Université Paris Descartes
UFR de Mathématiques et Informatique
ED Informatique, Télécommunications et Electronique

Thèse présentée pour l'obtention du grade de
Docteur de l'université Paris Descartes
Discipline: **Informatique**

Sujet de thèse:

Sélection de caractéristiques: méthodes et applications

Présentée par:
Hassan CHOUAIB

Direction de thèse : **Pr. Nicole Vincent**
Co-encadrement de thèse : **Dr. Florence Cloppet**

Soutenue le 8 juillet 2011, devant le jury composé de:

Pr. Djamel Abdelakder Zighed	Université Lumière Lyon 2	Rapporteur
Pr. Josep Llados	Université Autonoma de Barcelone	Rapporteur
Pr. Jean-Marc Ogier	Université de La Rochelle	Examinateur
Pr. Jin-Kao Hao	Université d'Angers	Examinateur
Pr. Salvatore-Antoine Tabbone	Université Nancy 2	Examinateur
Pr. Nicole Vincent	Université Paris Descartes	Directrice de thèse
Dr. Florence Cloppet	Université Paris Descartes	Co-encadrante de thèse

RÉSUMÉ DE LA THÈSE

Dans de nombreux domaines (vision par ordinateur, reconnaissance des formes, etc.), la résolution de la plupart des problèmes se base sur le traitement de données extraites à partir des données acquises dans le monde réel, et structurées sous forme de vecteurs. La qualité du système de traitement dépend directement du bon choix du contenu de ces vecteurs. Mais dans de nombreux cas, la résolution du problème devient presque impossible à cause de la dimension trop importante de ces vecteurs. Par conséquent, il est souvent utile, et parfois nécessaire, de réduire celle-ci à une taille plus compatible avec les méthodes de résolution, même si cette réduction peut conduire à une légère perte d'informations. Dans ce cadre, nous proposons dans cette thèse une nouvelle méthode rapide de sélection de caractéristiques. Les méthodes existantes présentent des faiblesses au niveau de leur complexité très élevée, de la dépendance des caractéristiques pertinentes sélectionnées par rapport au classificateur utilisé, de la redondance entre les caractéristiques sélectionnées ainsi que des interactions entre les caractéristiques. Dans le but de limiter ces inconvénients, la méthode proposée est basée sur la construction et la sélection de classificateurs simples associés à chacune des caractéristiques. Nous proposons d'utiliser des algorithmes génétiques, monoobjectifs et multiobjectifs, afin de trouver une bonne combinaison des classificateurs simples.

Les expérimentations ont montré que notre méthode est rapide, qu'elle a la capacité à sélectionner un nombre réduit de caractéristiques tout en conservant des taux de classification très satisfaisants. Les performances de la méthode proposée sont mises en évidence à travers une comparaison avec d'autres méthodes de la littérature du domaine.

Enfin notre méthode a été appliquée dans deux applications concernant des domaines bien différents, celui de l'indexation de lettrines extraites de documents anciens et celui de l'analyse de données biologiques. Nous avons montré qu'un petit nombre de caractéristiques suffisait aux historiens pour indexer automatiquement une base de lettrines. Dans le domaine de la biologie, la classification de molécules selon des cibles bien précises définies par les biologistes a été améliorée en qualité tout en diminuant de 78% le nombre des descripteurs moléculaires.

Mots clefs : *sélection de caractéristiques, algorithmes génétiques, optimisation multiobjectifs, combinaison de classificateurs, diversité de classificateurs, classification.*

ABSTRACT

In many domains such as computer vision or pattern recognition, solving a problem is based on processing data extracted from a set of real world data acquired by means of sensors or resulting from some data processing. Data are structured as vectors. The quality of a processing system highly depends on the choice of these vector content. However, in many cases the vectors' high dimensionality makes it almost impossible to use them to solve the problem, both because of the data themselves and of the learning set size. Hence, it is usually recommended, and sometimes required to reduce the vector size in order to make them more usable, even if the reduction might lead to information loss. Sometimes, solving complex problems with large descriptors can also be accomplished using a small set of features selected from initial data set. This can be done if the selected features are relevant with respect to the considered problem. Reducing vector dimensionality is often considered as a pre-processing step dedicated to noise and redundant information elimination. One type of dimensionality reduction methods is feature selection. It consists in selecting the most relevant features from an initial set.

Existing feature selection methods reveal limitations on many levels such as complexity, interaction between the features, dependence on the evaluation classifier, and so on.

In this thesis, in order to limit these drawbacks, we propose a fast selection method based on a genetic algorithm. Each feature is closely associated with a single feature classifier. We propose to use multi-objectives and mono-objective genetic algorithms, to find a good combination of simple classifiers.

Experiments have shown that our method is fast, it has the ability to select a small number of features while maintaining good classification rates. The performances of the proposed method are shown through a comparison with other state of art methods.

Finally, our method was applied in two types of applications: the indexing of drop caps extracted from old documents and biological data analysis. We have shown that a small number of features is enough for historians to index automatically a drop caps database. In the biology field, the quality of the molecules classification system according to specific classes identified by the biologists, has been improved while reducing the number of molecular descriptors by 78%.

Keywords : *feature selection, genetic algorithm, multi-objectives optimization , classifier's combinaison, classifier's diversity, classification.*

REMERCIMENTS

Je tiens à exprimer ma profonde gratitude à Nicole Vincent , professeur à l'univers- ité Paris Descartes, pour avoir encadré et dirigé mes recherches. Je la remercie pour m'avoir soutenu et appuyé tout au long de ma thèse. Ses précieux conseils, son exigence et ses commentaires ont permis d'améliorer grandement la qualité de mes travaux et de ce mémoire. Sincèrement, grâce à elle, j'ai pu apprendre beaucoup de choses dont certaines fort utiles pour mes travaux académiques bien sûr, mais aussi des choses importantes pour mon développement personnel. Je n'oublie pas enfin son aide précieuse dans la relecture et la correction de ma thèse.

Merci à Florence Cloppet, maître de conférences à l'université Paris Descartes, pour son co-encadrement de la thèse. Je la remercie pour ses conseils, sa bonne humeur, sa collaboration ainsi que son aide dans la relecture et la correction de ma thèse.

Je remercie infiniment Salvator-Antoine Tabbone, professeur à l'université Nancy II. Je le remercie pour son soutien depuis mon Master jusqu'à la fin de ma thèse à tous les niveaux. Son aide et ses conseils m'ont chaque fois permis de rebondir dans les moments difficiles. je le remercie vivement pour l'aide scientifique précieuse et tous les conseils qu'il a pu me fournir pendant la durée de cette thèse.

Cette thèse a été réalisée dans le cadre du projet ANR (Navidomass) en collaboration avec plusieurs équipes de recherches. Je tiens à remercier Jean-Marc Ogier, chef du projet Navidomass et professeur à l'université de La Rochelle pour m'avoir fait confiance et permis de travailler dans le cadre de ce projet.

Je tiens également à remercier Josep Llados et Djamel Abdelkader Zighed pour avoir accepté de rapporter mon travail et pour leurs remarques constructives. Je remercie aussi Jin-kao Hao, Jean-marc Ogier et Salvator-Antoine Tabbone pour avoir accepté de participer à mon jury de thèse.

Mes remerciements vont à Georges Stamon pour son soutien et ses conseils mais surtout pour ses qualités humaines.

Mes remerciements s'adressent également à tous les membres de l'équipe SIP. Merci à Laurent Wendling, Marwen, Nam-Jun, Arnaud, Nicolas Champion, Rabie, Khurram et Imran pour les moments agréables que nous avons passés ensemble et pour leur soutien dans les moments délicats.

A tous mes amis un grand merci. Merci à Hossein et sa famille, Wassim, Jad, Hassan Wehbé et à tous les amis qui m'ont soutenu de proche ou de loin.

A vous mes parents, je dis un grand merci. je vous suis infiniment reconnaissant pour votre soutien et vos encouragements.

Je remercie également mes sœurs qui m'ont toujours encouragé et soutenu moralement. Un remerciement spécial pour mon frère, sans qui, je n'aurai jamais eu l'opportunité de continuer mes études en France ni d'effectuer cette thèse.

Enfin et surtout, je remercie ma fiancée pour ses encouragements et son soutien. Malgré la distance qui nous séparait, elle a toujours été disponible et compréhensive. Je la remercie aussi pour son écoute et surtout son amour qui m'a été essentiel durant ces dernières années.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction générale

Dans de nombreux domaines (vision par ordinateur, reconnaissance de formes, etc.), la résolution des problèmes se base sur le traitement de données extraites à partir des données acquises dans le monde réel, et structurées sous forme de vecteurs. La qualité du système de traitement dépend directement du bon choix du contenu de ces vecteurs. Mais dans de nombreux cas, la résolution pratique du problème devient presque impossible à cause de la dimensionnalité trop importante de ces vecteurs. Par conséquent, il est souvent utile, et parfois nécessaire, de réduire celle-ci à une taille plus compatible avec les méthodes de résolution, même si cette réduction peut conduire à une légère perte d'informations. Parfois, la résolution de phénomènes complexes avec des descripteurs de grande taille pourrait être gérée en utilisant peu de caractéristiques extraites des données initiales, il suffit qu'elles représentent les variables pertinentes pour le problème à résoudre.
Une méthode de réduction de la dimensionnalité est souvent définie comme un processus de pré-traitement de données qui permet de supprimer les informations redondantes et bruitées. Avec l'accroissement de la quantité de données, mises à disposition, la redondance et le bruit dans les informations sont toujours présents. cette multiplication de données n'est pas sans introduire de bruit qui vient complexifier la résolution du problème.
Les méthodes de réduction de la dimensionnalité sont généralement classées en deux catégories :

- L'extraction de caractéristiques qui permet de créer de nouveaux ensembles de caractéristiques, en utilisant une combinaison des caractéristiques de l'espace de départ ou plus généralement une transformation effectuant une réduction du nombre de dimensions.
- La sélection de caractéristiques qui regroupe les algorithmes permettant de sélectionner un sous-ensemble de caractéristiques parmi un ensemble de départ, en utilisant divers critères et différentes méthodes.

L'approche par sélection permet de mieux appréhender la modélisation d'un problème et de limiter les mesures qui permettent la résolution du problème, par contre l'extraction de

nouvelles caractéristiques conserve une vision globale des observations et ne permet pas d'économiser des mesures pour décrire le phénomène observé. Dans ce mémoire, nous nous intéressons aux techniques de sélection de caractéristiques.
Comme mentionné précédemment, la sélection de caractéristiques est une technique permettant de choisir les caractéristiques, variables ou mesures les plus intéressantes, pertinentes, adaptées à un système de résolution d'un problème particulier. La difficulté des problèmes à résoudre et la masse des données disponibles conduisent à la complexification des systèmes. Une phase de sélection constitue alors un module important qui est intégré au système complexe. Les domaines d'application des techniques de sélection de caractéristiques sont variés, notons par exemple la modélisation, la classification, l'apprentissage automatique (Machine Learning), l'analyse exploratoire de données (Data Mining) et la reconnaissance de formes. Dans ce mémoire, nous nous intéressons plus particulièrement à la sélection de caractéristiques pour la classification et la reconnaissance de formes.
Une sélection de caractéristiques présente plusieurs avantages liés à la réduction de la quantité de données (moins de caractéristiques). D'une part, cette réduction rend beaucoup plus facile de gérer les données et d'autre part, elle aide à mieux comprendre les résultats fournis par un système basé sur ces caractéristiques. Par exemple, pour un problème de classification, ce processus de sélection ne réduit pas seulement le temps d'apprentissage mais il aide aussi à mieux comprendre les résultats fournis par le classificateur et à améliorer parfois la précision de la classification, en favorisant les caractéristiques les moins bruitées par exemple.

1.1 Motivation et objectifs de la thèse

Les méthodes de sélection de caractéristiques sont classées généralement en deux groupes : les méthodes *"filter"* et les méthodes *"wrapper"*. La première approche (méthodes de filtrage) utilise des mesures statistiques calculées sur les caractéristiques afin de filtrer les caractéristiques peu informatives. Cette étape est généralement réalisée avant d'appliquer tout algorithme de classification. Ces méthodes de filtrage présentent des avantages au niveau de leur efficacité calculatoire et de leur robustesse face au sur-apprentissage. Mais elles ne tiennent pas compte des interactions entre caractéristiques et tendent à sélectionner des caractéristiques comportant des informations redondantes plutôt que complémentaires. De plus, ces méthodes ne tiennent absolument pas compte des choix faits pour la méthode de classification par exemple qui suit la sélection.
La seconde approche (méthodes enveloppantes ou *"wrapper"*) est plus coûteuse en temps de calcul, mais en contre-partie, elle est souvent plus précise. Un algorithme de type *"wrapper"* explore l'espace des sous-ensembles de caractéristiques afin de trouver un sous-ensemble optimal pour un algorithme d'induction bien défini. Les sous-ensembles de caractéristiques sélectionnés par cette méthode sont bien adaptés à l'algorithme de classification utilisé, mais ils ne restent pas forcément valides si on change le classificateur. La complexité de

l'algorithme d'apprentissage rend les méthodes *"wrapper"* très coûteuses en temps de calcul. Les méthodes *wrapper* sont généralement considérées comme étant meilleures que celles de filtrage et de plus, elles sont capables de sélectionner des sous-ensembles de caractéristiques de plus petite taille, néanmoins aussi performants pour le classificateur utilisé. Les méthodes *"wrapper"* présentent des limitations, d'une part au niveau de la complexité et du temps de calcul nécessaire pour la sélection et d'autre part par la dépendance des caractéristiques pertinentes sélectionnées au classificateur utilisé.
Dans cette thèse, notre but est de limiter les inconvénients liés à ces deux types de méthodes tout en conservant leurs avantages respectifs. Nous proposons une nouvelle méthode de sélection de caractéristiques qui tend à optimiser deux aspects :

- La rapidité du processus de la sélection.
- La possibilité de prendre en compte les interactions entre caractéristiques.

