la méthode du plus proche voisin, la méthode de Parzen et la méthode d'appariement de graphes. Pour de plus amples informations **[Ben 02]**, **[Liu 05]**. Gosselin **[Gos96]** décrit un ensemble de méthodes statistiques en reconnaissance de formes.

Les réseaux de neurones

C'est déjà vu en détail dans le chapitre précédent. Haykin en propose la définition suivante « *Un réseau de neurones est un processus distribué de manière massivement parallèle, qui a une propension naturelle à mémoriser des connaissances de façon expérimentale et de les rendre disponibles pour utilisation. Il ressemble au cerveau en deux points:*

1. la connaissance est acquise au travers d'un processus d'apprentissage;

2. les poids des connections entre les neurones sont utilisés pour mémoriser la connaissance.»

C'est sur base de cette définition que repose l'élaboration des réseaux de neurones artificiels.

3. Le système mis en place

Comme tout système de reconnaissance, le notre suit la même démarche classique à laquelle nous avons introduit l'étape de sélection d'attributs qui consiste à déterminer les attributs approprié à la reconnaissance de caractères manuscrits isolé et d'éliminer les attributs non pertinents et redondants tout en améliorant les performances du système. Le schéma du système de reconnaissance mis en place est illustré dans la figure ci dessous

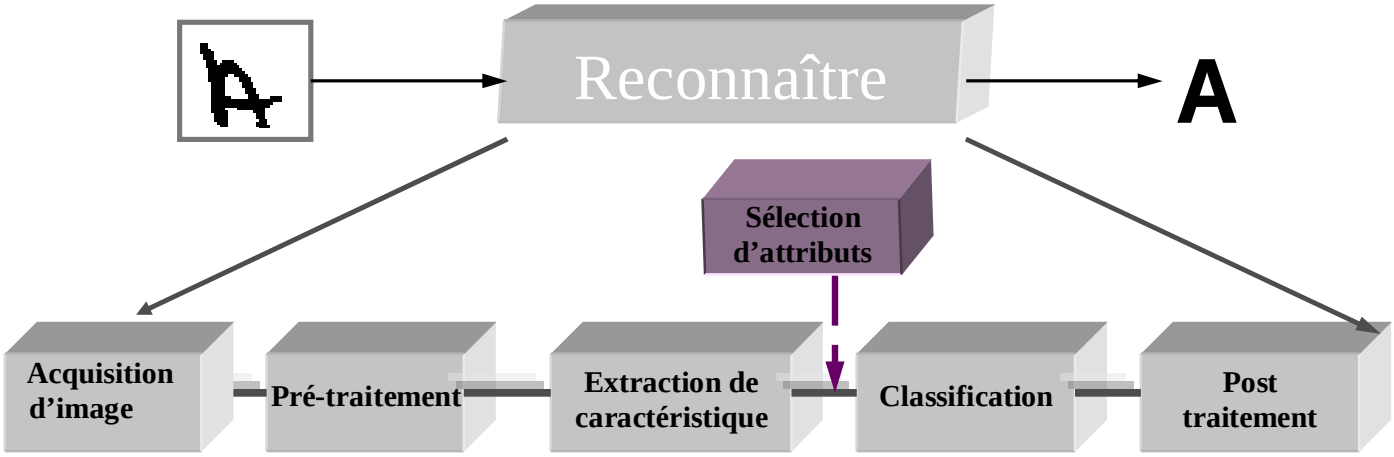


Figure 37 schéma du système mis en place.

3.1 Acquisition d'images

Notre base de données est un échantillon de la base CEDAR qui est une base standard bien connu au sein de la communauté de reconnaissance de formes (voir figure 38). Cette dernière contient 27837 images de caractères alphanumériques manuscrits divisés en deux ensembles, 90 % pour apprentissage et 10% pour test.

Toutes les données ont été digitalisées sur le scanner **Eikonix EC850 CCD.** Ces images ont été extraites à partir d'images d'enveloppes collectées au centre CEDAR à Buffalo (États Unis).

Cette base est un environnement réel caractérisé par

- Différents auteurs.
- Différents style d'écritures.
- Différents outils d'écritures.
- Les auteurs ne savent pas que leurs écritures seront utilisées.

Chaque image de caractère est représentée par 32×32 pixels de niveau de gris allant de 0 à 255 (Figure 37). Il est connu que l'ensemble de test de USPS est plutôt difficile - l'erreur humaine se situe autour de 2.5% [20].

Nous avons accès qu'à 307 caractères mais nous n'avons utilisé que 128 parce que nous nous sommes intéressé qu'aux caractères majuscules.

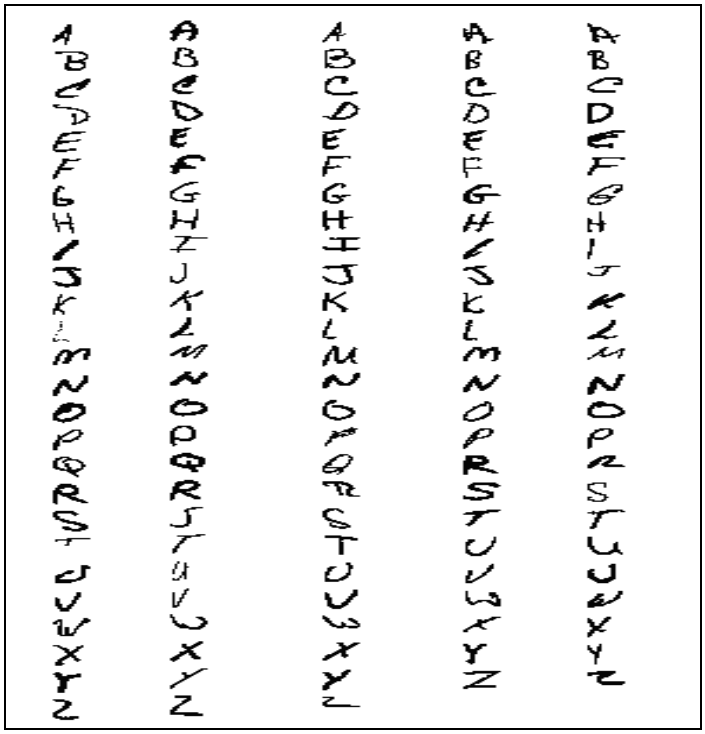


Figure 38 Aperçus sur un échantillon de la base CEDAR

3.2 Les prétraitements effectués

Pour améliorer la qualité des images caractères pour les phases de traitement qui vont suivre. Nous avons effectués les prétraitements suivants :

- Binarisation

Nous avons converti les images numérisées de la base de données (en niveau de gris) en des images binaires.

- Elimination du bruit

Pour éliminer ou tout au moins réduire au maximum le bruit. Nous avons utilisé dans notre système un filtrage bidimensionnel. Où nous avons travaillé avec cinq opérateurs de filtrage différents, pour chaque image de caractère nous appliquons l'opérateur qui lui convient.

- La normalisation

La taille des caractères était différente. Nous avons ramené toutes les images des caractères à la même taille (32×32) pixels. Nous avons utilisé l'algorithme de normalisation décrit auparavant.

3.3 Extraction de caractéristiques

La représentation des images des caractères par des matrices de pixels n'est pas la plus adaptée pour la plupart des systèmes de reconnaissance. Nous avons réalisé une étape d'extraction de caractéristiques de manière à extraire les informations les plus discriminantes pour la tâche de reconnaissance et également pour réduire le volume d'informations qui sera fourni au système. Comme nous l'avons vu auparavant, plusieurs méthodes d'extractions de caractéristiques ont été développées mais aucune d'entre elles, à elle seule, ne peut capter l'information pertinente d'une image donnée de manière idéale. Par conséquent, un amalgame de plusieurs types de caractéristiques nous a semblé nécessaire. Nous avons retenu, parmi ce qui existe, les caractéristiques suivantes :

Le zonage Une grille 4*4 est superposée sur l'image du caractère et pour chacune des 4*4 régions résultantes, la moyenne, l'écart type, pourcentage des points noirs est calculé. Nous obtenons un vecteur contenant 4*4*3 attributs.

La Surface : elle consiste à calculer le nombre de points noire dans chaque zone. Le vecteur est de 16 attributs.

Projection verticale la projection verticale d'une image I représentant un caractère donné est dénoté par :

$$P_y^v = \frac{1}{32}\sum_{i=1}^{32} I_{ij} \quad \textbf{où} \quad P^v = \left[P_1^v \;\; P_2^v, ... , \;\; P_{32}^v\right]$$

Projection horizontale La projection horizontale d'une image I représentant un caractère donné est dénoté par :

$$P_x^h = \frac{1}{32}\sum_{j=1}^{32} I_{ij} \quad \textit{où} \quad P^h = \left[P_1^h \; P_2^h, ... , \; P_{32}^h\right]$$

Nous obtenons 32 valeurs d'attributs pour les projections verticales et 32 pour les projections horizontales.

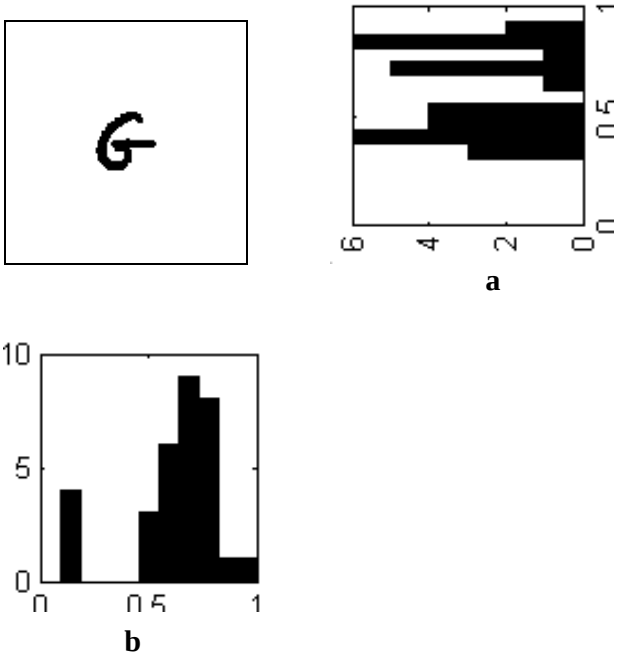


Figure 39 Les histogrammes des projections a. horizontales et b. verticales du caractère G.

Profils

Dans notre cas nous calculons les profils horizontaux de droite et de gauche. Ensuite nous partitionnons les images profils en 4*4 zones, et calculons la moyenne, écart type pourcentage des points noire sur chaque zone. Cela procure un vecteur de 96 attributs.