1.2 Organisation du manuscrit

La suite de ce mémoire est divisée en trois parties :

- État de l'art
- Une nouvelle méthode de sélection de caractéristiques
- Applications

La première partie, qui est composée de trois chapitres est consacrée à l'état de l'art : Le chapitre 2, présente l'état de l'art des techniques de réduction de dimensionnalité par sélection et par extraction de caractéristiques. Nous présentons en détail les techniques de sélection de caractéristiques ainsi que leurs avantages et leurs limitations. Une revue de quelques méthodes de sélection de caractéristiques est effectuée. La deuxième partie du chapitre est consacrée à une présentation synthétique de la réduction de dimensionnalité par extraction de caractéristiques.
Le chapitre 3 présente le formalisme de la notion d'apprentissage automatique. Il permet également de présenter un état de l'art des algorithmes de classification usuels, de leurs limitations, et de la manière avec laquelle ils abordent chacun le problème de la classification supervisée. Dans une deuxième partie de ce chapitre, nous présentons l'approche basée sur les ensembles de classificateurs et nous précisions les principaux algorithmes ensemblistes. La première partie s'achève avec le chapitre 4 qui présente rapidement les algorithmes génétiques ainsi que les techniques d'optimisation multi-objectifs et leur utilisation dans le domaine de la sélection de caractéristiques.

La deuxième partie introduit la nouvelle méthode de sélection de caractéristiques que nous proposons. Deux chapitres sont consacrés à cette présentation. Dans le chapitre 5,

nous présentons en détail cette nouvelle méthode de sélection, le processus général et ses différentes étapes. Nous validons notre approche et les choix que nous avons réalisés par une phase d'expérimentation et nous montrons la capacité de notre méthode à sélectionner un nombre réduit de caractéristiques tout en conservant des taux de classification très satisfaisants. Finalement, nous terminons ce chapitre par une comparaison entre notre méthode et d'autres méthodes représentatives de la littérature.
La richesse de notre approche est mise en évidence dans le chapitre 6 par plusieurs études qui y sont menées. La fonction de *fitness* de l'algorithme génétique auquel nous avons eu recours dans la méthode développée dans le chapitre 5 y est enrichie, le contrôle du nombre de caractéristiques sélectionnées est rendu possible et nous avons pu introduire une hiérarchisation des caractéristiques.

Notre méthode de sélection de caractéristiques est appliquée dans la troisième partie à différentes applications choisies dans des domaines variés, l'indexation des lettrines extraites des documents anciens et la sélection de caractéristiques pour des données biologiques.

Enfin, la conclusion générale présente une synthèse des contributions apportées ainsi que les pistes définissant des perspectives possibles pour de futurs travaux.

Première partie

État de l'art

Chapitre 2

Réduction de la dimensionnalité

Chercher à réduire la dimensionnalité d'un ensemble de données devient de plus en plus indispensable en raison de la multiplication des données. Dans de nombreux domaines, le système de résolution d'un problème est fondé sur un ensemble des variables (caractéristiques). L'augmentation du nombre de ces variables (caractéristiques) qui modélisent le problème introduit des difficultés à plusieurs niveaux comme la complexité, le temps de calcul ainsi que la détérioration du système de résolution en présence de données bruitées. Une méthode de réduction de la dimensionnalité consiste à trouver une représentation des données initiales dans un espace plus réduit. Les méthodes de réduction de la dimensionnalité sont généralement classées dans deux catégories :

- Une réduction basée sur une **sélection de caractéristiques** qui consiste à sélectionner les caractéristiques les plus pertinentes à partir de l'ensemble de données des variables décrivant le phénomène étudié.
- Une réduction basée sur une **transformation des données** appelée aussi une extraction de caractéristiques et qui consiste à remplacer l'ensemble initial des données par un nouvel ensemble réduit, construit à partir de l'ensemble initial de caractéristiques.

Dans ce chapitre, nous présenterons tout d'abord les méthodes de réduction par une sélection de caractéristiques en détaillant le processus de sélection, tout en donnant les avantages et les inconvénients des différentes techniques de sélection. Ensuite, nous détaillerons les techniques de réduction par une transformation de données, en présentant différentes approches linéaires et non linéaires.

2.1 Réduction basée sur une sélection de caractéristiques

Après avoir précisé l'objectif de cette approche, défini la pertinence d'une caractéristique et les différentes étapes qui interviennent classiquement dans les systèmes reposant sur un tel principe, nous présenterons quelques méthodes développées dans la littérature et que nous avons choisies en fonction de leur représentativité dans le domaine.

2.1.1 Définition de la sélection

La sélection de caractéristiques est généralement définie comme un processus de recherche permettant de trouver un sous-ensemble *"pertinent"* de caractéristiques parmi celles de l'ensemble de départ. La notion de pertinence d'un sous- ensemble de caractéristiques dépend toujours des objectifs et des critères du système. En général, le problème de sélection de caractéristiques peut être défini par :
Soit $F = \{f_1, f_2, .., f_N\}$ un ensemble de caractéristiques de taille N où N représente le nombre total de caractéristiques étudiées. Soit Ev une fonction qui permet d'évaluer un sous-ensemble de caractéristiques. Nous supposons que la plus grande valeur de Ev soit obtenue pour le meilleur sous-ensemble de caractéristiques. L'objectif de la sélection est de trouver un sous-ensemble $F'(F' \subseteq F)$ de taille $N'(N' \leq N)$ tel que :

$$Ev(F') = \max_{Z \subseteq F} Ev(Z) \tag{2.1}$$

Où $|Z| = N'$ et N' est, soit un nombre prédéfini par l'utilisateur ou soit contrôlé par une des méthodes de génération de sous-ensembles que nous décrirons dans la section 2.1.3.1.
Une procédure générale proposée par (Dash et Liu [1997]) pour une méthode de sélection de caractéristiques est illustrée par la figure 2.1.

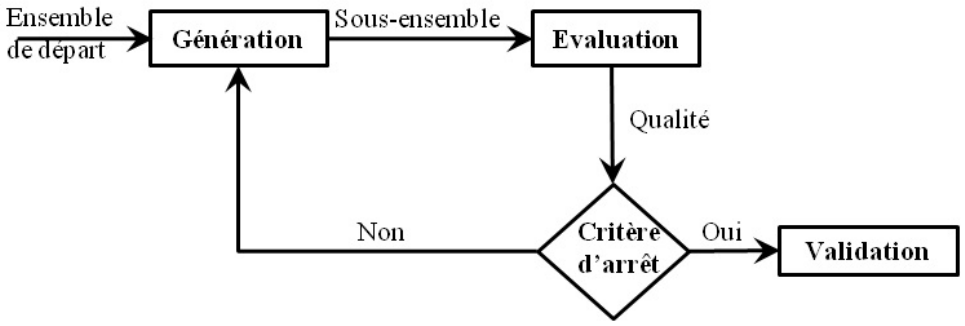

Figure 2.1 – Procédure générale d'un algorithme de sélection de caractéristiques

Il existe trois types de stratégies de sélection de caractéristiques :
Dans la première stratégie, la taille du sous-ensemble à sélectionner (N' par exemple) est prédéfinie et l'algorithme de sélection cherche à trouver le meilleur sous-ensemble de cette taille.
La deuxième stratégie consiste à sélectionner le plus petit sous-ensemble dont la performance est plus grande ou égale à un seuil prédéfini.
La troisième stratégie cherche à trouver un compromis entre l'amélioration de la performance (l'erreur de classification par exemple) et la réduction de la taille du sous ensemble. Le but est de sélectionner le sous-ensemble qui optimise les deux objectifs en même temps.

2.1.2 La pertinence d'une caractéristique

La performance d'un algorithme d'apprentissage dépend fortement des caractéristiques utilisées dans la tâche d'apprentissage. La présence de caractéristiques redondantes ou non pertinentes peut réduire cette performance. Dans la littérature, il existe plusieurs définitions de la pertinence d'une caractéristique, la plus connue est celle de (John *et al.* [1994], John [1997]). Selon cette définition, une caractéristique est classée comme étant très pertinente, peu pertinente et non pertinente.

Très pertinente : Une caractéristique f_i est dite très pertinente si son absence entraîne une détérioration significative de la performance du système de classification utilisé.

Peu pertinente : Une caractéristique f_i est dite peu pertinente si elle n'est pas "très pertinente" et s'il existe un sous-ensemble V tel que la performance de $V \cup \{f_i\}$ soit significativement meilleure que la performance de V.

Non pertinente : Les caractéristiques qui ne sont ni "peu pertinentes" ni "très pertinentes" représentent les caractéristiques non pertinentes. Ces caractéristiques seront en général supprimées de l'ensemble de caractéristiques de départ.

2.1.3 Caractéristiques générales des méthodes de sélection

Une méthode de sélection qui cherche à maximiser la fonction (Ev) de l'équation (2.1) passe généralement par quatre étapes (Liu et Yu [2005]) :
Les deux premières consistent à initialiser le point de départ à partir duquel la recherche va commencer et à définir une procédure de recherche ou une procédure de génération de sous-ensemble de caractéristiques. Une fois la stratégie de recherche définie, et les sous-ensembles générés, une méthode d'évaluation est définie dans la troisième étape. Les étapes deux et trois se répètent jusqu'à un critère d'arrêt. Ce test d'arrêt représente la quatrième étape de la méthode. Les quatre étapes sont détaillées dans les sections suivantes.

2.1.3.1 Initialisation et procédures de recherche

La première question que l'on peut se poser avant d'appliquer la procédure de recherche est : "Sur quel point de l'espace de caractéristiques la recherche peut-elle commencer ?"
Pour répondre à cette question, il est nécessaire de définir un point de départ (ou direction de recherche). Par exemple, une recherche peut commencer par une ensemble vide de caractéristiques, et continuer par l'ajout successif, à chaque itération, d'une ou plusieurs caractéristiques. Inversement, la recherche peut commencer avec l'ensemble de toutes les caractéristiques et continuer par la suppression séquentielle, à chaque itération, de la caractéristique la moins pertinente. Une autre façon est de commencer la recherche par un sous-ensemble quelconque de caractéristiques.
Une fois que le point de départ est bien choisi, une procédure de recherche (également connue sous le nom *"organisation de la recherche"*, servant à générer des sous-ensembles

de caractéristiques doit être définie. En général, les stratégies de recherche peuvent être classées en trois catégories : exhaustive, heuristique et aléatoire.

a) Génération exhaustive

Dans cette approche, une recherche exhaustive sur tous les sous-ensembles de caractéristiques est effectuée afin de sélectionner le *"meilleur"* sous-ensemble de caractéristiques. Cette stratégie de recherche garantit de trouver le sous-ensemble optimal. Le problème majeur de cette approche est que le nombre de combinaisons croît exponentiellement en fonction du nombre de caractéristiques. Pour un ensemble de N caractéristiques, et quand N devient grand, les 2^N combinaisons possibles rendent la recherche exhaustive impossible (problème NP-complet, Blum et Rivest [1993]).

b) Génération heuristique

Dans cette catégorie, une approche heuristique pour guider la recherche est utilisée. Les algorithmes qui utilisent cette approche sont généralement des algorithmes itératifs dont chaque itération permet de sélectionner ou de rejeter une ou plusieurs caractéristiques. Les avantages de ces algorithmes sont leur simplicité et leur rapidité. En revanche, ils ne permettent pas de parcourir totalement l'espace de recherche. Dans la littérature, les trois sous-catégories les plus connues de cette approche sont :

Forward : cette approche est également appelée ascendante, son principe est de commencer avec un ensemble de caractéristiques vide et à chaque itération une ou plusieurs caractéristiques seront ajoutées.

Backward : cette approche procède à l'inverse de "*Forward*". L'ensemble de départ représente l'ensemble total des caractéristiques et à chaque itération, une ou plusieurs caractéristiques seront supprimées. Cette approche est aussi qualifiée de descendante.

Stepwise : cette approche est un mélange des deux précédentes et consiste à ajouter ou supprimer des caractéristiques au sous-ensemble courant.

c) Génération aléatoire

Pour un ensemble de données et une initialisation particulière, une stratégie de recherche heuristique retourne toujours le même sous-ensemble, ce qui la rend très sensible au changement de l'ensemble de données. La procédure de recherche aléatoire (appelée aussi *stochastique* ou *non-déterministe*) consiste à générer aléatoirement un nombre fini de sous-ensembles de caractéristiques afin de sélectionner le meilleur. En outre, les stratégies de recherche aléatoires convergent en général rapidement vers une solution *"semi-optimale"*, ce qui est préférable pour éviter le phénomène de sur-apprentissage.

2.1.3.2 Procédures d'évaluation

Les méthodes utilisées pour évaluer un sous-ensemble de caractéristiques dans les algorithmes de sélection peuvent être classées en trois catégories principales : *"filter", "wrapper"* et *"embedded"*.

2.1.3.2.a Filter

Le modèle *"filter"* a été le premier utilisé pour la sélection de caractéristiques. Dans celui-ci, le critère d'évaluation utilisé évalue la pertinence d'une caractéristique selon des mesures qui reposent sur les propriétés des données d'apprentissage. Cette méthode est considérée, davantage comme une étape de pré-traitement (filtrage) avant la phase d'apprentissage. En d'autres termes, l'évaluation se fait généralement indépendamment d'un classificateur (John *et al.* [1994]). Les méthodes qui se basent sur ce modèle pour l'évaluation des caractéristiques, utilisent souvent une approche heuristique comme stratégie de recherche. La procédure du modèle *"filter"* est illustrée par la figure (2.2).