Enfin nous obtenons un vecteur de 224 attributs pour chaque caractère.

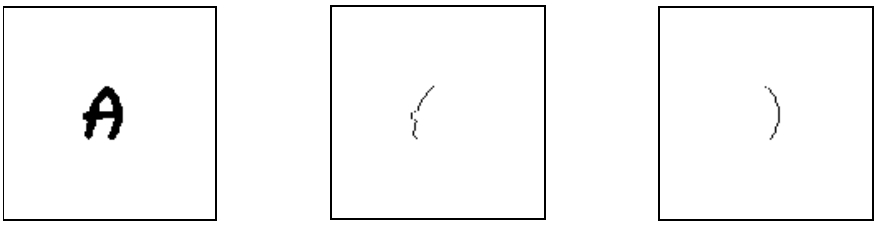


Figure 40 Les profils a. Gauche et b. droit du caractère A.

3.4 La sélection des attributs

Parmi l'ensemble des attributs effectués sur la base des images des caractères (issue de l'étape précédente), touts ne sont pas aussi pertinents. Il est possible que certains Correspondent à du

bruit ou qu'ils soient peu informants, corrélés ou même inutiles au système pour l'accomplissement de sa tâche. Pour surmonter ce problème nous avons introduit une étape de sélection des attributs après l'étape de l'extraction de caractéristiques, qui permet de sélectionner le sous ensemble des attributs nécessaire et suffisant rendant le système de reconnaissance fiable et robuste. Pour cela nous avons proposé deux nouvelles méthodes de sélection d'attributs améliorant la représentation de l'information dont dispose le système de reconnaissance (augmentant le pouvoir discriminant des caractéristiques). Les approches proposées suggère l'utilisation des algorithmes génétiques (SSGA, GGA) appropriés pour l'exploration de l'espace des sous ensembles d'attributs et d'une approche enveloppante pour l'évaluation des individus basée sur un réseau de neurones multicouches comme classifieur. Ce dernier est entraîné par les sous ensembles d'attributs représentés par les individus générés par l'algorithme génétique. Au niveau de l'algorithme génétique, un individu permet d'encoder un sous ensemble d'attributs en spécifiant la présence ou l'absence d'un attribut, l'approche est décrite en détail dans la partie suivante.

3.5 Classification

Lors de la classification, nous avons utilisé un réseau de neurones à rétropropagation de gradients d'erreurs à une couche cachée. Les 224 attributs développés précédemment sont les entrées du réseau de neurones. La couche cachée est composée de 125 neurones. Les classes à discriminer sont les 26 caractères latin, d'où le choix de 26 neurones pour la couche de sortie. La fonction d'activation des neurones est la sigmoïde. L'initialisation des poids et biais du réseau est aléatoire. Un échantillon de la base de caractères CEDAR est utilisé pour l'apprentissage du réseau de neurones. Il comporte un ensemble 26 caractères. Cette étape est bien expliquée en détail dans la prochaine section.

3.6 Décision

Si le neurone de sortie O_i a la valeur maximale au niveau j, alors le neurone d'entrée E_i (qui représente le vecteur d'attributs d'un caractère donné) appartient à la classe j.

4. Conclusion

Dans cette partie, nous avons présenté les différentes étapes d'un système de reconnaissance de caractères et les différentes alternatives pour chacune. Ensuite nous avons décrit le système mis en place. Dans la prochaine section, nous allons abordée l'étape de sélection d'attributs en détail parce qu'elle constitue la problématique de ce travail.

1. Introduction

Notre objectif est de proposer une nouvelle méthode de sélection d'attributs qui permet la réduction de l'ensemble d'attributs initiale en gardant que les attributs pertinent et non redondant, pour la mise en œuvre de celui-ci nous avons choisi la reconnaissance hors ligne de caractères manuscrits comme domaine d'application. Les deux méthodes proposées suggèrent l'utilisation de la technique d'optimisation que sont les algorithmes génétiques pour l'exploration de l'espace des sous ensembles, et le réseaux de neurones pour leur évaluation. Dans cette partie, nous expliquons en détail les approches proposées.

2. Les méthodes de sélection d'attributs mises en place

Comme nous avons vu précédemment, que la sélection du sous ensemble d'attributs optimal fait partie des problèmes NP-difficile. Pour résoudre ce problème, plusieurs algorithmes heuristiques ont été proposés dans la littérature. Dans ce travail nous avons proposé deux nouvelles méthodes basées sur un algorithme génétique neurale et qui suivent la même démarche citée précédemment (voir le tableau suivant).

	La 1^{ere} méthode	***La 2eme méthode***
Points de départ	Des sous ensembles d'attributs choisi aléatoirement (population initiale)	
Stratégie de recherche	Algorithme génétique stationnaire	Algorithme génétique générationnel
Stratégie d'évaluation	méthode enveloppante (Wrapper) où le classificateur est un perceptron multicouches	
Critère d'arrêt	nombre de génération.	
Validation des résultats	surveillance du changement des performances par rapport les changements des sous ensembles d'attributs calculés sur une base de caractères latins majuscule.	

Tableau 4 Les différentes étapes des méthodes de sélection proposées

2.1 Point de départ

Contrairement à la majorité des techniques de recherche qui démarrent en choisissant un point de l'espace de recherche et poursuivent en opérant des déplacements locaux à partir de ce point, un AG maintient une population de points de l'espace.

Comment l'initialisation de la population est faite ?

Comme dans tout problème d'optimisation, une connaissance de bons points de départ conditionne la rapidité de la convergence vers l'optimum. Et comme la position de l'optimum dans notre espace des sous ensembles est totalement inconnue, il est naturel de générer aléatoirement des individus pour avoir une population hétérogène que possible, afin d'explorer largement l'espace.

2.2 Les caractéristiques des algorithmes génétiques utilisés

Dans notre papier nous avons opté pour deux variantes des algorithmes génétiques (SSGA, GGA) comme stratégie de recherche de nos méthodes de sélection des attributs proposés, Les caractéristiques de nos algorithmes génétiques utilisés sont illustré dans le tableau suivant (Supposant que N est la taille de l'ensemble complet des attributs) :

CARACTÉRISTIQES	APPLICATION
Espace de recherche	(2^N-1) Sous ensembles d'attributs
Nature de la population	Ensemble des sous ensemble d'attributs
Taille de la population	Fixée par expérience
Nature de l'individu	Un sous ensemble d'attributs
Codage du chromosome	Binaire
Taille du chromosome	Fixe (N)
Fonction d'évaluation	Réseaux de neurones
Types d'hybridation	La mutation aléatoire et le croisement (slicing crossover)
Type de sélection	Sélection par tournoi
Nombre de génération maximale	Fixée par expérience
Fitness	Moyenne des taux d'erreurs des bases (Apprentissage, test1, test2)

Tableau 5 Caractéristiques des algorithmes génétiques utilisés

2.2.1 Codage du chromosome *:* L'individu dans notre système représente un sous ensemble d'attributs, il est codé sous forme de chaîne de bits. Le i^{eme} bit est à 1 ça indique la présence du i^{eme} attribut, il est à 0 ça indique l'absence du i^{eme} attribut. La taille de l'individu est égal au nombre total des attributs utilisés (224).

2.2.2 Sélection

Pour identifier les meilleurs individus de la population et d'éliminer partiellement les mauvais, nous avons utilisé la ***sélection par tournoi.*** Elle n'utilise que des comparaisons entre

individus. Elle possède un paramètre *T*, taille du tournoi. Pour sélectionner un individu, on en tire *T* uniformément dans la population, et on sélectionne les P meilleurs de ces *T* individus.

Le choix de T et P

Le choix de *T* permet de faire varier la *pression sélective*, c'est-à-dire les chances de sélection des plus performants par rapport aux plus faibles.

Dans le cadre de notre travail, nous tirons 2 paires d'individus (T = 4), et sélectionnons le meilleur individu de chaque paire (P = 2) en terme de sa fonction d'adaptation.

Pourquoi T est assez petit ?

Pour augmenter les chances que les individus de piètre qualité participent à l'amélioration de la population. Puisque un bon sous ensemble d'attribut peut être issu d'un croisement de deux mauvais sous ensembles ou d'un bon avec un mauvais.

Pourquoi la sélection par tournoi ?

- Ce choix se justifie, par les expériences que nous avons effectuées. Nous avons appliqué trois méthodes de sélection différentes (la sélection Roulette wheel, sélection par rang, sélection par tournoi). C'est bien qu'avec la sélection par tournoi que notre système donne les meilleurs résultats.
- Ce choix peut aussi se justifie théoriquement

  Limites de la sélection Roulette wheel

  - Elle privilège les individus ayant une forte fitness au détriment des individus moins forts (un Pb pour les sous ensembles d'attributs).
  - Un même individu pourra avec cette méthode être sélectionné plusieurs fois.
  - le cas idéal d'application de cette méthode est bien évidemment celui où la population est de taille très importante.
  - Lorsque la valeur d'adaptation des chromosomes varie énormément, on arriverait à une stagnation de l'évolution.

  Limites de la sélection par rang

  - Avec cette méthode, certes tous les chromosomes ont une chance d'être sélectionnés. Mais 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.

2.2.3 Croisement

Pour enrichir la diversité de la population, nous avons utilisé le croisement à découpages, ou *slicing crossover*. Pour effectuer ce type de croisement, on tire aléatoirement une position inter-gènes dans chacun des parents (les parents sont les individus sélectionnés). On échange ensuite les deux sous-chaînes de chacun des chromosomes, ce qui produit deux enfants.

Choix du croisement

Le choix de ce type croisement se justifie, par les expériences que nous avons effectuées. Nous avons appliqué trois méthodes de croisement différentes (le croisement à découpages, le croisement en deus points, le croisement uniforme), les résultats fournis par le système par rapport ces trois opérateurs se ressemble (une différence négligeable). Nous avons opté pour la configuration minimale (le croisement à découpages).

2.2.4 Mutation

Pour que l'algorithme génétique puisse atteindre tous les points de l'espace, nous avons utilisé un opérateur de mutation. Nous avons opté pour la mutation aléatoire qui consiste généralement à tirer aléatoirement un gène dans le chromosome et à remplacer ce dernier en appliquant un *non* logique à la valeur de l'allèle correspondant.

2.3 Critère d'évaluation

Les deux approches proposées utilisent une approche enveloppante pour l'évaluation des sous ensembles d'attributs, elle exige l'utilisation d'un classificateur. Nous avons opté pour le réseau de neurone et plus précisément le perceptron multicouches.

Ce système est composé d'un ensemble structuré de *neurones* organisés en couches (notre perceptron a une couche d'entrée, une couche cachée, une couche de sortie), ces neurones fonctionnent en parallèle et sont fortement interconnectées (Chaque neurone n'est relié qu'aux neurones de la couche précédente et de la couche suivante, excepté pour les couches d'entrée et de sortie et il n'y a pas de connexions entre les cellules d'une même couche). Chaque neurone évalue son activation grâce à la fonction sigmoïde décrite auparavant. Nous avons opté pour la méthode de rétropropagation de gradient de l'erreur comme algorithme d'apprentissage de notre perceptron.

Pourquoi le réseau de neurones

- C'est dans le domaine de la reconnaissance en particulier que les réseaux neuronaux ont montré leurs efficacités. ils ont à l'heure actuelle étés capables de faire aussi bien

que les programmes classiques dans ce domaine et pour certains problèmes spécifiques peuvent aborder des reconnaissances de formes pour lesquelles les approches classiques n'ont pas de solution.

- la flexibilité des réseaux neuronaux face aux variations de l'environnement à analyser (ils s'adaptent à la nouveauté et peuvent même travailler sur des signaux extrêmement dégradés.
- De plus, cette robustesse s'applique aussi à la dégradation interne du fonctionnement du réseau neuronal. Il peut encore donner des résultats valables quoique avec une certaine perte de précision.

Pourquoi le perceptron multicouches précisément

Nous avons opté pour le Perceptron multicouches comme classificateur, parce que nous nous sommes basé sur les travaux de Bernard Gosselin. Il a appliqué dans sa thèse de doctorat, au problème de reconnaissance de caractères manuscrits divers classificateurs afin de voir leurs performances d'une manière objective. Il a testé plusieurs classificateurs classiques et différents modèles de réseau de neurones parmi lesquelles se trouve le perceptron multicouches. Le meilleur résultat a été atteint à l'aide du perceptron multicouches (où la couche cachée ne contient que 15 neurones). Cet excellent résultat obtenu met en évidence l'intérêt du perceptron multicouches vis à vis les autres modèles.

Pourquoi une seule couche cachée

En se basant sur le théoreme de Cybenko , il a démontré qu'un perceptron multicouches comportant une seule couche cachée composée de neurones à fonction d'activation continue non linéaire, est suffisant pour approximer, au sens des moindres carrés, avec une erreur arbitrairement faible pour un ensemble donné d'objets d'apprentissage, n'importe quelle transformation continue représentée par un ensemble de vecteurs d'entrées et un ensemble de vecteurs de sorties désirées.

Théorèmes

Cybenko 89

Pour toute fonction F continue définie et bornée sur un ensemble borné, et pour tous ε, il existe un *réseau à une seule couche cachée* de neurones sigmoïdes qui approxime F à ε prés **[Mou 05]**.

Sussman 92

Les réseaux à une couche cachée forment une famille d'approximateurs parcimonieux : à nombre égal de paramètres on approxime correctement plus de fonctions q'avec les polynômes **[Mou 05]**.

2.3.1 L'algorithme d'apprentissage utilisé

Au moyen d'exemples. L'apprentissage consiste à présenter au réseau une série d'entrées, et à modifier ses connexions (les poids et biais) pour qu'à chacune de ces entrées corresponde la sortie attendue: c'est ici que nous avons utilisé le fameux algorithme de rétropropagation du gradient de l'erreur.

L'algorithme consiste dans un premier temps à propager vers l'avant les entrées jusqu'à obtenir une sortie calculée par le réseau. La seconde étape compare la sortie calculée à la sortie réelle connue. On modifie alors les poids de telle sorte qu'à la prochaine itération, l'erreur commise entre la sortie calculée et connue soit minimisé **[Sen 05]**. Comme on a une couche intermédiaire (cachée), on rétro propage l'erreur commise vers l'arrière jusqu'à la couche d'entrée tout en modifiant la pondération. On répète ce processus sur tous les exemples jusqu'à ce que le critère d'arrêt soit satisfait. Le critère d'arrêt que nous avons utilisé dans le cadre de ce travail est soit le nombre de cycles maximale est atteint soit l'erreur de sortie est considérée comme négligeable (en fixant un seuil pour celui-ci).

Notre réseau est composé d'une couche d'entrée qui représente les vecteurs d'attributs des exemples des caractères, une seule couche cachée, une couche de sortie qui représente les 26 lettre latines en majuscules (représenté par 26 neurones).

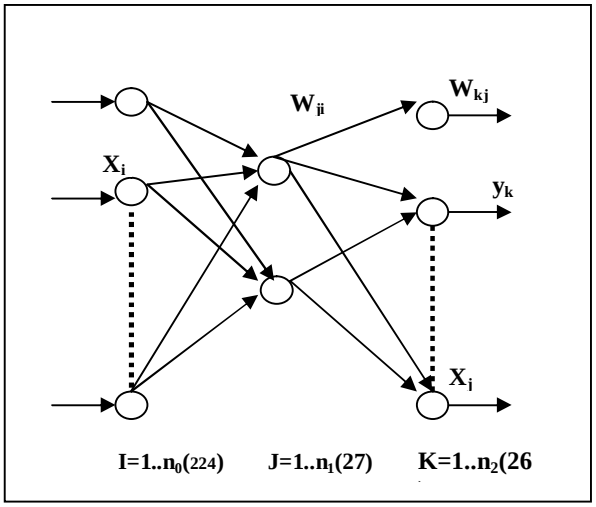


Figure 41 schéma représentatif du perceptron utilisé.

2.3.2 Fonctionnement de l'algorithme de rétropropagation

On calcule pour chaque neurone de la couche cachée la somme pondérée des valeurs de sortie des neurones de la couche d'entrée.

$$\textbf{Pour j= 1..n}_1 \textbf{ :}\ a_j^{(1)} = \sum_{i=1}^{n_0} w_{ij} xi$$

On calcule la valeur de sortie de chaque neurone de la couche cachée en bridant la somme pondérée correspondante :

$$x_j^{(1)} = F(a_j^{(1)})$$

On calcule pour chaque neurone de la couche de sortie la somme pondérée des valeurs de sortie des neurones de la couche cachée

$$a_k^{(2)} = \sum_{j=1}^{n1} w_{ij} x_j^{(1)}$$

On calcule la valeur de sortie de chaque neurone de la couche de sortie en bridant la somme pondérée correspondante :

$$y_k = F(a_k^{(2)})$$

Calcul de l'erreur

Fonction de coût

x=[x_1…x] (avec y^{des} sortie désirée).

On calcule la sortie correspondante y= [y_1…y_{n2}]

Erreur : $e_k = y^{des} - y_k$

Fonction coût associé à l'exemple : $J_{exemple} = \frac{1}{2}\sum_{k=1}^{n2} e_k^2$ (erreur quadratique)

Coût global : $J = \sum_{l=1}^{n} J_{(exemplel)}$ **H**

Calcul du gradient : Le but de cette méthode est de minimiser la fonction coût par rapport à chaque poids, ce qui revient à calculer la dérivée du coût globale par rapport à chaque poids : $\frac{\partial J}{\partial W}$

Mise à jour de wW_{ji} et W_{kj} selon la règle delta : $\Delta W = h\frac{\partial J}{\partial W}$ où η est une constante réelle positive (constante d'apprentissage ou coefficient d'apprentissage).

H n est le nombre d'exemples.

Calcul de $\frac{\partial J}{\partial W_{ji}}$ et $\frac{\partial J}{\partial W_{kj}}$:

« Couche de sortie

Calcul de : où η est une constante réelle positive (constante d'apprentissage ou coefficient d'apprentissage).

Calcul de : $\frac{\partial J}{\partial W_{kj}} = \frac{\partial J}{\partial y_k} \frac{\partial y_k}{\partial a_k^{(2)}} \frac{\partial a_k^{(2)}}{\partial W_{kj}}$

On a $\frac{\partial J}{\partial y_k} = -(y_k^{des} - y_k)$ car $J = \frac{1}{2}\sum (y_k^{des} - y_k)^2$

$\frac{\partial y_k}{\partial a_k^{(2)}} = F'(a_k^{(2)})$ car $y_k = F'(a_k^{(2)})$

$\frac{\partial a_k^{(2)}}{\partial W_{kj}} = x_j^{(1)}$ car $a_k^{(2)} = \sum W_{kj} x_j^{(1)}$

Posons $Err_k = \frac{\partial J}{\partial a_k^{(2)}} = -(y_k^{des} - y_k)F'(a_k^{(2)})$ Donc $\frac{\partial J}{\partial W_{kj}} = Err_k \; x_j^{(1)}$

« Couche cachée

Le changement des poids W_{ji} , change $x_j^{(1)}$ et ce dernier change chaque entrée des cellules de la couche de sortie. Le changement de J avec un changement de W_{ji} est la somme des changements de chaque cellule de sortie.

Calcul de :

$$\frac{\partial J}{\partial Wji} = \sum_{k=1}^{n2} \frac{\partial J}{\partial y_k} \frac{\partial y_k}{\partial a_k^{(2)}} \frac{\partial a_k^{(2)}}{\partial x_j^{(1)}} \frac{\partial x_j^{(1)}}{\partial a_j^{(1)}} \frac{\partial a_j^{(1)}}{\partial w_{ji}}$$

$$\frac{\partial J}{\partial W_{ji}} = \sum -(y_k^{des} \_ y_k) F'(a_k^{(2)}) W_{kj} \; F'(a_j^{(1)}) x_i$$

$$= \sum_{k=1}^{n2} Err_k \; w_{kj} \; F'(a_j^{(1)}) x_i$$

Posons $Err_j = \sum Err_k W_{kj} F'(a_j^{(1)})$ donc : $\frac{\partial J}{\partial Wji} = Err_j \; x_i$

Changement des poids

- *Couche de sortie :*

$$W_{kj(nouveau)} = W_{kj} + \Delta W_{kj} \Rightarrow W_{kj(nouveau)} = W_{kj} - h\frac{\partial J}{\partial W_{kj}} \Rightarrow \boxed{W_{kj(nouveau)} = W_{kj} - h\, Err_k\, x_j^{(1)}}$$

a *Couche cachée :*

$$W_{ji(nouveau)} = W_{ji} + \Delta W_{ji} \Rightarrow W_{ji(nouveau)} = W_{ji} - h\frac{\partial J}{\partial W_{ji}} \Rightarrow \boxed{W_{ji(nouveau)} = W_{ji} - hErr_j\, x_i}$$

Ce processus est répété autant de fois tant que le critère d'arrêt n'est pas satisfait **[Sen 05]**.