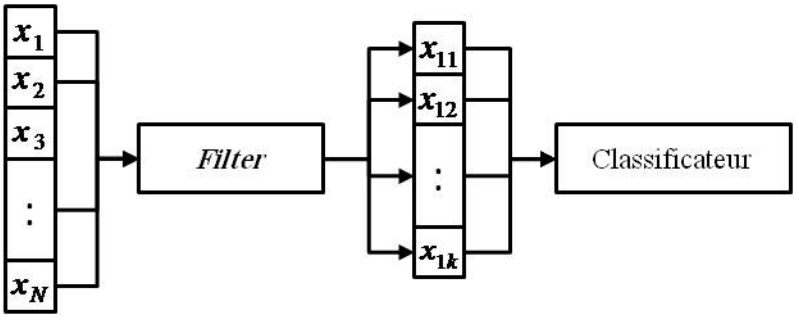

Figure 2.2 – La procédure du modèle *"filter"*

Les caractéristiques sont généralement évaluées par des mesures calculées pour chacune des caractéristiques.
Soit $X = \{x_k | x_k = (x_{k1}, x_{k1}, ..., x_{kn}), k = 1, 2, ..., m\}$ un ensemble de m exemples d'apprentissage dans un espace de représentation comportant n caractéristiques. Soit $Y = \{y_k, k = 1, 2, ..., m\}$ où y_k représente l'étiquette de la classe de l'exemple x_k. Si $x^i = (x_{1i}, x_{2i}, ..., x_{mi})$ représente la $i^{\text{ème}}$ caractéristique $(i = 1, 2, ..., n)$ alors le but d'une méthode d'évaluation *"filter"* est de calculer un score pour évaluer le degré de pertinence de chacune des caractéristiques (x^i). Ci-dessous, nous présentons quelques mesures utilisées dans la littérature comme score ou critère d'évaluation (Guyon et Elisseeff [2003]) :

Le critère de corrélation : ce score est utilisé dans le cas d'une classification binaire $y_k \in \{-1, 1\}$. Il est estimé comme suit :

$$C(i) = \frac{\sum_{k=1}^{m}(x_{ki} - \mu_i)(y_k - \mu_y)}{\sqrt{\sum_{k=1}^{m}(x_{ki} - \mu_i)^2 \sum_{k=1}^{m}(y_k - \mu_y)^2}} \tag{2.2}$$

où μ_i et μ_y représentent respectivement les valeurs moyennes de la $i^{\text{ème}}$ caractéristiques et des étiquettes de l'ensemble d'apprentissage, $||.||$ est la norme euclidienne usuelle.
Cette fonction calcule le cosinus de l'angle entre chacune des caractéristiques et le vecteur des étiquettes. En d'autres termes, et pour une caractéristique donnée, une grande valeur absolue de cette mesure indique sa forte corrélation linéaire avec le vecteur des étiquettes (Y).

Le critère de Fisher : permet de mesurer le degré de séparabilité des classes à l'aide d'une caractéristique donnée (Duda *et al.* [2000], Furey *et al.* [2000]). Il est défini par :

$$F(i) = \frac{\sum_{c=1}^{C} n_c(\mu_c^i - \mu^i)^2}{\sum_{c=1}^{C} n_c(\sigma_c^i)^2} \tag{2.3}$$

où n_c, μ_c^i et σ_c^i représentent respectivement l'effectif, la moyenne et l'écart type de la $i^{\text{ème}}$ caractéristique au sein de la classe c. μ^i est la moyenne globale de la *ième* caractéristique. On pourrait dire que la mesure est liée à la variance interclasse de la caractéristique.

L'information mutuelle est une mesure de dépendance entre les distributions de deux populations (Fraser et Swinney [1986]). Soient X et Y deux variables aléatoires dont les instances sont respectivement les valeurs de la *ième* caractéristique et les étiquettes des classes. L'information mutuelle $I(i)$ est définie comme la divergence de Kullback-Leibler (KL)(Cover et Thomas [1991]) entre la probabilité $P(x^i, y)$ et le produit des probabilités $(P(x^i)P(y))$. L'information mutuelle est estimée empiriquement par :

$$I(i) = \sum_{x_i}\sum_{y} P(X = x_i, Y = y) log \frac{P(X = x_i, Y = y)}{P(X = x_i)P(Y = y)} \tag{2.4}$$

où les probabilités $P(x_i)$, $P(y)$ et $P(x_i, y)$ sont estimées par les fréquences des différentes valeurs possibles.

SNR(**S**ignal-to-**N**oise **R**atio coefficient) est un score qui mesure le pouvoir de discrimination d'une caractéristique entre deux classes. D'une manière similaire au critère de Fisher, cette méthode classe les caractéristiques en calculant le rapport de la valeur absolue de la différence des moyennes des classes et de la moyenne des écart-types des classes. La formule de SNR pour une caractéristique et pour un problème à deux classes est calculée par :

$$SNR(i) = \frac{2 \times |\mu_{C_{i1}} - \mu_{C_{i2}}|}{(\sigma_{C_{i1}} + \sigma_{C_{i2}})} \tag{2.5}$$

D'autres critères d'évaluation sont proposés dans (Golub *et al.* [1999], Tusher *et al.* [2001], Hastie *et al.* [2001]).
Le principal avantage des méthodes de filtrage est leur efficacité calculatoire et leur robustesse face au sur-apprentissage. Malheureusement, ces méthodes ne tiennent pas compte des interactions entre caractéristiques et tendent à sélectionner des caractéristiques comportant de l'information redondante plutôt que complémentaire (Guyon et Elisseeff [2003]). De plus, ces méthodes ne tiennent absolument pas compte de la performance des méthodes

de classification qui suivent la sélection (Kohavi et John [1997]).

2.1.3.2.b Wrapper

Le principal inconvénient des approches *"filter"* est le fait qu'elles ignorent l'influence des caractéristiques sélectionnées sur la performance du classificateur à utiliser par la suite. Pour résoudre ce problème, Kohavi et John ont introduit le concept *"wrapper"* pour la sélection de caractéristiques (Kohavi et John [1997]). Les méthodes *"wrapper"*, appelées aussi méthodes enveloppantes, évaluent un sous-ensemble de caractéristiques par sa performance de classification en utilisant un algorithme d'apprentissage. La procédure du modèle *"wrapper"* est illustrée par la figure 2.3.

L'évaluation se fait à l'aide d'un classificateur qui estime la pertinence d'un sous-ensemble

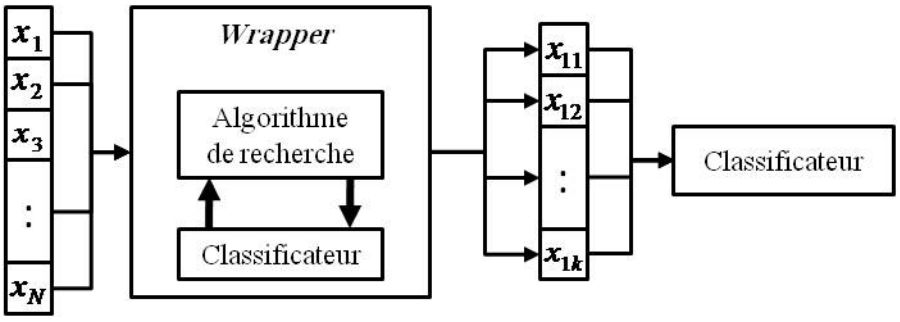

Figure 2.3 – La procédure du modèle *"wrapper"*

donné de caractéristiques. Les sous-ensembles de caractéristiques sélectionnés par cette méthode sont bien adaptés à l'algorithme de classification utilisé, mais ils ne sont pas forcément valides si on change le classificateur. La complexité de l'algorithme d'apprentissage rend les méthodes *"wrapper"* très coûteuses en temps de calcul. En général, pour diminuer le temps de calcul et pour éviter les problèmes de sur-apprentissage, le mécanisme de validation croisée est fréquemment utilisé. Une meilleure performance des méthodes *"wrapper"* par rapport à certaines méthodes de filtrage a été démontré par (Kohavi et John [1997]).
Le problème de la complexité de cette technique rend impossible l'utilisation d'une stratégie de recherche exhaustive (problème NP-complet). Par conséquent, des méthodes de recherche heuristiques ou aléatoires peuvent être utilisées. La recherche devient néanmoins, de plus en plus irréalisable avec l'augmentation de la taille de l'ensemble initial de caractéristiques.
Les méthodes *"wrapper"* sont généralement considérées comme étant meilleures que celles de filtrage selon (Li et Guo [2008], Huang *et al.* [2008]). Elles sont capables de sélectionner des sous-ensembles de caractéristiques de petite taille qui sont performants pour le classificateur utilisé mais il existe deux inconvénients principaux qui limitent ces méthodes :

a) La complexité et le temps de calcul nécessaire pour la sélection.

Le principal inconvénient de l'approche *"wrapper"* est le temps nécessaire pour la sélection des caractéristiques, il est nettement plus long que celui des approches de filtrage et d'autres

approches de sélection de caractéristiques. L'utilisation d'un classificateur pour évaluer les sous-ensembles ainsi que les techniques d'évaluation (validation croisée par exemple) rendent les méthodes *"wrapper"* très coûteuses en terme de temps de calcul.

b) La dépendance des caractéristiques pertinentes sélectionnées par rapport au classificateur utilisé.
La deuxième limitation de l'approche *"wrapper"* est que l'évaluation des caractéristiques se fait par un seul classificateur lors de la sélection. Chaque classificateur a ses spécificités et ses hypothèses. Ainsi le sous-ensemble sélectionné dépend toujours du classificateur utilisé.

2.1.3.2.c Embedded

A la différence des méthodes *"wrapper"* et *"filter"*, les méthodes *"embedded"* (appelées aussi méthodes intégrées) incorporent la sélection de variables lors du processus d'apprentissage. Un tel mécanisme intégré pour la sélection des caractéristiques peut être trouvé, par exemple, dans les algorithmes de type SVM (§ 3.1.3), AdaBoost (§ 3.2.2.2.a), ou dans les arbres de décisions (§ 3.1.2). Dans les méthodes de sélection de type *"wrapper"*, la base d'apprentissage est divisée en deux parties : une base d'apprentissage et une base de validation pour valider le sous-ensemble de caractéristiques sélectionné. En revanche, les méthodes intégrées peuvent se servir de tous les exemples d'apprentissage pour établir le système. Cela constitue un avantage qui peut améliorer les résultats. Un autre avantage de ces méthodes est leur plus grande rapidité par rapport aux approches *"Wrapper"* parce qu'elles évitent que le classificateur recommence de zéro pour chaque sous-ensemble de caractéristiques.

2.1.4 Critère d'arrêt

Certains critères doivent être définis pour arrêter le processus de recherche sur les sous-ensembles de caractéristiques. Pour les méthodes de filtrage, le critère d'arrêt couramment utilisé est basé sur l'ordre des caractéristiques, rangées selon certains scores de pertinence (généralement des mesures statistiques). Une fois les caractéristiques ordonnées, celles qui ont les scores les plus élevés seront choisies et utilisées par un classificateur. Pour les méthodes de type *"wrapper"*, le processus de recherche peut s'arrêter lorsque il n'y a plus d'amélioration de précision. En d'autres termes, lorsqu'il n'y a plus la possibilité de trouver un sous-ensemble meilleur que le sous-ensemble actuel. Un critère d'arrêt pour les méthodes enveloppantes est de continuer à rechercher jusqu'au moment où la précision dépasse un certain seuil défini par l'utilisateur.

2.1.5 Revue de quelques méthodes de sélection

Dans cette section, nous présentons quelques méthodes de sélection de caractéristiques de la littérature. Nous avons choisi de présenter de méthodes fondées sur les différentes techniques de recherche définies précédemment ainsi que différentes techniques d'évaluation.

2.1.5.1 SFS et SBS

SFS (**S**equential **F**orward **S**election) ou (sélection séquentielle croissante) est la première méthode proposée pour la sélection de caractéristiques. Cette méthode a été proposée en 1963 par Marill et Green (Marill et Green [1963]). Une approche heuristique de recherche est utilisée dans cette méthode, en commençant par un ensemble vide de caractéristiques. A chaque itération, la meilleure caractéristique parmi celles qui restent sera sélectionnée, supprimée de l'ensemble de départ et ajoutée au sous-ensemble des caractéristiques sélectionnées (Algorithme 2.1). Le processus de sélection continue jusqu'à un critère d'arrêt.
En 1971, Whitney (Whitney [1971]) a proposé une méthode similaire au SFS appelée SBS (**S**equential **B**ackward **S**election) ou (sélection séquentielle arrière). A la différence de la méthode SFS, cette méthode commence par l'ensemble de toutes les caractéristiques et à chaque itération, la caractéristique la plus mauvaise sera supprimée (Algorithme 2.2).
Bien que les deux méthodes SFS et SBS semblent similaires, Ahan et Bankert (Aha et Bankert [1995]) ont montré que la méthode SBS est plus performante parce qu'elle prend en considération l'interaction d'une caractéristique avec un ensemble de caractéristiques plus large, contrairement au SFS qui ne prend en considération que l'interaction de cette caractéristique avec le sous-ensemble déjà sélectionné. Par ailleurs, l'évaluation des sous-ensembles de grande taille avec la méthode SBS pose un problème au niveau de temps de calcul.

Algo 2.1 Algorithme *SFS*

Entrées:
$F = \{f_1, f_2, .., f_N\}$
M : taille de l'ensemble final
Sorties: $E = \{f_{s1}, f_{s2}, .., f_{sM}\}$
$E = \varnothing$
Pour $i = 1$ à M **Faire**
Pour $j = 1$ à $|F|$ **Faire**
Évaluer $f_j \cup E$
Fin Pour
f_{max} = meilleure f_j
$E = E \cup f_{max}$, $F = F \setminus f_{max}$
Fin Pour
Retourner E

Algo 2.2 Algorithme *SBS*

Entrées:
$F = \{f_1, f_2, .., f_N\}$
M : taille de l'ensemble final
Sorties: $E = \{f_{s1}, f_{s2}, .., f_{sM}\}$
$E = F$
Pour $i = 1$ à N-M **Faire**
Pour $j = 1$ à $|E|$ **Faire**
Évaluer $E \setminus f_j$
Fin Pour
f_{min} = la plus mauvaise f_j
$E = E \setminus f_{min}$
Fin Pour
Retourner E

En 1978, des généralisations des méthodes SBS et SFS appelées GSFS et GSBS, sont proposées par Kittler (Kittler [1978]). Dans ces méthodes, l'auteur propose, au lieu d'inclure (ou exclure) une caractéristique à chaque itération, d'inclure (ou exclure) un sous ensemble de caractéristiques. Ces algorithmes ont montré une meilleure performance par rapport aux méthodes initiales, mais ils conservent toujours les mêmes problèmes que les méthodes de base.

Deux autres méthodes de la famille (FS, BS) qui limitent les inconvénients des méthodes décrites ci-dessous, appelées SFFS (**S**equential **F**loating **F**orward **S**election) et SFBS (**S**equential **F**loating **B**ackward **S**election) sont proposées en 1994 par Pudil et al (Pudil *et al.* [1994]). Ces méthodes consistent à utiliser l fois l'algorithme SFS de manière à ajouter l variables, puis à utiliser r fois l'algorithme SBS afin d'en supprimer r. Ces étapes sont alors réitérées jusqu'à l'obtention du critère d'arrêt. La dimension du sous-ensemble à chaque étape sera alors dépendante des valeurs de l et r. Les valeurs optimales de ces paramètres ne pouvant pas être déterminées théoriquement, les auteurs proposent de les laisser flottantes au cours du processus de sélection afin de se rapprocher au maximum de la solution optimale.

2.1.5.2 Branch and Bound

Ce type de méthode est lié à la modélisation du problème de recherche du meilleur sous-ensemble sous forme de graphe. Alors les algorithmes développés sur les graphes sont applicables, par exemple la methode *"Branch and Bound"*. La méthode *"Branch and Bound"* (BB) consiste à énumérer un ensemble de solutions d'une manière intelligente en ce sens que, en utilisant certaines propriétés du problème en question, cette technique arrive à éliminer des solutions partielles qui ne mènent pas à la solution que l'on recherche. Pour ce faire, cette méthode se dote d'une fonction qui permet de mettre une borne sur certaines solutions pour soit les exclure, soit les maintenir comme des solutions potentielles. Bien entendu, la performance de cette méthode dépend de la qualité de cette fonction d'évaluation partielle. Cette technique a été appliquée pour résoudre des problèmes de sélection de caractéristiques en 1977 par Narendra et Fukunaga (Narendra et Fukunaga [1977]). Son principe est de construire un arbre de recherche où la racine représente l'ensemble des caractéristiques et les autres nœuds représentent des sous-ensembles de caractéristiques. En parcourant l'arbre de la racine jusqu'aux feuilles, l'algorithme enlève successivement la plus mauvaise caractéristique du sous ensemble courant (nœud courant) qui ne satisfait pas le critère de sélection. Une fois que la valeur attribuée à un nœud est plus petite qu'un seuil (bound), les sous-arbres de ce nœud sont supprimés. Cette technique garantit de trouver un sous-ensemble optimal de caractéristiques à condition d'utiliser une fonction d'évaluation monotone. L'inconvénient de cette méthode est son temps de calcul qui croît vite avec l'augmentation du nombre de caractéristiques et qui devient impraticable à partir d'un certain nombre (30 caractéristiques). Une amélioration de cette méthode en utilisant d'autres techniques de recherche dans l'arbre afin d'accélérer le processus de sélection a été proposée dans (Chen [2003], Somol *et al.* [2004]).

2.1.5.3 FOCUS

Un algorithme de filtrage pour la sélection de caractéristiques, appelé FOCUS, a été proposé par Almuallim et Dietterich en 1991 (Almuallim et Dietterich [1991]). Cette méthode repose sur une recherche exhaustive sur l'ensemble initial de caractéristiques pour trouver le sous-

ensemble le plus performant de taille minimale. L'algorithme FOCUS (algorithme 2.3) commence par générer et évaluer tous les sous-ensembles de taille T (initialement un), puis tous les couples de caractéristiques, les triplets et ainsi de suite jusqu'à ce que le critère d'arrêt soit satisfait.

Algo 2.3 Algorithme de sélection *FOCUS*

Entrées: Une base d'apprentissage $A = \{X_1, X_2, .., X_M\}$ où $X_i = \{x_{i1}, x_{i2}, .., x_{iN}\}$
T : Taille maximale de l'ensemble final et un seuil ϵ
Sorties: S : ensemble final des caractéristiques
$S = \varnothing$
Pour $i = 1$ à T **Faire**
Pour chaque sous-ensemble (S_1) de taille (i) **Faire**
$Cons = Inconsistance(A, S_1)$
Si $Cons < \epsilon$ **alors**
$S = S_1$
Retourner S
Fin Si
Fin Pour
Fin Pour

Les inconvénients de cette approche sont d'un côté la sensibilité de sa méthode d'évaluation au bruit et de l'autre côté son temps de calcul qui devient énorme avec l'augmentation de la taille de l'ensemble des caractéristiques et du nombre d'exemples de la base. Une année plus tard, les mêmes auteurs ont proposé FOCUS2 comme une amélioration de leur méthode initiale (Almuallim et Dietterich [1992]). FOCUS2 est beaucoup plus rapide que FOCUS, mais elle est toujours sensible au bruit.

2.1.5.4 Relief

Une des méthodes de filtrage les plus connues pour la sélection de caractéristiques est la méthode relief. Cette méthode fut proposée en 1992 par Kira et Rendell (Kira et Rendell [1992]). Son principe est de calculer une mesure globale de la pertinence des caractéristiques en accumulant la différence des distances entre des exemples d'apprentissage choisis aléatoirement et leurs plus proches voisins de la même classe et de l'autre classe. L'algorithme 2.4 montre le pseudo code de cette méthode. La simplicité, la facilité de la mise en œuvre ainsi que la précision même sur des données bruitées, représentent les avantages de cette méthode. En revanche, sa technique aléatoire ne peut pas garantir la cohérence des résultats lorsqu'on applique plusieurs fois la méthode sur les mêmes données. Par ailleurs, cette méthode ne prend pas en compte la corrélation éventuelle entre les caractéristiques. Afin d'éviter le caractère aléatoire de l'algorithme, John et al. (John *et al.* [1994]) ont proposé une version déterministe appelée ReliefD. D'autres variantes de cet algorithme, pour

améliorer sa performance, sa vitesse ou les deux, ont été proposées dans (Koller et Sahami [1996], Liu *et al.* [2002]).

Algo 2.4 Algorithme de sélection de *Relief*

Entrées: Une base d'apprentissage $A = \{X_1, X_2, .., X_M\}$ où chaque exemple $X_i = \{x_{i1}, x_{i2}, .., x_{iN}\}$
Nombre d'itérations T

Sorties: W[N] : vecteur de poids des caractéristiques (f_i), $-1 \leq W[i] \leq 1$

$\forall i, W[i] = 0$;
Pour $t = 1$ à T **Faire**
Choisir aléatoirement un exemple X_k
Chercher deux plus proches voisins (un dans sa classe (X_a) et un deuxième dans l'autre classe (X_b))
Pour $i = 1$ à N **Faire**
$W[i] = W[i] + \frac{|x_{ki} - x_{bi}|}{M \times T} - \frac{|x_{ki} - x_{ai}|}{M \times T}$
Fin Pour
Fin Pour
Retourner W

2.1.5.5 LVW et LVF

LVW (**L**as **V**egas **W**rapper) est une méthode de sélection de caractéristiques proposée en 1996 par Liu et Setiono (Liu et Setiono [1996]). Cette méthode consiste à générer aléatoirement et à chaque itération, un sous-ensemble de caractéristiques et à l'évaluer avec un classificateur.

Algo 2.5 Algorithme *LVW*

Entrées: Une base d'apprentissage A
Une base de caractéristiques S
Nombre d'itérations T

Sorties: S : Ensemble sélectionné

$Err = Classificateur(A, S)$
$k = 0$, $N = |S|$
Répéter
$S_1 = Générer\_Al()$,$N_1 = |S_1|$
$Err_1 = Classificateur(A, S1)$
Si $(Err_1 < Err) ou (Err = Err_1$ et $N1 < N)$ **alors**
$k = 0, N = N_1, S = S_1, Err = Err_1$
Fin Si
$k = k + 1$
Jusqu'à k=T
Retourner S

Algo 2.6 Algorithme *LVF*

Entrées: Une base d'apprentissage A
Une base de caractéristiques S
Nombre d'itérations T et un seuil ϵ

Sorties: S : Ensemble sélectionné

$N = |S|$
Pour i=1 à T **Faire**
$S_1 = Générer\_Al()$
$N_1 = |S_1|$
Si $Inconsistance(A, S_1) < \epsilon$ et $(N_1 < N$ **alors**
$N = N_1$
$S = S_1$
Fin Si
Fin Pour

Retourner S

Après avoir évalué, si sa performance est meilleure que la meilleure performance trouvée auparavant (au départ, l'ensemble de base est supposé comme le meilleur sous-ensemble), ce sous-ensemble devient le meilleur sous-ensemble courant. Ce processus est répété jusqu'à ce que T essais consécutifs soient infructueux pour l'amélioration. L'algorithme 2.5 résume le pseudo-code de cette méthode. Cette méthode présente l'inconvénient de ne pas garantir l'optimalité de la solution finale ainsi qu'un temps de calcul très élevé.

LVF (**L**as **V**egas **F**ilter) est une méthode de filtrage pour la sélection de caractéristiques, proposée deux ans plus tard par les mêmes auteurs (Liu et Setiono [1998]). Elle est similaire à la méthode LVW mais l'évaluation des sous-ensembles se fait par le calcul d'une mesure appelée "taux d'incohérence" ou "taux d'inconsistance". L'inconsistance pour un sous-ensemble de caractéristiques est définie par le rapport entre le nombre d'exemples inconsistants de la base de données et le nombre total d'exemples. Un exemple est dit inconsistant s'il existe un autre exemple qui a la même représentation dans l'espace des caractéristiques du sous-ensemble de caractéristiques étudié (appelé exemple équivalent), mais qui appartient à une autre classe. Dans la suite, nous illustrons par un exemple représentatif qui montre comment calculer l'inconsistance d'un sous ensemble de caractéristiques. Soit un problème de classification à deux classes (1 et 0) avec $A = \{X_0, X_1, .., X_{12}\}$ une base d'apprentissage composée de treize exemples, six exemples dans la première classe (de X_0 jusqu'à X_5) et sept dans l'autre (X_6 jusqu'à X_{12}). Chaque exemple X_i est représenté par six caractéristiques binaires. Le tableau 2.1 résume la base d'apprentissage. Si

Exemples	x_1	x_2	x_3	x_4	x_5	x_6	Classe
X_0	0	1	1	1	0	0	1
X_1	1	0	1	1	0	0	1
X_2	1	1	0	0	0	0	1
X_3	1	1	0	1	0	1	1
X_4	1	1	1	0	1	0	1
X_5	0	0	1	1	0	0	1
X_6	0	0	0	1	1	1	0
X_7	0	0	1	0	0	1	0
X_8	1	0	0	0	1	1	0
X_9	1	0	1	0	0	1	0
X_{10}	0	1	0	0	0	1	0
X_{11}	0	1	0	1	1	1	0
X_{12}	0	1	1	0	1	0	0

Table 2.1 – Exemple d'une base d'apprentissage

la méthode LVF choisit aléatoirement le sous-ensemble $S1 = \{x_1, x_2, x_3\}$ alors il y aura trois exemples inconsistants de la première classe et trois de la deuxième (par exemple $X_0 = X_{12} = \{0, 1, 1\}$ mais ils ne sont pas dans la même classe et donc ils sont inconsistants). Le taux d'inconsistance de $S1$ est égal à $\frac{3+3}{13}$. Par contre le taux d'inconsistance pour le sous-ensemble $S1 = \{x_1, x_4, x_6\}$ est nul et c'est donc le meilleur sous-ensemble à trouver.

Cette méthode présente les mêmes inconvénients que la méthode *FOCUS*. Elle est donc

très sensible au bruit et comme toutes les méthodes de recherche exhaustive, elle est très coûteuse en temps de calcul.

2.1.5.6 SAC

SAC (**S**élection **A**daptative de **C**aractéristiques) est une méthode de sélection de descripteurs proposée par Kachouri et al. en 2010 (Kachouri *et al.* [2010]). Cette méthode, développée dans le cadre d'un ensemble de descripteurs à plusieurs dimensions, peut être adaptée pour une sélection de caractéristiques. L'idée générale de la méthode est de construire un ensemble de classificateurs SVM appris sur chacun des descripteurs et de sélectionner les meilleurs par discrimination linéaire de Fisher (FLD). Ils proposent de considérer la performance d'apprentissage des modèles correspondant à ces descripteurs pour l'identification d'une meilleure discrimination de Fisher. L'algorithme 2.7 donne le pseudo code de cette méthode.