Pourquoi l'algorithme de rétropropagation

Après avoir présenté les détails des étapes communes aux deux méthodes de sélection d'attributs proposés. Il est temps de les présenter en décrivant leurs principes de fonctionnement.

Pourquoi une méthode enveloppante **?**

Nous avons opté pour les méthodes enveloppantes parce qu'elles réalisent les meilleurs taux de réussite par rapport les filtrantes, et ont tendance de choisir des sous ensembles de taille moins importantes que les filtrantes. Elles sont certainement très lentes mais avec le classificateur Perceptron où l'algorithme de rétropropagation avec momentum accélère l'apprentissage.

3. Première approche basée Algorithme génétiques stationnaire (SSGA)

La figure ci-dessous présente un aperçu général de cette approche (Elle explique très bien les étapes de l'algorithme SSGA proposé)

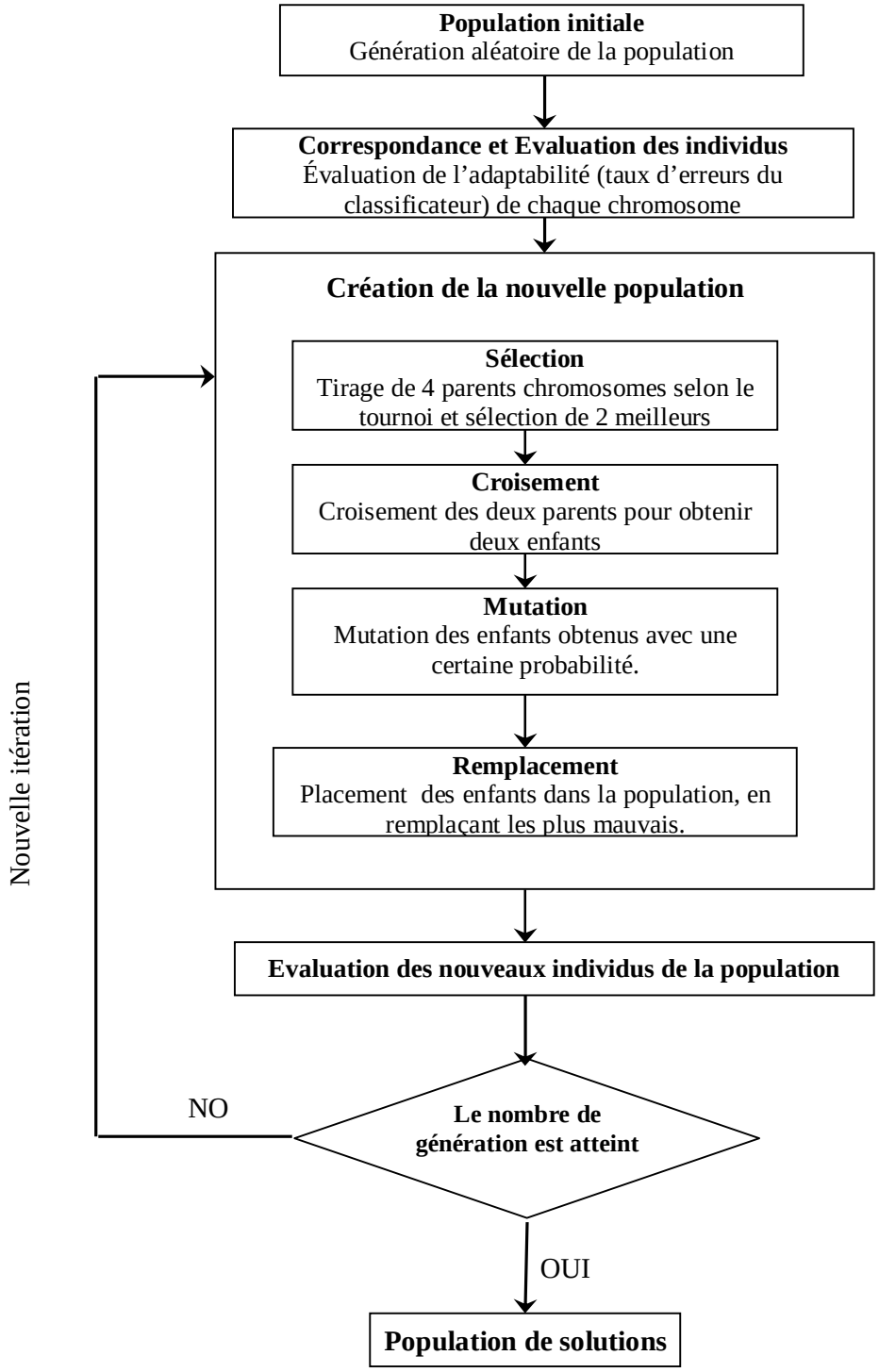


Figure 42 Sélection d'attributs basée SSGA.

En partant d'un ensemble de solutions potentielles* générées aléatoirement, Nous évaluons la fonction d'adaptation de chaque individu sur les bases d'apprentissage et tests, et ceci ça se passe en présentant au réseau de neurones les vecteurs d'attributs correspondant aux individus de la population calculés sur la base d'apprentissage et les bases de test, et nous aurons en retour le taux d'erreurs de reconnaissance pour chaque individu. Ensuite nous sélectionnons deux individus parents selon leurs fonctions d'adaptation par le tournoi, ces derniers vont être croisé avec une probabilité **Pc** pour produire deux enfants, après nous mutons ces enfants avec une certaine probabilité **Pm,** et nous les plaçons dans la population pour avoir une nouvelle qui va participer à la prochaine génération. Si le critère d'arrêt est satisfait ″ c'est bien le nombre de génération dans notre cas″ on aura comme solution une population de sous ensembles d'attributs, de cette dernière on en choisira un seul qui est meilleur. Sinon les étapes de sélection, croisement, mutation, remplacement s'itère jusqu'à la satisfaction du critère d'arrêt.

4. Deuxième approche basée Algorithme génétique Générationnel

La figure ci-dessous présente un aperçu général de cette approche

* Ensemble de sous ensembles d'attributs.

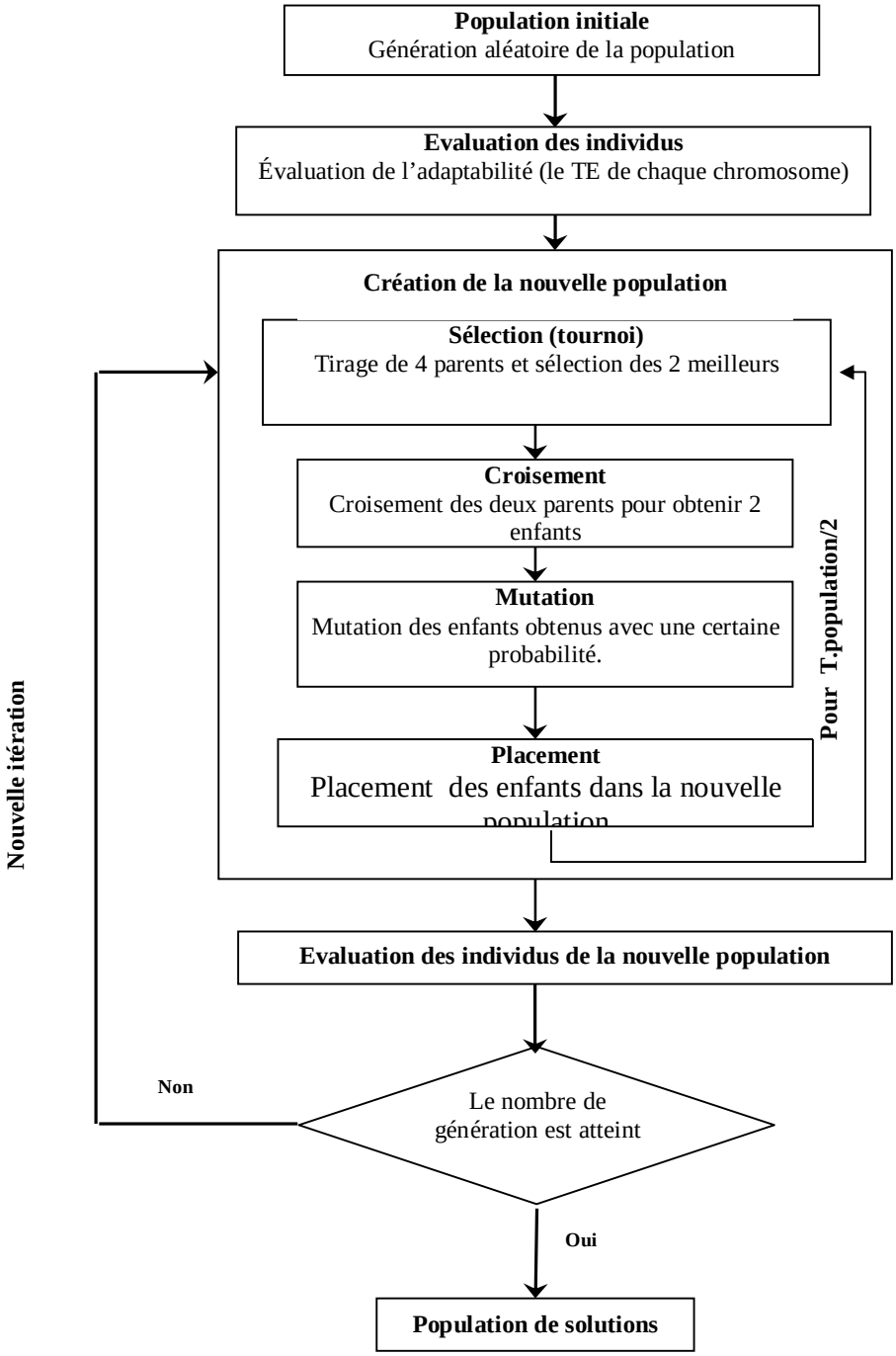


Figure 43 Sélection d'attributs basée GGA.

Le même principe que l'algorithme précédent, sauf que la sélection tournoi est exécuté (T.population/ 2)[†] fois pour chaque génération, et les enfants reproduits seront placés dans une

[†] Si t.population est paire alors le nombre d'exécution de tournoi est t.population/2
Sinon (tournoi est t.population/2) + 1 tournoi avec 2 participants

nouvelle population et non pas remplacé les mauvais individus dans l'ancienne comme le cas de la précédente, çà d à chaque génération c'est toute une population qui va être reproduite.