Algo 2.7 Algorithme de sélection de *SAC*

Entrées: Une base d'apprentissage $A = \{X_1, X_2, .., X_M\}$ où chaque exemple $X_k = \{desc_{k1}, desc_{k2}, .., desc_{kN}\}$, $k = 1..m$ et $X^i = \{desc_{1i}, desc_{2i}, .., desc_{Mi}\}$, $i = 1..N$
Sorties: M_s : les classificateurs retenus

Pour $i = 1$ à N **Faire**
$M_i = Apprentissage\ SVM(X^i)$
$Pr(M_i) =$ taux de classification en utilisant le modèle M_i
Fin Pour
$\mathfrak{L} =$ Trier $(Pr(M_i))$ par ordre décroissant $\forall i \in \{1, 2, .., N\}$
$k = FLD(\mathfrak{L})$
Retourner $M_s = (M_{s1}, M_{s2}, .., M_{sk})$

Après avoir construit la base de modèles d'apprentissage $M = \{M_1, M_2, ..., M_N\}$ où N représente le nombre total de descripteurs et M_i est le modèle construit sur le $i^{\text{ème}}$ descripteur en utilisant un classificateur SVM, les auteurs proposent une suite $\mathfrak{L}$ qui représente la performance des modèles M_i triés par ordre décroissant sur lequel le score de Fisher sera calculé ($\mathfrak{L} = \{Pr(M_{s1}), Pr(M_{s2}), .., Pr(M_{sN})\}$). Pour calculer ce score, ils proposent de calculer deux valeurs moyennes $m_1(i)$ et $m_2(i)$ (équation 2.6) avec $i = 1..N$, qui représentent les deux moyennes de performances d'apprentissage qui ont une valeur respectivement plus grande (plus petite) que la performance du modèle M_i ($Pr(M_{si})$).

$$m_1(i) = \frac{1}{i}\sum_{j=1}^{i} Pr(M_{sj}),\ m_2(i) = \frac{1}{N-i}\sum_{j=i+1}^{N} Pr(M_{sj}) \tag{2.6}$$

En fonction de ces deux moyennes, deux variances sont calculées (équation 2.7).

$$v_1^2(i) = \frac{1}{i}\sum_{j=1}^{i} |Pr(M_{sj}) - m_1(i)|^2,\ v_2^2(i) = \frac{1}{N-i}\sum_{j=i+1}^{N} |Pr(M_{sj}) - m_2(i)|^2 \tag{2.7}$$

Finalement le sous-ensemble sélectionné est celui qui maximise le discriminant de Fisher ($P(i)$) calculé en fonction de $m_1(i)$, $m_2(i)$, $v_1^2(i)$ et $v_2^2(i)$. $P(i)$ est calculé comme suit :

$$P(i) = \frac{|m_1(i) - m_2(i)|}{v_1^2(i) + v_2^2(i)} \tag{2.8}$$

2.1.5.7 Max-relevance, Min-Redundancy (mRMR)

"Max-relevance, Min-Redundancy" (mRMR) est une méthode de filtrage pour la sélection de caractéristiques proposée par Peng et al. en 2005 (Peng *et al.* [2005]). Cette méthode est basée sur des mesures statistiques classiques comme l'information mutuelle, la corrélation etc. (§ 2.2). L'idée de base est de profiter de ces mesures pour essayer de minimiser la redondance (**mR**) entre les caractéristiques et de maximiser la pertinence (**MR**). Les auteurs proposent deux variantes de leur méthode. Une pour des données discrètes et l'autre pour des données continues.
Pour les données discrètes, les auteurs utilisent l'information mutuelle pour calculer les deux facteurs **mR** et **MR**. Le calcul de la redondance et de la pertinence d'une caractéristique est donné par l'équation 2.10.

$$Redondance(i) = \frac{1}{|F|^2}\sum_{i,j\in F} I(i,j),\ Pertinence(i) = \frac{1}{|F|^2}\sum_{i,j\in F} I(i,Y) \tag{2.9}$$

où F et $|F|$ représentent, respectivement, l'ensemble des caractéristiques et sa taille. I(i,j) est l'information mutuelle entre la *ième* et la *jème* caractéristique et finalement I(i,Y) est l'information mutuelle entre la *ième* caractéristique et l'ensemble des étiquettes de classes (Y). Le score d'une caractéristique est la combinaison de ces deux facteurs tel que :

$$Score(i) = \frac{Pertinence(i)}{Redondance(i)} \text{ ou } Score(i) = Pertinence(i) - Redondance(i) \tag{2.10}$$

Pour les données continues, les auteurs ont remplacé l'information mutuelle par d'autres mesures. Pour la redondance ils ont utilisé la mesure de corrélation, par contre, la mesure F-statistique est utilisée pour calculer la pertinence.
Après cette évaluation individuelle des caractéristiques, une technique de recherche avant séquentielle est utilisée avec un classificateur pour sélectionner le sous-ensemble final de caractéristiques. En d'autres termes, un classificateur est utilisé pour évaluer les sous-ensembles en commençant par la caractéristique qui a le meilleur score, les deux meilleures, etc., jusqu'à trouver le sous-ensemble qui minimise l'erreur de classification.

Méthode	Type	Stratégie de recherche	Non élimination de la redondance	Non prise en compte ses interactions	Complexité	Dépendance à la fonction d'évaluation	Sensibilité aux bruits
SFS	Filter	Heuristique	X	X			
SBS	Filter	Heuristique	X		X		
B and B	Filter ou Wrapper	Heuristique			X	X	
Focus	Filter	Exhaustive		X			X
Relief	Filter	Aléatoire	X	X			
LVW	Wrapper	Aléatoire			X	X	
LVF	Filter	Aléatoire	X		X		X
SAC	Hybride	Heuristique	X	X		X	
mRMR	Filter	Heuristique	X	X			
AG	Filter ou Wrapper	Aléatoire			X	X	

Table 2.2 – Résumé des méthodes de sélection présentées

2.1.5.8 Les algorithmes génétiques

Les algorithmes génétiques ont été utilisés dans le domaine de la sélection de caractéristiques afin d'accélérer la recherche et d'éviter les optima locaux. De nombreuses études rapportées dans la littérature ont montré que les méthodes qui utilisent les AGs comme technique de recherche ont donné de meilleurs résultats que les résultats obtenus par les autres méthodes de sélection (Jain et Zongker [1997], Kuncheva et Jain [1999], Ishibuchi et Nakashima [2000]). Le chapitre 4 sera consacré à la présentation d'une description détaillée des algorithmes génétiques ainsi que des méthodes de sélection qui utilisent ces techniques.

Le tableau 2.2 résume les inconvénients de toutes les méthodes de sélection de caractéristiques présentées ci-dessus.

2.2 Réduction basée sur une transformation de données

La réduction de la dimensionnalité par une transformation de données (appelée aussi extraction de caractéristiques) ne se fait pas par une sélection de certaines caractéristiques, mais par une construction de nouvelles caractéristiques obtenues en combinant les caractéristiques initiales. Une transformation de données risque de faire perdre la sémantique de l'ensemble initial de caractéristiques et donc l'utilisation de cette famille de méthodes n'est applicable que dans le cas où la sémantique n'intervient plus dans les étapes qui suivent la réduction.
Les sections suivantes décrivent brièvement plusieurs techniques de réduction connues. Elles sont généralement groupées en deux catégories : les méthodes linéaires et les méthodes non linéaires.

2.2.1 Méthodes linéaires

Nous rappelons brièvement les principes de quelques méthodes classiques d'analyse de données, elles sont le fondement de plusieurs méthodes non linéaires plus récentes.

2.2.1.1 Analyse en Composantes Principales

L'**A**nalyse en **C**omposantes **p**rincipales (ACP) fait partie du groupe des méthodes descriptives multidimensionnelles appelées méthodes factorielles.
L'ACP est une technique qui permet de trouver des espaces de dimensions plus petites dans lesquels il est possible d'observer au mieux les individus. Sa démarche essentielle consiste à transformer les variables quantitatives initiales, plus ou moins corrélées entre elles, en des variables quantitatives, non corrélées, combinaisons linéaires des variables initiales et appelées composantes principales. Les composantes principales sont donc de nouvelles variables

indépendantes, combinaisons linéaires des variables initiales, possédant une variance maximale. Globalement l'ACP consiste à rechercher la direction suivant laquelle le nuage de points des observations s'étire au maximum. A cette direction correspond la première composante principale. La seconde composante principale est déterminée de telle sorte qu'elle soit la plus indépendante possible de la première ; elle est donc perpendiculaire à celle-ci. Ces deux composantes forment le premier plan principal. Cette opération est réitérée de manière à trouver toutes les composantes principales expliquant le maximum de variance. La figure 2.4 montre à gauche, un exemple de données 3D qui se trouvent dans un plan 2D et à droite les deux premières composantes principales sur ces données.

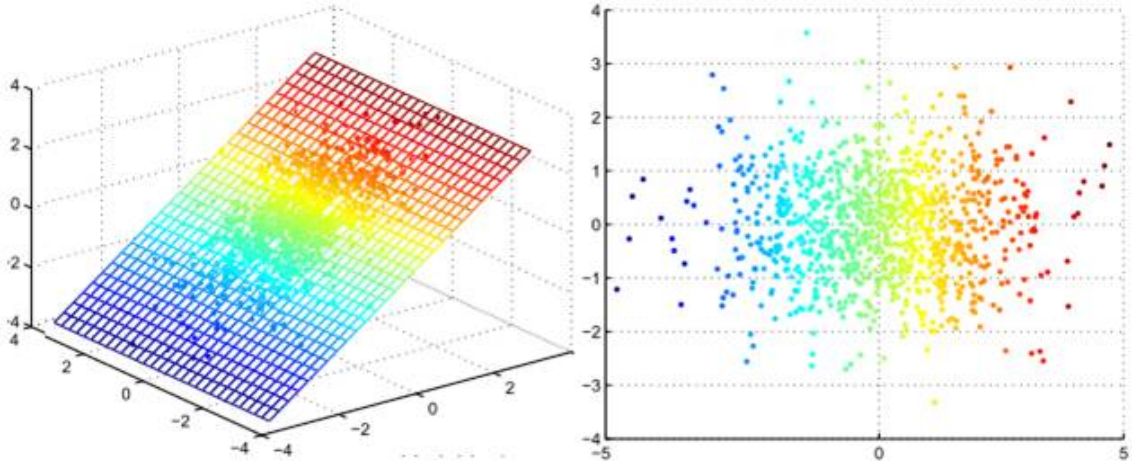

Figure 2.4 – ACP sur des données linéaires

Supposons que nous ayons un ensemble de données $X = \{x_1, x_2, .., x_M\}$ composé de M observations où chaque observation $x_i = (x_{i1}, x_{i2}, .., x_{iN})$ est composée de N caractéristiques. X est associé à une matrice de données A de taille $N \times M$ où chaque colonne représente une caractéristique. En pratique, le calcul de l'ACP pour la matrice X revient à réaliser les opérations ci-dessous afin de trouver les composantes principales :

1. Calculer le vecteur $\mu = (\mu_1, \mu_2, \cdots, \mu_m)^T$ qui représente le vecteur moyen où μ_i est la moyenne de la *ième* composante des données.

2. Calculer la matrice χ en soustrayant le vecteur moyen à toutes les colonnes de A dans le but d'obtenir des données centrées.

3. Calculer la matrice S (de taille $N \times N$) de covariance de χ avec ($S = \chi \cdot \chi^T$).

4. Calculer la matrice U (de taille $N \times N$) qui est composée des coordonnées des vecteurs propres $\vec{u_j}$ de S triés par ordre décroissant des modules des valeurs propres λ_j (la première colonne de U est le vecteur propre qui correspond à la plus grande valeur propre)

5. Garder les R premières colonnes de U pour former la matrice $\tilde{U} : N \times R$ qui représente les R premières composantes principales.

Pour plus de détails, le lecteur pourra consulter (Jolliffe [1986]).
L'ACP étant une méthode de réduction de dimension, il est important de savoir qu'elle ne peut pas retenir la totalité de l'information contenue dans le nuage de points initial. Enfin, l'ACP prend uniquement en compte les dépendances linéaires entre les variables et ne peut donc pas fournir une projection pertinente pour une distribution non-linéaire de points. La figure 2.5 montre à gauche, un exemple de données non-linéaires (non réparties dans un plan) et à droite le résultat de leur projection dans un plan généré par les deux premières composantes principales calculées sur ces données.

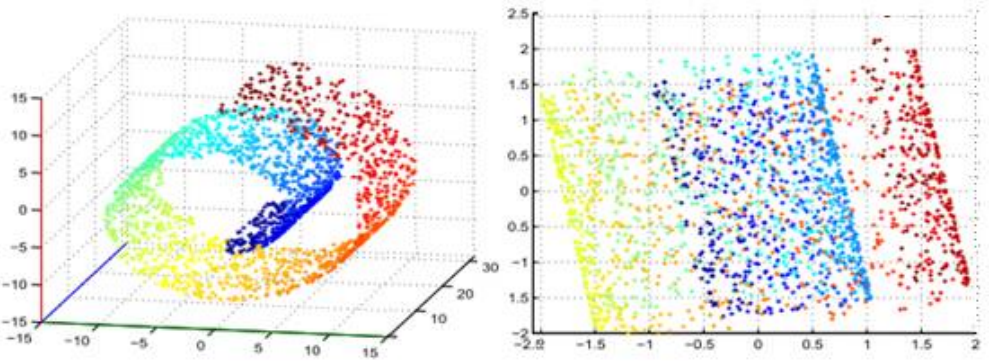

Figure 2.5 – ACP sur des données non-linéaires

2.2.1.2 Analyse Linéaire Discriminante

L'analyse linéaire discriminante, appelée aussi analyse discriminante linéaire de Fisher, est une méthode de réduction du nombre de dimensions proposée par Fisher en 1936 (Fisher [1936]). Cette méthode s'applique lorsque les classes des individus sont connues. L'idée de Fisher a été de créer une méthode pour choisir entre les combinaisons linéaires des variables celles qui maximisent l'homogénéité de chaque classe. En d'autres termes, cette méthode consiste à chercher un espace vectoriel de faible dimension qui maximise la variance inter-classe (pour une description complète de la méthode voir (Lebart et Morineau [1995])).

2.2.1.3 Positionnement Multi-Dimensionnel

La méthode de positionnement multidimensionnel (Messick et Abelson [1956]), MDS (**M**ulti **D**imensional **S**caling) en anglais, permet de construire une représentation en faible dimension des points de l'espace. Son objectif est de construire, à partir d'une matrice de distances ou des mesures de similarité calculées sur chaque paire de points, une représentation euclidienne des individus dans un espace de dimension réduite qui préserve "au mieux" ces distances.
Supposons que nous ayons un ensemble de données $X = \{x_1, x_2, .., x_M\}$ composé de M observations où chaque observation $x_i = (x_{i1}, x_{i2}, .., x_{iN})$ est composée de N caractéristiques. Soit d une matrice symétrique de taille $M \times M$ où chaque élément d_{ij} représente la

distance entre x_i et x_j et $d_{ii} = 0$. L'idée de MDS est de trouver une configuration de points y_i, $i = 1..M$ dans un espace de dimension plus réduite qui conserverait les distances entre les points initiaux x_i. Autrement dit, il cherche les points y_i dans un espace de dimension $q < N$ tels que $d(y_i, y_j) \approx d_{ij} = d(x_i, x_j)$. Ceci revient à optimiser le critère suivant :

$$J = \sum_{i=1}^{N} = \sum_{j=1}^{N} (d_{ij} - d(y_i, y_j))^2 \tag{2.11}$$

L'exemple classique est d'obtenir la carte d'un pays en partant de la connaissance des distances entre chaque paire de villes. Le tableau 2.3 représente la matrice de distances de dix villes françaises.

	Amiens	Caen	Grenoble	Lyon	Metz	Nancy	Nantes	Nice	Paris	Troyes
Amiens	0									
Caen	240	0								
Grenoble	710	790	0							
Lyon	610	690	110	0						
Metz	350	570	560	460	0					
Nancy	360	540	510	400	59	0				
Nantes	530	280	740	630	710	670	0			
Nice	1080	1160	330	470	930	870	1110	0		
Paris	150	240	570	460	330	300	380	930	0	
Troyes	280	400	450	350	230	180	520	820	790	0

Table 2.3 – Matrice de distances kilométriques de 10 villes françaises

La figure 2.6 montre une représentation graphique des données obtenue par l'algorithme appliqué sur les données du tableau 2.3 en passant à un espace de dimension 2. Sur la figure, nous pouvons remarquer que l'approximation des distances ainsi que le positionnement sont très proches de la réalité.

Comme l'ACP, l'algorithme de Positionnement Multi-Dimensionnel n'est valable que sur une distribution linéaire de données.

2.2.2 Méthodes non-linéaires

Les méthodes non linéaires ont pour objectif d'optimiser les représentations afin qu'elles reflètent au mieux la topologie initiale des données.

2.2.2.1 Isomap

Isomap (Tenenbaum *et al.* [2000]) est une technique de réduction de dimensions qui, comme la méthode de positionnement multidimensionnel (MDS), part de la connaissance d'une matrice de dissimilarités entre les paires d'individus. Le but est cette fois de trouver une

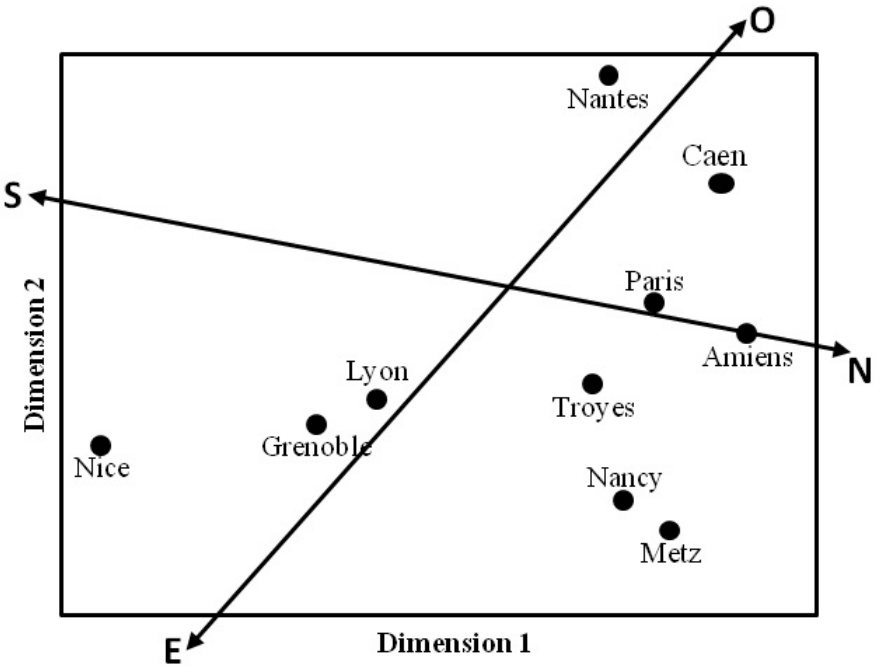

Figure 2.6 – Positionnement de 10 villes françaises à partir de la matrice de leurs distances kilométriques

variété (non linéaire) contenant les données. La figure 2.7 montre à gauche, un exemple des données non-linéaires et à droite le résultat de la représentation de ces points dans l'espace dirigé par les deux premières dimensions trouvées par l'application de l'algorithme isomap sur ces données.

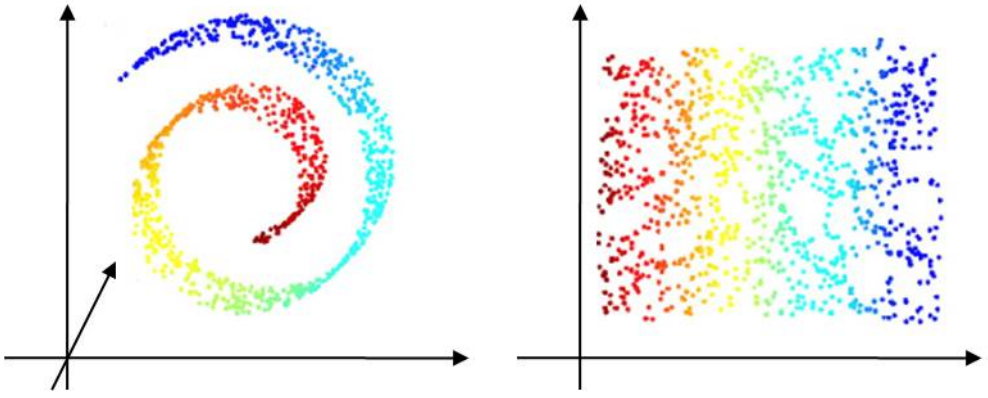

Figure 2.7 – Isomap sur des données non-linéaires

Dans la suite, nous présentons une description des étapes de l'algorithme isomap. Supposons que nous ayons un ensemble de données $X = \{x_1, x_2, .., x_M\}$ composé de M observations où chaque observation $x_i = (x_{i1}, x_{i2}, .., x_{iN})$ est composée de N caractéristiques. Soit d une matrice symétrique de taille $M \times M$ où chaque élément d_{ij} représente la distance entre x_i et x_j et $d_{ii} = 0$. L'algorithme Isomap, qui a X et d en entrée, passe en général par trois étapes :

1. Déterminer un « voisinage » pour chaque point x_i par la méthode des k-plus proches voisins (§ 3.1.1) calculés à partir des distances euclidiennes d en considérant que les distances euclidiennes approchent les distances géodésiques quand les vecteurs se trouvent à petite distance.

2. Estimer la matrice des distances géodésiques (D) entre tous les points. Isomap construit un graphe dont les sommets sont les points et les arêtes sont labelisées par les distances entre eux.

3. Appliquer la méthode MDS à la matrice de distances D pour obtenir un nouveau système de coordonnées euclidiennes qui préserve la géométrie intrinsèque de la variété.

2.2.2.2 Plongement localement linéaire

Plongement localement linéaire, LLE (Locally Linear Embedding) en anglais, est une méthode de réduction non-linéaire proposée en 2000 (Roweis et Saul [2000]). Elle utilise une approche différente d'Isomap mais partage la même philosophie : les points dans un espace de haute dimensionnalité étant voisins doivent se retrouver proches dans une projection de faible dimension. Donc elle s'attache à ce que la disposition des plus proches voisins soit préservée.
LLE n'a pas besoin de calculer les distances entre tous les points de la variété. Il agit sur le voisinage de chaque vecteur pour découvrir la structure globale de l'ensemble. L'idée principale consiste donc à déterminer pour chaque vecteur x_i, point de l'espace contenant les données à projeter, ses k voisins calculés sur la base de la distance euclidienne et de considérer que dans ce voisinage, les données sont linéaires. Une fois les voisins de x_i déterminés, nous calculons les poids de reconstruction w_{ij} représentant le voisinage de chaque vecteur x_i associé à chaque couple (x_i, x_j), où x_j appartient au voisinage de x_i. Les w_{ij} permettront par la suite de reconstruire la topologie de voisinage de chaque vecteur x_i dans l'espace de projection.

2.3 Conclusion

Dans ce chapitre, nous avons traité du domaine de la réduction de dimensionnalité. Dans un premier temps, une revue du domaine de la sélection de caractéristiques a été présentée. Après avoir exposé les différents composants nécessaires à un algorithme de sélection de caractéristiques, les alternatives possibles pour leurs mises en œuvre ont été présentées. Un certain nombre d'algorithmes a alors été décrit en fonction de la combinaison de composants utilisée cela en présentant les avantages et les inconvénients des techniques de sélection en général mais aussi de ceux des algorithmes décrits. Les méthodes existantes présentent des limitations au niveau de la complexité très élevée des approches *"wrapper"* ainsi qu'en raison de la dépendance des caractéristiques pertinentes sélectionnées par rapport au classificateur utilisé. Par ailleurs, les approches *"filter"* présentent plusieurs limitations concernant la

redondance ainsi que les interactions entre les caractéristiques.
Nous avons détaillé dans la deuxième partie de ce chapitre, les techniques de réduction par une transformation de données, en présentant différentes approches linéaires et non linéaires. C'est une approche qui pourrait être incluses dans les méthodes précédentes pour fournir à celles-ci des informations plus variées.

Chapitre 3

Classification supervisée et ensembles de classificateurs

Les méthodes de classification ont pour but d'identifier les classes auxquelles appartiennent des objets à partir de certains traits descriptifs. Elles s'appliquent à un grand nombre d'activités humaines et conviennent en particulier au problème de la prise de décision automatisée. La procédure de classification sera extraite automatiquement à partir d'un ensemble d'exemples. Un exemple consiste en la description d'un cas avec la classification correspondante. Un système d'apprentissage doit alors, à partir de cet ensemble d'exemples, extraire une procédure de classification, il s'agit en effet d'extraire une règle générale à partir des données observées. La procédure générée devra classifier correctement les exemples de l'échantillon et avoir un bon pouvoir prédictif pour classifier correctement de nouvelles descriptions. Dans ce chapitre, une définition du concept d'apprentissage automatique est donnée. Nous introduisons dans les sections suivantes, les concepts de la classification supervisée ainsi qu'un état de l'art des méthodes et algorithmes usuels en apprentissage automatique. Finalement, nous présentons l'approche d'ensembles de classificateurs et nous établissons un état de l'art sur les principaux algorithmes ensemblistes.

3.1 Apprentissage automatique et classification supervisée

L'apprentissage automatique est un domaine dont l'intérêt majeur est le développement des algorithmes permettant à une machine d'apprendre à partir d'un ensemble de données. La motivation originale de ce domaine était de mettre en œuvre des systèmes artificiels intelligents. Les algorithmes issus de ce domaine sont utilisés par plusieurs autres domaines, tels que la vision par ordinateur, la reconnaissance de forme, la recherche d'information, la bioinformatique, la fouille de données et beaucoup d'autres.
Un algorithme d'apprentissage peut être défini comme suit :
C'est un algorithme qui prend en entrée un ensemble de données Ω qui contient les informations nécessaires pour caractériser un problème donné et qui retourne un modèle f qui

représente des concepts caractérisant ces données. Nous appelons Ω un ensemble d'apprentissage ou ensemble d'entraînement et chacun de ses éléments, représentant un exemple d'apprentissage.
Il existe plusieurs types d'apprentissage automatique qui se distinguent essentiellement par leur objectif. Les deux types les plus connus sont : l'apprentissage supervisé et l'apprentissage non supervisé. Si l'on dispose d'un ensemble de points étiquetés, nous parlerons de classification supervisée. Dans le cas contraire, nous effectuons une classification non supervisée.
L'objectif, dans le cas d'apprentissage supervisé, est déterminé explicitement par la prédiction d'une cible. Dans ce cas, pour chaque exemple d'apprentissage x_i de Ω, une cible (étiquette) y_i est associée. La tâche d'un algorithme d'apprentissage est alors d'entraîner un modèle qui puisse prédire, pour une entrée x quelconque, la valeur de la cible y.
La nature de l'ensemble de cibles (noté Y) dépendra du type de problèmes à résoudre. Deux types de problèmes fréquents sont les problèmes de classification et de régression. Pour un problème de classification, Y correspond à un ensemble fini de classes auxquelles peuvent appartenir les différents x_i. Par contre pour celui de la régression, Y correspond à un ensemble de valeurs continues.
Dans le cadre de cette thèse, nous parlerons surtout de l'apprentissage supervisé pour des problèmes de classification. Dans la suite, nous présentons les principaux algorithmes de classification supervisée proposés dans la littérature. Il ne s'agit pas de faire une présentation exhaustive de toutes les méthodes mais seulement de préciser les méthodes les plus classiques que nous utiliserons dans le cadre de notre travail en fonction de leurs propriétés particulières.