5. Aperçus générale du système de reconnaissance (avec la sélection d'attributs)

Après l'introduction de l'étape de sélection d'attributs nous pouvons schématiser le système globale comme suit

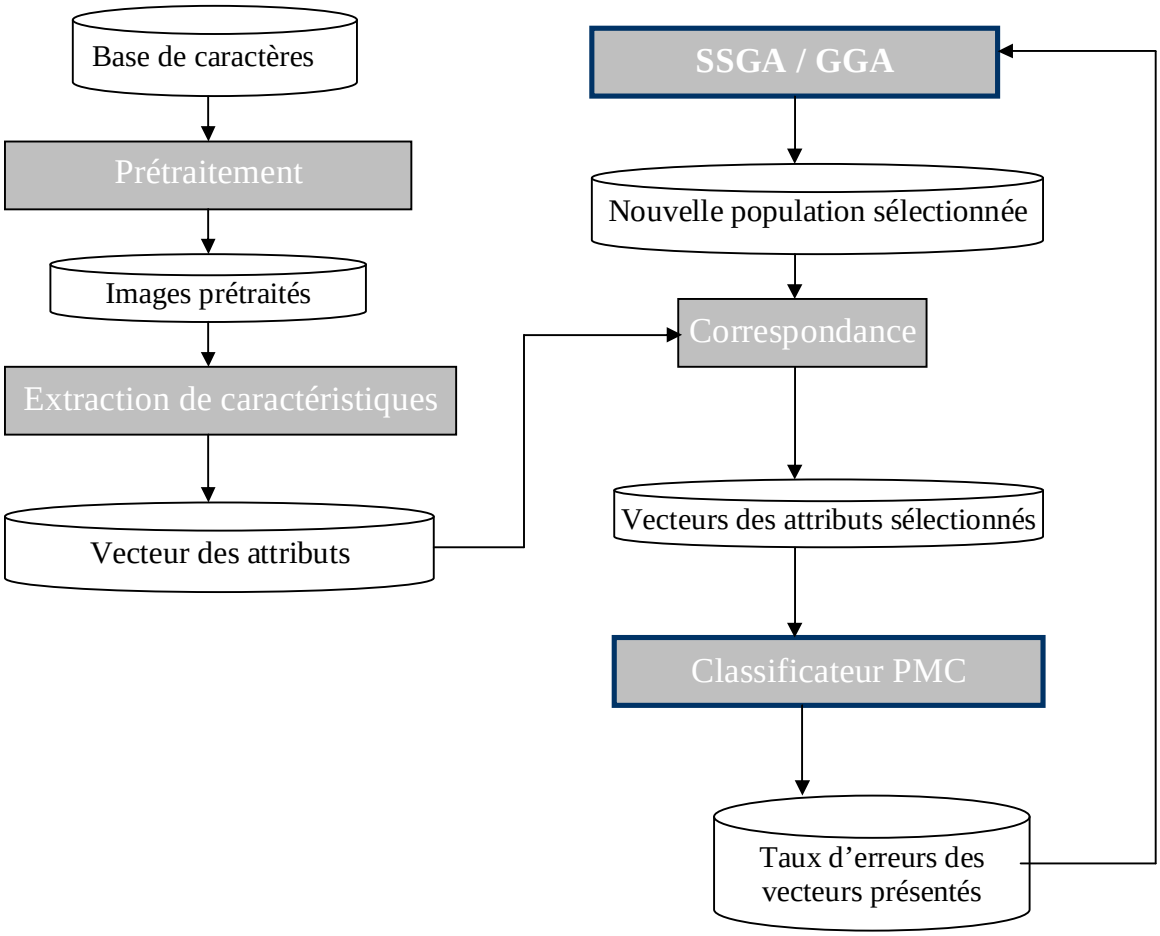


Figure 44 Aperçu générale du système de reconnaissance mis en place.

Ce schéma explique Le fonctionnement du système avec les approches de sélection d'attributs proposées.

6. Conclusion

Dans cette partie nous avons décrit les deux méthodes proposées en détails, ensuite nous avons donné une vue générale sur le système avec les deux méthodes proposés. Et nous pouvons dire Que c'est toute la combinaison (Méthodes d'extraction de caractéristiques,

L'Application de reconnaissance, Stratégie de recherche ″Algorithme génétique″, Stratégie d'évaluation ″ PMC″) qui est le point clé de cette innovation.

Comme nous avons utilisé des algorithmes génétiques dans ce travail, nous se retrouvons évidemment avec leur problème critique qui est l'adaptation des paramètres, et il n'y a que les expériences qui peuvent surmonter ce problème. Pour cela nous sacrifions la prochaine section pour les expériences effectuées, dans le but de régler les paramètres des algorithmes génétiques d'une part et d'une autre, bien valider les résultats obtenus par les deux méthodes proposées.

1. Introduction

Dans cette partie, nous présentons les expériences effectuées dans le cadre de ce travail.

2. Logiciel utilisé

Nous nous sommes tournée ver l'outil Matlab.

Matlab est l'abréviation de matrix laboratory, où l'élément de base de donnée est une matrice. Avec ses fonctions spécialisées, MatLab peut être aussi considéré comme un langage de programmation adapté pour les problèmes scientifiques. C'est un interpréteur : les instructions sont interprétées et exécutées ligne par ligne. Il fonctionne dans plusieurs environnements

Matlab est organisé en boîte à outils (toolbox) spécialisés [http://www.mathworks. com /products/neuralnet/description6.html]. Les toolboxes sont réellement des boîtes à outils comportant une collection de fonctions relatives à plusieurs domaines scientifiques et techniques. Nous trouvons :

- *Image Processing Toolbox* propose un ensemble complet d'algorithmes et d'outils pour le traitement, l'analyse, la visualisation et le développement d'algorithmes de traitement d'images. MATLAB prend en charge des formats de données et d'images, notamment JPEG, TIFF, PNG, HDF, HDF-EOS, FITS, Microsoft Excel, ASCII et les fichiers binaires. Il prend également en charge les formats d'images multi-bandes, tels que LANDSAT. Image Processing Toolbox fournit un environnement intégré pour un affichage et une bonne exploration des images.
- Une boîte à outil spéciale qui fournit des outils pour la conception, la mise en œuvre, la visualisation et la simulation de réseaux de neurones. *Neural Network Toolbox* offre une prise en charge complète de nombreuses structures réseau éprouvées, ainsi que des interfaces utilisateur graphiques vous permettant de concevoir et de gérer vos réseaux. La conception modulaire, ouverte et extensible de cette boîte à outils simplifie la création de fonctions et de réseaux personnalisés.

C'est pour ces deux raisons là que Matlab nous a semblé le langage le mieux adapté pour notre travail.

3. Le matériel utilisé

Le matériel utilisé est un ordinateur avec processeur Pentium M à 1.73 GHz avec 512 mégaoctets de mémoire, 2 mégaoctets de cache, sous le système d'exploitation Windows XP.

Résultats expérimentaux

Pour démontrer les performances de nos algorithmes proposés, nous les avons appliqué sur des échantillons de caractères différents (Une base d'apprentissage, base test1, base test2). En plus, nous avons comparé le taux d'erreur du classificateur sur l'ensemble complet des attributs avec le taux d'erreur du classificateur sur le sous ensemble sélectionné.

Tout d'abord nous avons partitionné notre base en trois :

Base d'apprentissage : l'entraînement est effectué sur un ensemble des caractères (A ….. Z), contenant deux exemples pour chaque classe, donc nous avons 52 exemples pour la phase d'apprentissage.

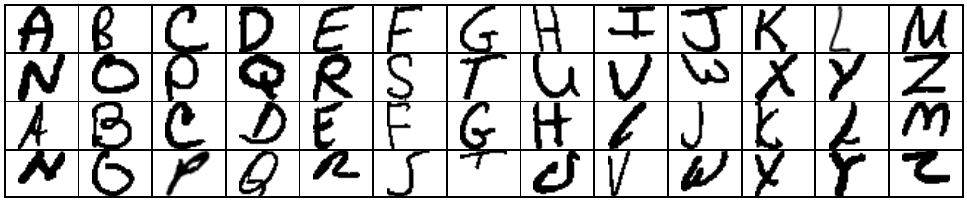


Figure45 Les exemples de la base d'apprentissage.

- Base de test1 : elle est différente de la précédente, contient 26 exemples, un pour chaque classe.

Figure46 des exemples de la base test1.

- Base de test2 : elle est aussi différente des 2 précédentes, elle contient 26 exemples.

Figure 47 des exemples de la base test2.

Le réglage des paramètres de notre système

Le point critique de notre système est bien le réglage des paramètres que se soit pour l'algorithme génétique ou pour le réseau de neurones. Et comme il n'existe pas de règles universelles de réglages (les outils théoriques disponibles pour effectuer les choix optimaux sont quasi-inexistants), donc seuls les résultats expérimentaux peuvent donner une idée du comportement de notre système (algorithme génétique + réseau de neurones) par rapport ces paramètres. Pour cela il nous a semblé nécessaire d'expérimenter de nombreuses combinaisons pour obtenir de bons résultats.

Les paramètres modifiables de notre système sont les suivants :

1. Nombre de générations.

2. Taille de la population.
3. Probabilité de croisement.
4. Probabilité de mutation.
5. la sélection (pression de sélective).
6. Opérateurs de croisement et mutation.
7. la population initiale
8. Nombre de cycles d'apprentissage du réseau de neurones.
9. Seuil de gradient.

1. ²SSGA²

Critère d'évaluation : ²Erreur de classification²,

Nous avons effectué plusieurs expériences sur les trois bases en fixant les autres paramètres et jouant sur les valeurs de chacun de ″nombre de générations, nombre de cycles, taille de population″. Les résultats de ces expériences sont présentés dans les tableaux en dessous, et pour bien déterminer les valeurs optimales des paramètres par rapport les trois bases en même temps, nous construisons les courbes à partir des résultats des tableaux (voir figure).

Ë Nombre de cycles (Nombre de génération=2 ; taille population=10).

Nombre de cycles	100	200	500	800	1000	2000	3000	5000
Erreur apprentissage	26 .5016	0.5460	9 .8047	3.6900 e^{-18}	0.9698 e^{-10}	7.7500 e^{-31}	1.411 e^{-30}	3.2372 e^{-30}
Erreur de la b.test1	26.4878	24.7478	30.1138	0.0322	1.3746 e^{-10}	7.6700 $e^{-3\ 1}$	1.1231 e^{-28}	0.0013 e^{-20}
Erreur de la b.test2	26.4876	24.7143	32.5409 7.75	0.8283	5.0641 e^{-6}	1.5794 e^{-30}	4.6393 e^{-30}	0.9897 e^{-29}

Tableau 6 l'erreur de classification par rapport le nombres de cycles.