3.1.1 k plus proches voisins

La méthode des k plus proches voisins (Knn **k-n**earest **n**eighbor en anglais) (Indyk et Motwani [1998]) se base sur une comparaison directe entre le vecteur caractéristique représentant l'entité à classer et les vecteurs caractéristiques représentant des entités de référence. La comparaison consiste en un calcul de distances entre ces entités. L'entité à classer est assignée à la classe majoritaire parmi les classes des k entités les plus proches au sens de la distance utilisée.
Notons par $X_p = (x_{p1}, x_{p2}, ..., x_{pN})$ le vecteur caractéristique de l'entité p, avec N le nombre de caractéristiques et par p et q deux entités à comparer.
Les distances suivantes sont usuellement employées par les classificateurs Knn :

$$Distance\ Euclidienne : \quad D(X_p, X_q) = \sqrt{\sum_{i=1}^{N} (x_{pi} - x_{qi})^2} \tag{3.1}$$

$$Distance\ de\ Manhattan : \quad D(X_p, X_q) = \sum_{i=1}^{N} (|x_{pi} - x_{qi}|) \tag{3.2}$$

$$Distance\ de\ Minkowski : \quad D(X_p, X_q) = (\sum_{i=1}^{N} (x_{pi} - x_{qi})^r)^{1/r} \tag{3.3}$$

$$Distance\ de\ Tchebychev : \quad D(X_p, X_q) = \max_{i=1}^{N} (|x_{pi} - x_{qi}|) \tag{3.4}$$

Dans la figure 3.1, à gauche, la classification est simple quel que soit le nombre de voisins choisis : le nouvel objet est noir. A droite, en revanche, tout dépend du nombre de voisins choisis et de l'heuristique de classification. Pour k = 1, le nouvel objet est gris. Pour k = 3, si les trois voisins ont le même poids, alors le nouvel objet est noir. Par contre, si le poids est pondéré par l'inverse de la distance alors le nouvel objet peut être gris. Cela revient à pondérer l'affectation de classe avec la distance : plus un voisin est éloigné, plus son influence est faible.

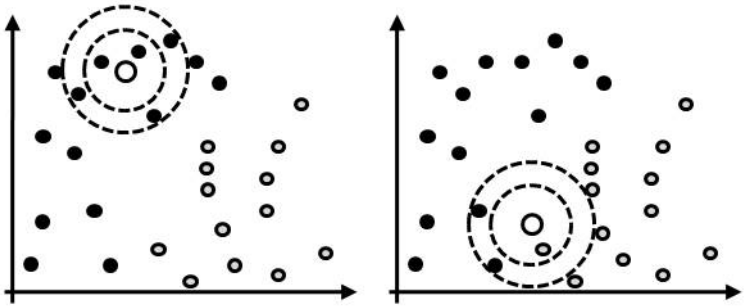

Figure 3.1 – Exemple de classification avec les Knn

Les principaux inconvénients de cette méthode sont le nombre d'opérations nécessaires pour classer une entité dans le cas d'une grande base de référence ainsi que sa sensibilité au bruit présent dans les données d'apprentissage.

3.1.2 Arbres de décision

Le formalisme des arbres de décision permet de classifier un nouvel objet en testant ses caractéristiques les unes à la suite des autres. La classification se fait à travers une séquence de questions dans laquelle chaque question dépend de la réponse à la question précédente. Cette séquence de questions est représentée par un *arbre de décision* dont les feuilles terminales représentent les classes.
La figure 3.2 illustre un exemple de classification sur des données continues en deux dimensions en utilisant les arbres de décision.
Dans la phase de construction de ce classificateur, les exemples de l'ensemble d'apprentissage sont divisés récursivement par des tests définis sur les caractéristiques pour obtenir des sous-ensembles d'exemples ne contenant que des exemples appartenant tous à une même classe. Les algorithmes existants (CART (Breiman *et al.* [1984]), ID3 (Quinlan [1986]), C4.5 (Quinlan [1993]) ...) différent essentiellement par leur façon de choisir, à une étape donnée, et parmi les caractéristiques disponibles, la caractéristique de segmentation et par le critère d'arrêt.

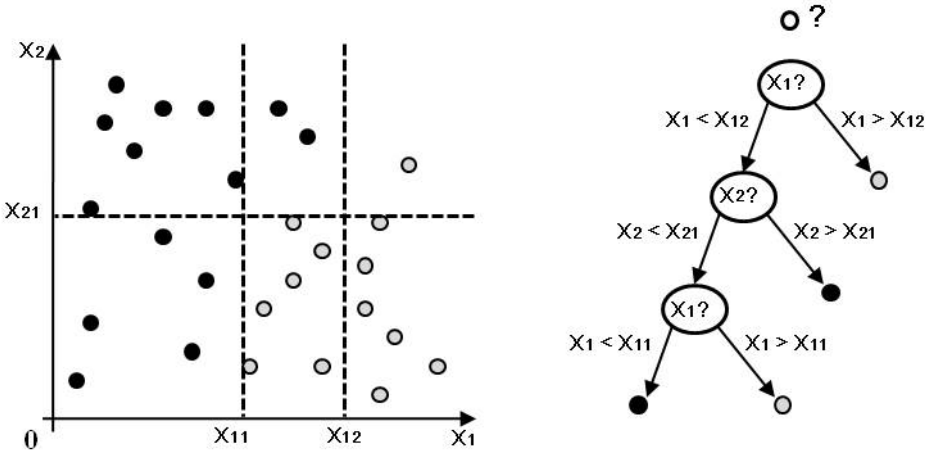

Figure 3.2 – Exemple de classification avec les arbres de décision

3.1.3 Séparateurs à vastes marges

Dans le cas de données linéairement séparables, il n'est pas possible de déterminer de manière unique un hyperplan séparateur en se basant sur le seul critère de minimisation du nombre d'observations mal classées. En 1998, V. Vapnik (Vapnik [1998]) a défini un critère d'optimalité basé sur la "marge" pour séparer des classes linéairement séparables et l'a généralisé à des frontières non linéaires grâce à un changement d'espace. L'hyperplan séparateur a pour équation : $x^T\beta+\beta_0 = 0$, avec x un vecteur caractéristique et β un vecteur de coefficients. Maximiser la marge revient à maximiser la plus petite distance séparant un point de l'espace des observations à l'hyperplan séparateur, ce qui peut s'écrire sous la forme du problème d'optimisation suivant :

$$\begin{cases} \min_{\beta,\beta_0} \|\beta\|^2 \\ \forall j, \quad y^j(x^{j^T}\beta + \beta_0) \geq 1 \end{cases}$$

avec x^j est le vecteur caractéristique de la j^{me} observation et y^j est la classe correspondante. Notons que $y^j = 1$ ou $y^j = -1$ (cas de classification binaire). Ce problème admet une solution unique qui ne dépend que des points situés sur la marge (les points supports), donc les plus difficiles à classer, d'où la robustesse de cette méthode de classification. La fonction de décision des séparateurs à vastes marges (SVM) s'écrit ainsi :

$$f(x) = \beta_0 + \sum_{support} \alpha_i y_i x_i^T x$$

avec α_i les multiplicateurs de Lagrange. Dans le cas de données non séparables linéairement dans leur espace d'origine, un changement d'espace, généralement de dimension plus grande, peut les rendre séparables. A une frontière linéaire dans l'espace transformé, correspond une frontière non linéaire dans l'espace d'origine. L'hyperplan optimal dans l'espace transformé s'écrit $f(x) = \beta_0 + \sum_{support} \alpha_i y_i < \phi(x_i)|\phi(x) > = 0$, avec $\phi(x)$ le transformé de x dans le nouvel espace. Pour éviter de calculer explicitement les transformés des points dans

le nouvel espace, on choisit une transformation qui permet le calcul du produit scalaire $< \phi(x_i)|\phi(x) >$ en fonction de x_i et de x ($< \phi(x_i)|\phi(x) >= K(x_i, x)$). K est appelé *fonction noyau* (ou *kernel*).

Les SVMs (Burges [1998]) présentent également de bonnes performances en terme de généralisation. La généralisation est la faculté d'un classificateur à prédire correctement les classes de nouvelles observations et non pas seulement les classes des observations d'apprentissage. Pour un échantillon d'apprentissage de taille M et une probabilité d'erreur δ, V. Vapnik a vérifié l'inégalité suivante :

$$R \leq R_{emp} + \epsilon(M, \delta)$$

avec R est le risque réel de se tromper et R_{emp} est le risque empirique de se tromper (sur l'ensemble d'apprentissage). Cette inégalité donne une borne pour le risque à partir du risque empirique dépendant de la taille de l'échantillon d'apprentissage et de propriétés intrinsèques au classificateur (la VC-dimension), mais pas de la distribution des observations d'apprentissage, ce qui permet d'assurer une bonne généralisation du classificateur sans avoir besoin de connaître *a priori* la distribution des observations.

Cette inégalité peut être également utilisée pour affiner le classificateur en choisissant les bons paramètres du modèle. En effet, plus le modèle est complexe, plus sa VC-dimension est élevée. Le risque empirique R_{emp} décroît en fonction de la VC-dimension tandis que le terme ϵ croît en fonction de la VC-dimension. Il s'agit donc de trouver le modèle qui assure le meilleur compromis entre *ajustement* et *généralisation*. Il est possible d'utiliser alternativement à cette méthode la méthode de la validation croisée et cela plus particulièrement dans les cas où la VC-dimension n'est pas simple à calculer.

3.1.4 Approche Bayésienne

Un classificateur bayésien (Jensen [2001], Neapolitan [2003]) est basé sur une approche probabiliste employant la règle de Bayes. Notons $P(C_i)$ la probabilité *a priori* d'une classe C_i, $P(x)$ la probabilité d'observer un vecteur caractéristique x et $P(x|C_i)$ la probabilité d'observer le vecteur x sachant que la classe est C_i. La règle de Bayes permet alors de calculer la probabilité *a posteriori* de la classe C_i quand x est observé :

$$P(C_i|x) = \frac{P(x|C_i)P(C_i)}{\sum_j P(x|C_j)P(C_j)}$$

Dans la pratique, puisque le dénominateur de la formule de Bayes ne dépend pas de C_i, nous ne nous intéressons qu'au numérateur. Les probabilités $P(C_i)$ de chaque classe ainsi que les distributions $P(x|C_i)$ doivent être préalablement estimées à partir d'un échantillon d'apprentissage. Le vecteur x est assigné à la classe C_i si :

$$\forall j \neq i, \quad P(C_i|x) > P(C_j|x)$$

L'**analyse discriminante** se présente comme un cas particulier de l'approche Bayésienne. Dans ce cas, les données d'apprentissage sont modélisées par des distributions Gaussiennes. Sur la base des paramètres estimés, des *fonctions discriminantes* sont construites permettant de classer tout vecteur de caractéristiques.

La **régression logistique** admet une hypothèse différente portant sur les probabilités *a posteriori*. Introduite initialement pour résoudre des problèmes de classification binaire, cette méthode modélise le log du rapport des probabilités *a posteriori* par des modèles linéaires :

$$\log \frac{P(C_1|x)}{P(C_2|x)} = a_0 + a_1 x_1 + ... + a_d x_d$$

et comme la somme des deux probabilités *a posteriori* vaut à 1, il en résulte que ces probabilités sont modélisées par des *lois logistiques* :

$$P(C_1|x) = \frac{\exp^{a_0+a_1x_1+...+a_d}}{1+\exp^{a_0+a_1x_1+...+a_d}}, \quad P(C_2|x) = \frac{1}{1+\exp^{a_0+a_1x_1+...+a_d}}$$

3.1.5 Réseaux de neurones

Les réseaux de neurones sont inspirés de la structure neurophysiologique du cerveau. Un neurone formel est l'unité élémentaire d'un système modélisé par un réseau de neurones artificiels. A la réception de signaux provenant d'autres neurones du réseau, un neurone formel réagit en produisant un signal de sortie qui sera transmis à d'autres neurones du réseau. Le signal reçu est une somme pondérée des signaux provenant de différents neurones. Le signal de sortie est une fonction de cette somme pondérée :

$$y_j = f(\sum_{i=1}^{N} w_{ij} x_i)$$

avec y_j la sortie du neurone formel j en question, $x_{i,\ (i=1...N)}$ les signaux reçus par le neurone j de la part des neurones i, et w_{ij} les poids des interconnexions entre les neurones i et j. Selon l'application, la fonction f, appelée *fonction d'activation*, est le plus souvent une fonction identité, sigmoïde, tangente hyperbolique ou une fonction linéaire par morceaux. En classification, les réseaux de neurones (Dietz *et al.* [1989], Broadbent et Lucas [1988]) permettent d'introduire de la non-linéarité dans la séparation entre les classes grâce au choix de la fonction d'activation. Les neurones sont ainsi organisés en trois couches ou plus : les cellules d'entrée associées aux données, les neurones de sortie associés chacun à une classe, et les neurones cachés qui sont entre les neurones d'entrée et les neurones de sortie. L'apprentissage du classificateur consiste à faire évoluer les poids w_{ij} par des méthodes d'optimisation non linéaires pour minimiser une fonction de coût qui constitue une mesure de l'écart entre les réponses réelles du réseau et les réponses désirées.