Ë *Nombre de générations (taille population=10 ; nombre de cycle =2000).*

Nombre de générations	2	6	10	20
Erreur apprentissage	7.75 e^{-31}	8.1085 e^{-25}	0.9795 e^{-20}	3.44 e^{-31}
Erreur de la b.test1	2.09 e^{-29}	0.0078 e^{-20}	0.9796 e^{-20}	2.4 e^{-32}
Erreur de la b.test2	0.9921 e^{-20}	1.0568 e^{-20}	3.1109 e^{-20}	2.1047 e^{-30}

Tableau 7 l'erreur de classification par rapport le nombre de générations.

Ë *Taille de population (Nombre de génération=2 ; nombre de cycle =2000).*

taille population	10	20	30	40	50
Erreur apprentissage	7.7500 e^{-31}	1.8043 e^{-30}	3 .2256 e^{-32}	6.0335 e^{-32}	4.1454 e^{-32}
Erreur de la b.test1	7.6700 e^{-31}	1.3151 e^{-28}	3.0150 e^{-26}	1.5653 e^{-33}	2.6047 e^{-29}
Erreur de la b.test2	1.5794 e^{-30}	2.9342 e^{-28}	81.7916 e^{-26}	4.3540 e^{-33}	2.8572 e^{-29}

Tableau 3 l'erreur de classification par rapport la taille population.

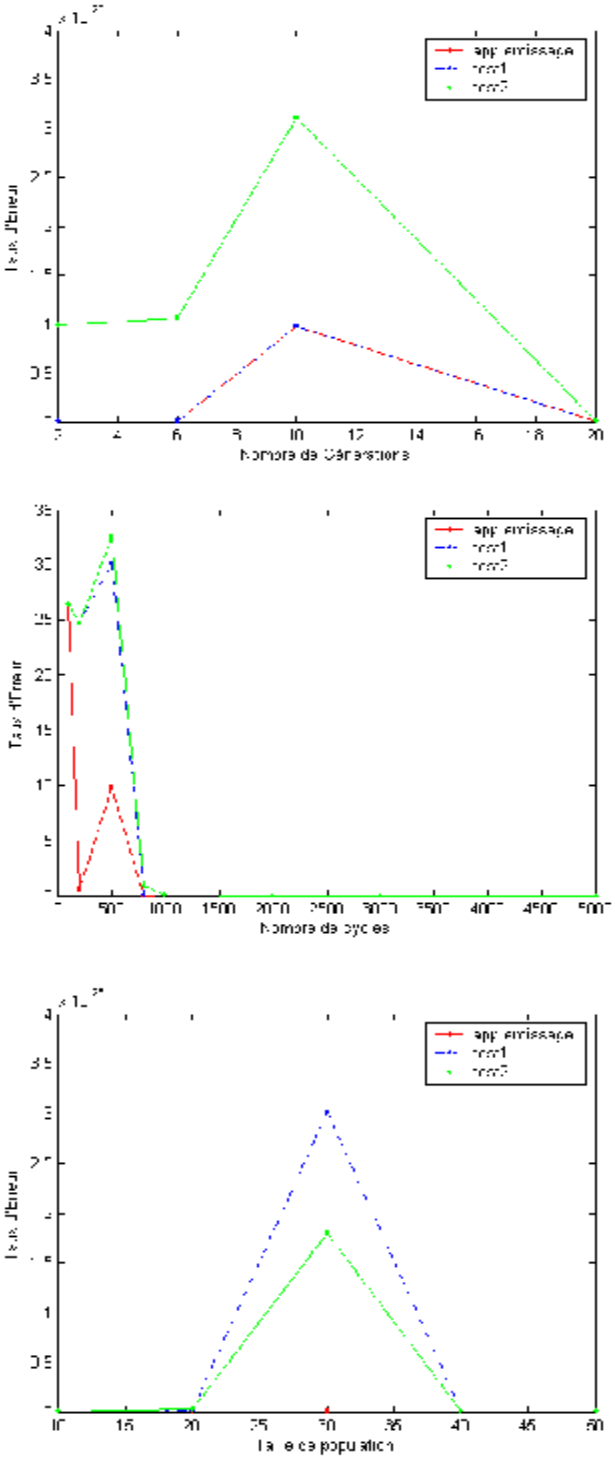


Figure 48, 49, 50 les erreurs de classification par rapport nombre de génération, nombres de cycles, taille population

En observant les courbes, Nous trouvons que les meilleurs résultats sont obtenus pour les valeurs des paramètres suivantes :

Nombre de cycles=2000, taille de population=10, nombre de génération=20.

En fin, nous exploitons ces valeurs obtenues pour le système de reconnaissance avec et sans sélection d'attributs, et nous comparons ensuite ses résultats dans les deux cas (voir tableau suivant).

		L'erreur de classification	Le nombre des attributs
sans la sélection des attributs	TE.A	22,093	224
	TE.Test1		
	TE.Test2		
Avec la sélection des attributs	TE.A	$7{,}7258\ e^{-11}$	118
	TE.Test1		
	TE.Test2		

Tableau 9 Qualité du système par rapport l'ensemble complet et le sous ensemble sélectionné.

Remarque

Normalement, le nombre de générations devrait être important, mais pour des raisons de temps d'exécution, nous nous somme arrêté à 20 générations, et nous devrions aussi adapter la taille de population par rapport 20 générations, mais nous nous sommes contenté de 2 parce que nous avons remarqué qu'il n'y a pas de grande différence entre 2 et 20, donc nous avons opté pour 2 générations.

Nous n'avons pas réadapté ces paramètres même après l'optimisation du système où le temps d'exécution s'est réduit au cinquième (gains de plusieurs heures), parce que ce critère a prouvé ses limites par rapport les bases de tests.

Limites de ce critère

- Nous ne connaissons pas le seuil de l'erreur de classification, pour lequel la reconnaissance réelle est bonne.
- Malgré l'erreur de classification est assez petite, mais l'erreur de reconnaissance est très importante.
- Ce critère améliore parfaitement l'erreur du système sur la base d'apprentissage, mais les erreurs sont très importantes par rapport les bases de tests.
- Il soufre généralement du problème de sur-apprentissage.

2. Critère d'évaluation : [2]Taux d'erreur de reconnaissance[2]

Nous avons effectué plusieurs expériences sur les trois bases en fixant les autres paramètres et jouant sur les valeurs de chacun de ″nombre de générations, nombre de cycles, taille de population″. Les résultats de ces expériences sont présentés dans les tableaux en dessous, et pour bien déterminer les valeurs optimales des paramètres par rapport les trois bases en même temps, nous construisons les courbes à partir des résultats des tableaux (voir figure).

Nombre de cycles (Nombre de cycles =1400 ; taille population=20).

Nombre de générations	2	6	8	10	16	20	24
Taux d'Erreur de l'apprentissage : **TE**	3.8462	0	0	0	0	0	0
Taux d'Erreur de la b.test1 :**TE1**	53.8462	61.5384	53.8462	38.4615	42.3077	46.1538	30.7692
Taux d'Erreur de la b.test2 :**TE2**	69.2307	73.0769	73.0769	57.6923	53.8462	61.5384	38.4615
Nombre d'attributs du vecteur sélectionné	122	114	113	112	115	124	111
Fonction d'adaptation	42.3077	44.8718	42.3077	32.0512	32.0512	35.2307	23.0769

Nombre de générations	30	40	50	100	150	200	300
Taux d'Erreur de la b.apprentissage	0	0	0	0	0	0	0
Taux d'Erreur de la b.test1	34.6153	30.7692	34.6153	30.7692	30.7692	15.3846	30.7692
Taux d'Erreur de la b.test2	46.1538	46.1538	42.3077	50	50	50	50
Nombre d'attributs du vecteur sélectionné	112	118	117	119	119	120	118
Fonction d'adaptation	26.9230	25.641	25.641.	26.9230	26.9230	21.7948	26.9230

Tableau 10 Le taux d'erreur par rapport le nombre de générations.

Nombre génération =200, taille population =20

Nombre de cycles	1400	3000	5000	10000
Taux d'Erreur apprentissage	0	0	0	
Taux d'Erreur de la b.test1	15.3846	34.6153	23.0769	
Taux d'Erreur de la b.test2	50	46.1538	46.1538	
Nombre d'attributs du vecteur sélectionné	120	111	102	
Moyenne	21.7948	26.9230	23.0769	

Tableau 11 Le taux d'erreur par rapport le nombre de cycles.

Nombre génération =200, Nombre de cycles =1400

taille population	5	10	15	20	50	70	100	130	150
TE	0	0	0	0	0	0	0	0	0
TE1	34.6153	30.7692	34.6153	15.3846	38.4615	34.6153	34.6153	30.7692	30.7692
TE2	42.3077	50	42.3077	50	26.9230	38.4615	42.3077	46.1586	42.3077
Nombre d'attributs	112	116	107	116	112	133	111	110	104
Moyenne	25.641	25.641	25.641	21.7948	21.7948	24,2048	25.608	25.641	24.3590

Tableau 12 Le taux d'erreur par rapport la taille de population.

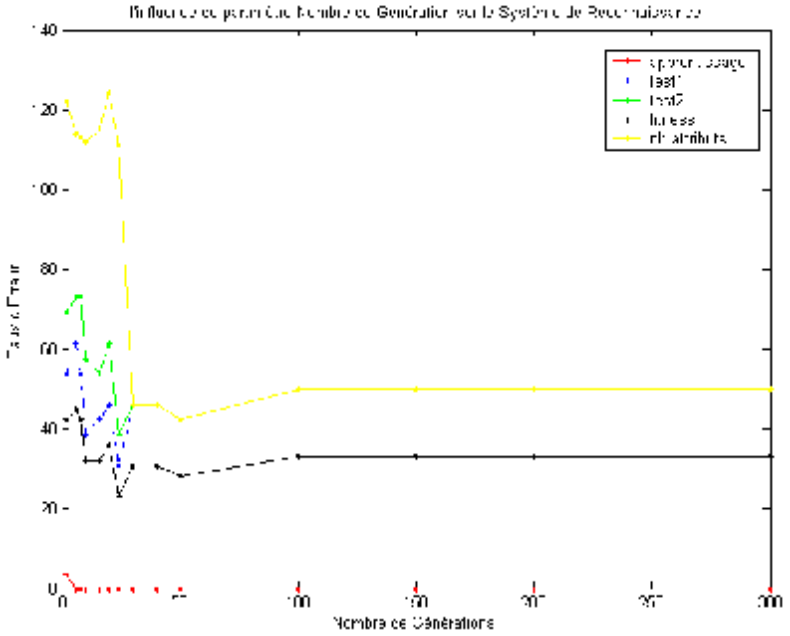


Figure 51 influence du paramètre nombre de génération sur le système de reconnaissance

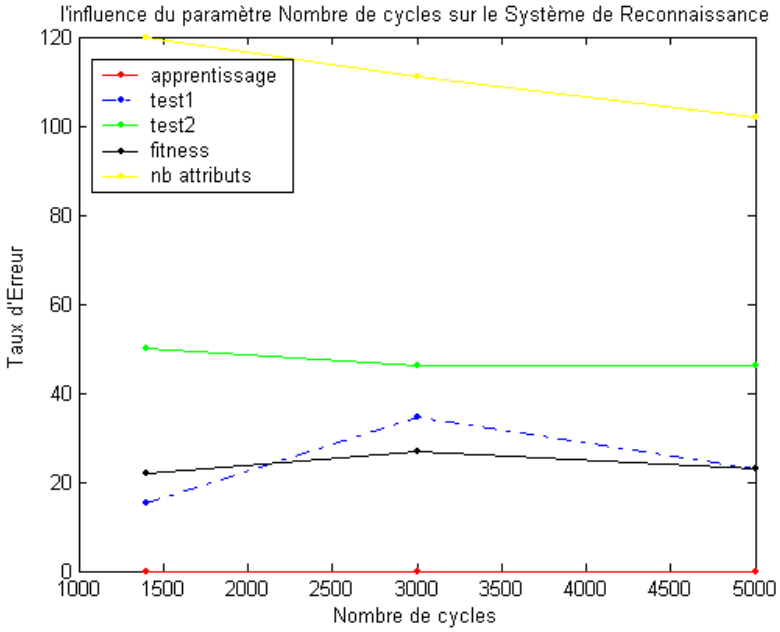


Figure 52 influence du paramètre nombre de cycles sur le système de reconnaissance

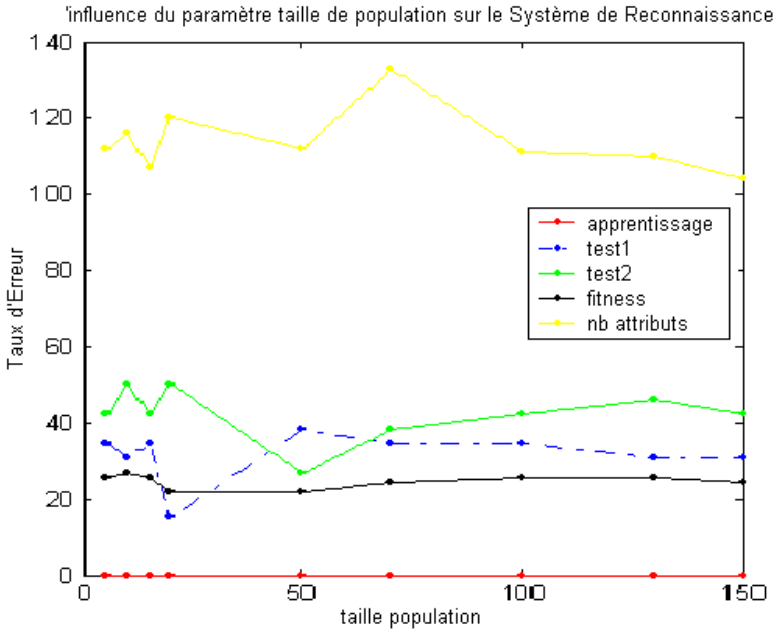


Figure 53 influence du paramètre taille de population sur le système de reconnaissance

En observant les courbes, Nous trouvons que les meilleurs résultats (la moyenne des taux d'erreurs des trois base est 21.7948) sont obtenus pour les valeurs des paramètres suivantes : *Nombre de cycles=1400, taille de population=20, nombre de génération=200*.

Etude de la complexité de l'algorithme proposé

Cette étude est faite grâce à des fonctions intégrées en Matlab, une partie de cette dernière est illustrés dans les tableaux suivants

Report generated 23-Jun-2007 01:25:16 (Complexité par rapport les parameters du sous ensemble d'attributs sélectionné)

Total recorded time:	10501.56 s
Number of M-functions:	93
Number of M-scripts:	6
Number of M-subfunctions:	50
Number of MEX-functions:	2
Clock precision:	0.0000001 s
Clock Speed:	800 Mhz

Section 1.01 Function List

Name	Time		Calls	Time/call	Self time	
main1	10501.2970000	100.0%	1	10501.29699999999900	0.5500000	0.0%
rn3	10478.5080000	99.8%	420	24.94882857142857	0.4030000	0.0%
network/train	10408.3150000	99.1%	420	24.78170238095227	0.3310000	0.0%
traingdx	10405.8090000	99.1%	840	12.38786785714277	1052.6250000	10.0%
calcgx	5204.4960000	49.6%	554849	0.00938002231238	828.0180000	7.9%
calcgrad	2162.7460000	20.6%	554849	0.00389790014941	2023.3910000	19.3%
setx	2055.1070000	19.6%	588000	0.00349507993197	2055.1070000	19.6%
calcperf	1958.3720000	18.6%	588420	0.00332818734917	1884.3560000	17.9%
formgx	1827.5330000	17.4%	554849	0.00329374838920	1827.5330000	17.4%
dmse	369.8830000	3.5%	1109698	0.00033331861461	268.8470000	2.6%
cell2mat	135.4850000	1.3%	1116418	0.00012135687529	107.4950000	1.0%
mse	90.5070000	0.9%	1144949	0.00007904893580	90.5070000	0.9%
dnetsum	90.2100000	0.9%	2219396	0.00004064619383	90.2100000	0.9%
plotperf	88.6110000	0.8%	3360	0.02637232142857	62.4830000	0.6%
newff	53.0600000	0.5%	420	0.12633333333328	0.8440000	0.0%
network/subsasgn	50.7450000	0.5%	13020	0.00389746543778	6.5720000	0.1%
etime	44.1980000	0.4%	588420	0.00007511301451	37.8030000	0.4%
network/subsasgn/hint	31.7680000	0.3%	13020	0.00243993855606	26.3400000	0.3%
network/sim	29.8000000	0.3%	1260	0.02365079365084	0.6760000	0.0%
plotperf/new_figure	20.8500000	0.2%	420	0.04964285714277	20.8500000	0.2%
network/subsref	19.6170000	0.2%	32340	0.00060658627087	19.6170000	0.2%
getx	16.7920000	0.2%	1680	0.00999523809522	5.7390000	0.1%
ddotprod	15.0850000	0.1%	1109698	0.00001359378858	15.0850000	0.1%

isobject	13.1340000	0.1%	1113058	0.00001179992417	13.1340000	0.1%

Nous remarquons que pour les paramètres suivants : ″200 génération et 20 individus″ l'algorithme d'apprentissage s'exécute 420 fois, la durée moyenne de chaque exécution est de 24 seconde, pour cela notre système s'exécute lentement.

Remarque

- L'algorithme de sélection d'attributs doit s'exécuter une seule fois tant que nous n'avons pas changé ni la base d'apprentissage, ni l'ensemble complet des attributs (le système de reconnaissance s'exécute en utilisant le meilleur sous ensemble sélectionné précédemment).
- La complexité de l'algorithme de sélection change selon les valeurs des paramètres utilisées.

Comparaison entre le sous ensemble sélectionné et l'ensemble complet des attributs

Afin de voir si l'algorithme proposé répand aux objectifs pour lesquelles il a été conçu, nous validons le sous ensemble d'attributs sélectionné en le comparant avec l'ensemble complet des attributs.

sans la sélection des attributs (avec l'ensemble complet des attributs)	Nombre de cycles	1400	3000	5000	Nombre d'attributs
	Base d'apprentissage	0	0	0	224
	Base test1	65.3846	57.6923	50	
	Base test2	73.0769	73.0769	80.7692	
	Moyenne (fitness)	46.1538	43.5897	43.5897	
Avec la sélection des attributs	Base d'apprentissage	0	0	0	116
	Base test1	15.3846	34.6153	23.0769	
	Base test2	50	46.1538	46.1538	
	Moyenne (fitness)	21.7948	26.9230	23.0769	

Tableau 13 Qualité du système par rapport l'ensemble complet et le sous ensemble sélectionné.

Complexité du système	
sans la sélection des attributs (avec l'ensemble complet des attributs)	Total recorded time: 33.81 s Number of M-functions: 86 Number of M-subfunctions: 50 Number of MEX-functions: 2 Clock precision: 0.0000001 s Clock Speed: 792 Mhz
Avec la sélection des attributs	Total recorded time: 12.76 s Number of M-functions: 70 Number of M-subfunctions: 50 Number of MEX-functions: 2 Clock precision: 0.0000001 s Clock Speed: 800 Mhz

Tableau 14 Complexité du système par rapport l'ensemble complet et le sous ensemble sélectionné.

En observant les tableau au dessus nous pouvons conclure que :

1. l'algorithme proposé a réduit le nombre d'attributs de 49 %.
2. l'algorithme proposé a amélioré la qualité du système de 54 %.
3. l'algorithme proposé a réduit la complexité du système de 54 %.

L'algorithme proposé est robuste au bruit. Donc l'algorithme proposé conçu a répandu aux objectifs pour lesquelles a été conçu.

²GGA² Critère d'évaluation : ² Taux d'erreur² On refait la même chose pour le GGA .

Taille population=20 ; nombre de cycle =5000.

Nombre de générations	2				6	10	16	20	24
TE apprentissage	7.6923	19.2308	11.5385 (107)	11.5385 (108)	0 (123)	7.6923 (117)	3.8462 (122)	3.8462 (117)	0 (106)
TE la b.test1	65.3846	76.9231	84.6154	80.7692	73.0769	76.9231	80.7692	76.9231	69.2308
TE de la b.test2	/	80.7692	96.1538	84.6154	88.4615	84.6154	92.3077	80.7692	80.7692

Tableau 15 Le taux d'erreur par rapport le nombre de générations.

Nombre de génération=6(meilleur est 24) ; nombre de cycle =5000.

5	10		15		20	30		40		50	
3.84 (125)	3.84 (113)	0 (124)	7.6923 (124)	3.84 (115)	0 (123)	15.38 (108)	0(118)	0 (111)	3.84 (109)	0 (117)	0(109)
84.61	73.07	80.76	88.46	76.92	73.07	84.61	73.07	69.23	69.23	84.61	69.23
100	88.46	88.46	76.92	88.46	84.61	84.61	76.92	88.46	80.76	80.76	80.76

Tableau 16 Le taux d'erreur par rapport la taille de la population.

En observant les tableaux, Nous trouvons que les meilleurs résultats (la moyenne des taux d'erreurs des trois bases) sont obtenus pour les valeurs des paramètres suivantes :
Nombre de cycles=5000, taille de population=40, nombre de génération=24.

Comparaison entre le sous ensemble sélectionné et l'ensemble complet des attributs

Afin de voir si l'algorithme proposé répand aux objectifs pour lesquelles il a été conçu, nous validons ce dernier en le comparant avec l'ensemble complet des attributs.

		L'erreur de classification	Le nombre des attributs
sans la sélection des attributs	A	0	224
	T1	50	
	T2	80.7692	
Avec la sélection des attributs Nb génération=24 ; Nombre de cycle =5000. Taille pop =40.	A	0 %	106
	T1	65.3846 %	
	T1	76.92 %	

Tableau 17 Complexité du système par rapport l'ensemble complet et le sous ensemble sélectionné.

Remarque

Nous nous somme arrêté à 24 générations parce que cet algorithme est très gourmand en temps de calculs. ″Sa complexité est très élevée″

Nous pouvons conclure que :

4. l'algorithme proposé a réduit le nombre d'attributs de 53 %.
5. Il y a un certain maintien de la qualité du système.
6. l'algorithme proposé a réduit la complexité du système de 60 %.
7. l'algorithme proposé est robuste au bruit.

Donc l'algorithme proposé conçu a répandu aux objectifs pour lesquelles a été conçu

Conclusion

Dans cette partie nous avons proposé deux nouvelles méthodes de sélection d'attributs, basées sur les techniques d'optimisations tel que les algorithmes génétiques ″SSGA/ GGA″ pour l'exploration de l'espace de recherche, et le Perceptron multicouche pour l'évaluation des sous ensembles d'attributs. Nous avons validé ces algorithmes par différentes expériences qui ont prouvé leurs efficacités.

Conclusion générale

Le but de ce travail est le traitement du problème de sélection d'attributs, qui consiste à réduire l'ensemble complet des attributs à un sous ensemble ne contenant pas celles qui sont inutiles et redondant et non pertinents, tout en améliorant la qualité et les performances du système à lequel elle est destinée. Pour la mettre en œuvre un domaine d'application doit être choisi. Nous avons opté pour la reconnaissance hors ligne de caractères manuscrits isolés qui est connu par son utilisation d'un grand nombre d'attributs pour la description des caractères. De ce fait elle a besoin de l'étape de sélection pour ne laisser que les attributs qui lui permettre de faire une reconnaissance fiable et robuste. Ce système de reconnaissance est constitué de modules.

- Le premier sert à bien préparer et nettoyer les images des caractères pour le reste de modules, les prétraitements effectués sont la binarisation, normalisation, élimination du bruit.

- Le deuxième élément est l'extraction de caractéristique qui permet d'extraire les informations qui ont un pouvoir discriminant (ces informations doivent présenter une certaine invariance géométrique et statistique), plusieurs méthodes existent dans la littérature, mais aucune d'entre elles, à elle seule, ne peut capter l'information pertinente d'une image donnée de manière idéale. Par conséquent, un amalgame de plusieurs types de caractéristiques nous a semblé nécessaire. Nous avons retenu, parmi ce qui existe, le zonage, la surface, la projection horizontale et verticale, le profil gauche et droit. Le problème de cette étape est que la dimension de l'espace de représentation est assez importante (224 attributs utilisés dans le cadre de ce travail), et parmi les attributs extraits, certains ne sont pas aussi pertinents. Il est possible que certains correspondent à du bruit ou qu'ils soient peu informants, corrélés ou même inutile au système de reconnaissance. Pour surmonter ce problème nous avons introduit l'étape de sélection d'attribut juste après ce module pour réduire les entrées du classifieur, en lui fournissant les attributs utiles et pertinents, qui lui permettent d'accomplir sa tache avec efficacité.

Comme nous avons vu auparavant, que notre première contribution est au niveau de l'extraction de caractéristiques, le fait de combiner plusieurs méthodes existantes qui nous ont semblés intéressantes, mais la principale contribution de notre travail est bien le développement de deux algorithmes de sélection d'attributs basées sur un algorithme génétique neural. Dans le premier nous avons proposé un algorithme génétique stationnaire pour explorer les différents sous ensembles des attributs, sa sélection des meilleurs sous ensembles est basée sur les taux d'erreur donnés par le perceptron multicouches ayant comme entrées les différents sous ensembles. Le critère d'arrêt choisi est le nombre de génération. Le deuxième est exactement comme le premier sauf que l'algorithme génétique est générationnel, c à d à chaque génération c'est toute une population de solutions qui va se reproduire.

Nous avons testé les algorithmes de sélection d'attributs sur un échantillon de la base des caractères standard CEDAR. Après l'adaptation des paramètres des algorithmes utilisés (Algorithme génétique :la taille de population, le nombre de génération, les opérateurs de reproduction. Perceptron : le nombre de cycles, le seuil d'apprentissage, nombre de neurones de la couche caché), les résultats expérimentaux obtenus dans ce cadre montre l'efficacité des algorithmes proposés :

1. le nombre d'attributs s'est réduit de 52 % et le taux d'erreur s'est réduit de 26% et le temps de calcul s'est réduit de 68%.
2. le nombre d'attributs s'est réduit de % et le taux d'erreur s'est réduit de % et le temps de calcul s'est réduit de %.

Nous concluons que notre objectif est atteint, qui est la réduction du nombre d'attributs et l'amélioration de la qualité du système de reconnaissance par rapport l'ensemble complet des attributs, mais l'erreur reste comme même considérable. Elle est due à certains points, évoqués dans la section des perspectives.

Abstract

The vision task is a spiny problem standing at the level superior in the hierarchy of the most computational tasks in computer science, because of the important features volume used for the description of the object. To surmount this problem a phase of features selection is introduced. The main idea of the selection is to choose a subset of features from the complete set by eliminating the redundant and irrelevant features, which decrease the complexity of the system and ameliorate its quality. This process is classified among NP hard problems because for N features we have a 2^N-1 subsets possible (it is considered like a combinative optimization problem). To develop and implement a method of features selection, a domain application had to be chosen. The writing recognition needs to identify the relevant features facilitating a reliable and robust recognition, it appears like a privileged application domain for this survey.

In this memory we proposed two methods of features selection based on a ***neural genetic algorithm***, where we used the genetic algorithm (steady state / generational) to explore the different features subsets and the perceptron multilayered to evaluate them.

The proposed methods are applied on a sample of the standard character base CEDAR. The experimental results gotten show the efficiency of the proposed method, where the number of features goes down to 52% and the error rate cut down of 26% and the computation time is reduced of 68%.

Sélection d'Attributs basée sur un Algorithme Génétique Neural : Application Reconnaissance des Caractères Manuscrits.

Mémoire de Magister en Informatique option Intelligence Artificielle et Génie Logiciel.

Résumé

La tâche de vision est un problème épineux se plaçant au niveau supérieur dans la hiérarchie des tâches de l'informatique les plus computationelles à cause du volume d'attributs important utilisé pour la description des objets. Pour surmonter ce problème une phase de *sélection d'attributs* est introduite. L'idée principale de la sélection est de choisir un sous ensemble d'attributs de l'ensemble complet tout en éliminant les attributs redondants et non pertinents, ce qui va diminuer la complexité du système et améliorer sa qualité. Ce processus est classé parmi les problèmes ***NP difficiles*** puisque pour **N** attributs nous avons $\mathbf{2^N-1}$ sous ensembles possibles (il est considéré comme un problème d'optimisation combinatoire). Pour développer et mettre en œuvre une méthode de sélection d'attributs, un domaine d'application devait être choisi. *La reconnaissance de l'écriture* de part son besoin d'identification des caractéristiques pertinentes facilitant une reconnaissance fiable et robuste apparaît comme un domaine d'application privilégié pour cette étude. Dans ce mémoire nous avons proposé deux méthodes de sélection d'attributs basées sur un *algorithme génétique neural*, où nous avons utilisé l'algorithme génétique (stationnaire/ générationnel) pour explorer les différents sous ensembles d'attributs et le perceptron multicouches pour les évaluer. Les méthodes proposées sont appliquées sur un échantillon de la base de caractères standard *CEDAR*. Les résultats expérimentaux obtenus dans ce cadre montre l'efficacité des méthodes proposées où le nombre d'attributs s'est réduit de **52 %** et le taux d'erreur s'est réduit de **26%** et le temps de calcul s'est réduit de **68%.**

Mot clés : Sélection d'attributs, reconnaissance de l'écriture, Algorithme génétique stationnaire/ Générationnel, perceptron multicouches.

ملخص

مهمة الابصار تعد مشكلة عويصة, لاتها تندرج بين مهمات الاعلام الالي الاكثر استهلاكا للوقت. وهذا راجع للكم الكبير من الخاصيات التي تستعملها في وصف الاشياء. و لتجاوز هذا المشكل, لابد من ادماج مرحلة انتقاء الخاصيات في هذه المهمة. الفكرة الرئسية هي اختيار مجموعة دوتية من المجموعة الكاملة للخاصيات وذلك بحذف الخاصيات الغيرملائمة و المسهبة, وهذا يقلل من تعقيد النظام ويحسن من نوعيته.

عملية الانتقاء هذه تندرج ضمن مشاكل NP الصعبة, لانه من اجل **ن** خاصية هتاك $\mathbf{2^{ن}-1}$ مجموعة دونية ممكنة (انها تعتبرمن بين مشاكل التوسع التركيبي).

لانجاز عملية انتقاء الخاصيات لابد من اختيار مجال للتطبيق.و مادام معرفة الكتابة تحتاج الى انتقاء الخاصيات الملائمة لتسهيل معرفة موثوقة و قوية. فقدبدت لنا التطبيق المناسب لهذه الدراسة.

في هدا البحث قمنا باقتراح طريقتي انتقاء معتمدتين على خوارزم جيني عصبي, اين استعملنا خوارزم الجينات لاستكتشاف مختلف المجموعات الدوتية للخاصيات. perceptronمتعدد الطيقات لتقييم هذه المحموعات , وذلك باعطائها نسب اخطائها بالنسبة للنظام. تم تطبيق الطريقتين المقترحتين على عينة من CEDAR قاعدةالحروف النموذج.

النتائج التجريبية المتحصل عليها في نطاق هذا العمل تثبت فعالية الطريقتين المقترحتين, بحيث تم اختزال عدد من الخاصبات 52% ونسبة الخطاء ب 26% وقت الحساب ب 68% .