3.2 Ensemble de classificateurs

Le recours a un ensemble de classificateurs (**E**nsemble **o**f **C**lassifier **EoC**) est une approche qui consiste à produire un modèle de classification plus précis en construisant plusieurs classificateurs différents pour résoudre le problème initial. L'idée principale est de générer, à partir d'un unique algorithme d'apprentissage, un ensemble de classificateurs capables de donner des prédictions différentes sur un même ensemble de données à classer et ce que nous appelons l'induction d'EoC. Les méthodes d'induction d'EoC, sont regroupées en plusieurs classes (Kuncheva [2004]) selon leur niveau d'action. Les deux niveaux les plus connus sont : "Données" et "Caractéristique". Le niveau "Données" regroupe les méthodes basées sur la manipulation des exemples d'apprentissage et leur distribution en générant une sous-base d'apprentissage pour chacun des classificateurs. Au niveau "Caractéristique", les classificateurs seront appris sur des sous-espaces de caractéristiques qui caractérisent le problème.

3.2.1 Fusion de décisions

La méthode la plus couramment utilisée pour fusionner les décisions fournies par plusieurs classificateurs est le vote majoritaire. La sortie de chaque expert est considérée comme étant un vote pour une classe. Le nombre de votes pour chacune des classes est compté et l'ensemble choisit la classe en ayant remporté le plus. Une limitation majeure des méthodes fondées sur le vote majoritaire est liée à l'hypothèse que les différents classificateurs possèdent une fiabilité similaire. Afin de prendre en compte la précision individuelle de chaque classificateur dans la décision finale, il est possible de leur associer un poids w_i proportionnel à sa précision.
Nous trouvons aussi d'autres familles de fusions de décisions basées sur un modèle additif. Les opérateurs de somme, somme pondérée, moyenne ainsi que moyenne pondérée, font partie de cette famille. La sortie de chaque classificateur doit être un vecteur de taux de confiance, ou une probabilité *a posteriori* (à chaque classe correspond un taux de confiance). La fusion est obtenue en appliquant un de ces opérateurs sur les vecteurs de sortie de tous les classificateurs, puis en sélectionnant la classe ayant la valeur la plus élevée.
Une autre méthode de fusion de classificateur, nommée AWFO (**A**ggregation **W**eight-**F**unctional **O**perators) a été proposée en 1996 (Dujet et Vincent [1998]). Dans cette méthode, des poids affectés à chacune des valeurs fournies par les classificateurs sont définis. Ces poids dépendent de la valeur et également de la distribution générale des données.
Nous trouvons aussi d'autres opérateurs de fusions basés sur le produit, la médiane, le minimum, le maximum etc.

3.2.2 Techniques

Au niveau "Données" les deux classes d'algorithmes les plus connues dans la littérature pour construire un ensemble de classificateurs sont le Boosting et le Bagging. Ces deux types d'algorithmes permettent de construire des ensembles de même type différenciés au niveau des exemples manipulés pendant l'apprentissage. Par contre au niveau "Caractéristique", la méthode la plus représentative est la méthode *"Random SubSpace"*. Les forêts aléatoires représentent une méthode qui mélange les deux techniques. Nous présentons dans la suite ces différents types d'algorithmes.

3.2.2.1 Bagging

Le Bagging (*"bootstrap aggregating"*) est une méthode qui consiste à construire un ensemble de classificateurs à partir de différents ré-échantillonnages d'un même ensemble de données d'apprentissage. Cette méthode a été présentée par Breiman en 1996(Breiman [1996]). Cet algorithme utilise une méthode nommée *"bootstrapping"* pour générer différents ensembles. Le *"bootstrapping"* (Efron et Tibshirani [1993]) est une méthode de ré-échantillonnage avec remise. Elle consiste à tirer des exemples de l'échantillon d'apprentissage pour créer L nouveaux ensembles. Ceux-ci sont nommés échantillons *"boostrap"* (*"bootstrap samples"*). Comme ces échantillons sont construits avec remise, des exemples d'apprentissage risquent de s'y retrouver en plusieurs exemplaires. Après avoir généré L échantillons *"boostrap"*, un ensemble de classificateurs h sera construit. Chaque classificateur élémentaire h_i de l'ensemble h, sera entraîné sur un des échantillons de sorte qu'ils soient tous entraînés sur un ensemble d'apprentissage différent. Les classificateurs de l'ensemble h, peuvent ensuite être combinés par un vote majoritaire ou toute autre méthode de fusion. La figure 3.3 décrit le fonctionnement de cet algorithme.

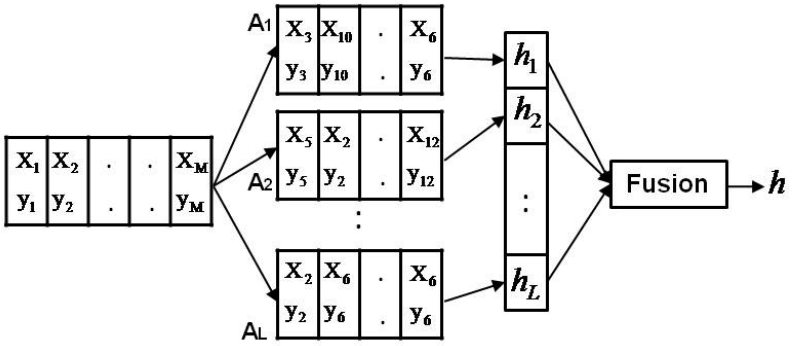

Figure 3.3 – Description schématique d'un ensemble de classificateurs par Bagging

Une étude plus approfondie sur la technique *"bagging"* a été mise en place par (Skurichina et Duin [1998], Grandvalet [2004]). Cette étude a montré que, généralement, le *"bagging"* permet d'améliorer la performance des classificateurs instables. L'instabilité d'un algorithme de classification traduit le fait qu'une légère modification des données d'apprentissage entraîne des différences importantes au niveau de l'estimation des frontières de décision.

Finalement, le *"bagging"* peut réduire l'instabilité des classificateurs comme les arbres de décisions ainsi que les réseaux de neurones afin d'améliorer leurs performances, en revanche sur un classificateur de type Knn, qui est un classificateur stable, son effet est minime.

3.2.2.2 Boosting

Le *"Boosting"* (Schapire [1990]) désigne un principe général d'apprentissage permettant d'améliorer la précision d'un algorithme d'apprentissage donné. Le principe général est de combiner linéairement des résultats de classificateurs dits *"faibles"* afin de construire un classificateur *"fort"* d'apprentissage à partir de l'ensemble original et une méthode de combinaison de classificateurs construits à partir de chaque nouvel ensemble.
Pour définir sa nouvelle technique de *"Boosting"*, Shapire se base sur l'idée que tout classificateur faible capable d'apprendre avec une certaine confiance et une erreur de classification inférieure à $(0,5)$, peut être transformé en un classificateur plus confiant et avec une erreur de classification aussi petite que désirée. En d'autres termes, un classificateur faible donnant de meilleurs résultats qu'un simple pile ou face (50% de risque) peut être la base pour construire un ensemble de classificateurs.
A chaque itération, l'algorithme cherche à trouver un classificateur faible qui peut corriger au mieux les erreurs des classificateurs obtenus aux itérations précédentes. Dans le principe de *"Boosting"*, cet objectif est réalisé à l'aide d'une pondération des données d'apprentissage.
Le premier algorithme dérivé de ce concept est *"AdaBoost"*. Dans la section suivante, nous décrivons en détails de l'algorithme *"AdaBoost"* et quelques unes de ses variantes.

3.2.2.2.a AdaBoost

L'algorithme *"AdaBoost"* (**A**daptive **Boost**ing), développé par Freund et Schapire (Freund et Schapire [1995]), est un algorithme qui crée impérativement une combinaison linéaire de classificateurs de la forme :

$$H(x) \;=\; Signe(\sum_{t=1}^{T} \alpha_t h_t(x)) \tag{3.5}$$

Où h_t représente un classificateur faible et α_t son poids.

Algo 3.1 *"AdaBoost"*

Entrées: Une base d'apprentissage $A = \{X_1, X_2, .., X_M\}$

Les étiquettes $Y = \{y_1, y_2, .., y_M\}$ où $y_i \in \{-1, 1\}$

Algorithme de classification C, Nombre d'itérations T

Sorties: Classificateur fort H

$D_1 = \frac{1}{M}$ /* Initialiser le vecteur de poids des exemples avec $i = 1, 2, .., M$ */

Pour $t = 1$ à T **Faire**

/* Trouver un classificateur faible h_t qui minimise l'erreur selon D_t */

$h_t = Entrainer\_Classificateur\_Faible(C, A, D_t)$

$\epsilon_t = Erreur(h_t, A)$ /* Taux d'erreur de h_t */

$\alpha_t = \frac{1}{2} ln(\frac{1-\epsilon_t}{\epsilon_t})$ /* Poids du h_t */

$D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(X_i))}{Z_t}$ /* Mettre à jour le vecteur de poids */

Fin Pour

Retourner $H(x) = Signe(\sum_{t=1}^{T} \alpha_t h_t(x))$

"AdaBoost" est un algorithme qui prend en entrée, en plus d'un ensemble d'apprentissage, un second algorithme de classification. A chaque itération, *"AdaBoost"* appelle le second algorithme afin d'apprendre un nouveau classificateur qui corrige les erreurs de prédiction de la combinaison linéaire courante. Une fois ce nouveau classificateur appris, il est ajouté dans la combinaison linéaire avec un poids associé qui maximise la performance de la nouvelle combinaison. L'algorithme 3.1 montre le pseudo code d'*"AdaBoost"*.

A chaque étape t de l'algorithme, l'apprenant cherche un bon classificateur $h \in \{-1, 1\}$ pour la distribution de probabilité, *a priori*, sur les exemples d'apprentissage. Cette distribution est représentée grâce à un poids D_t^i pour chaque exemple i qui est mis à jour en fonction de la qualité de la prédiction obtenue à l'étape précédente. L'une des idées principales est de donner aux exemples difficiles à classer un poids D_t^i élevé, (et inversement) de manière à amener l'algorithme à se concentrer sur ces derniers. Chaque classificateur est affecté d'un coefficient proportionnel à l'erreur pondérée. Après T étapes, le classificateur final obtenu est $H(x)$.

La figure 3.4 illustre un exemple d'*"AdaBoost"* appliqué sur des données en deux dimensions.

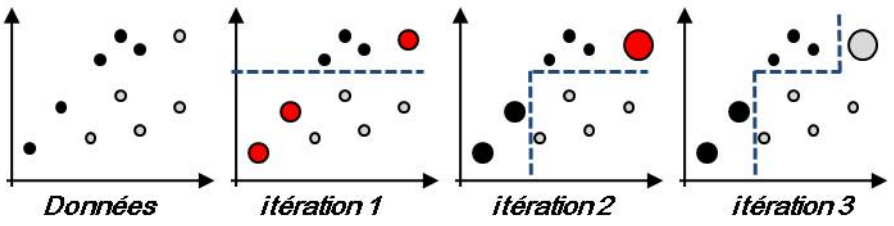

Figure 3.4 – Exemple d'*"AdaBoost"*

Un point assez important d'*"AdaBoost"* est de choisir un classificateur faible. Ce choix constitue un des paramètres de cet algorithme. Ce paramètre peut être n'importe quel algorithme d'apprentissage. Le classificateur faible le plus simple est le classificateur à seuil qui minimise le taux d'erreur (*decision stump* en anglais). Ce type de classificateur correspond à un arbre de décision binaire comportant un seul nœud éclaté en deux feuilles étiquetées par des classes distinctes. Le *"boosting"* a été appliqué aussi avec succès avec plusieurs classificateurs comme les arbres de décision (Dietterich [2000]), les réseaux bayésiens (Hugo *et al.* [199]) ainsi que les réseaux de neurones (Murphey *et al.* [2001]).

3.2.2.2.b Variantes d'AdaBoost

"AdaBoost" a montré une bonne performance ainsi qu'une certaine robustesse au problème de sur-apprentissage sauf s'il y a présence de données bruitées (Dietterich [2000]). Son inconvénient est qu'il augmente (à tort) exponentiellement le poids des exemples bruités afin de trouver des hypothèses faibles qui corrigent les erreurs et qui se concentrent sur ces exemples bruités. La conséquence est que la combinaison des hypothèses faibles tend à sur-apprendre le bruit. Pour éviter ce problème, les techniques proposées apportent des modifications au niveau de la mise à jour des poids des exemples ainsi que sur les poids des classificateurs faibles. Dans la suite, nous présentons quelque variantes d'*"AdaBoost"* proposées dans ce cadre.

"MAdaBoost" (Domingo et Watanabe [2000]) est une variante d'*"AdaBoost"* qui propose une modification simple sur le calcul de poids des exemples. Cette modification a pour but de limiter la croissance incontrôlée des poids. Les auteurs proposent de borner le poids de chaque exemple par son poids initial, ce qui lui évitera de devenir arbitrairement trop grand. L'inconvénient de cette méthode est qu'elle propose une modification qui réduit l'effet positif initial de la croissance exponentielle (rapidité de la convergence).

"IAdaBoost" (Sebban et Suchier [2003]) propose une modification de la mise à jour des poids des exemples dans *"AdaBoost"* en prenant en compte non seulement le problème de sur-apprentissage, mais aussi la vitesse de convergence. Cette mise à jour est calculée en exploitant une mesure d'entropie locale adaptative, issue d'un graphe de voisinage construit sur les exemples.

Modest"AdaBoost" (Vezhnevets *et al.* [2005]) est une autre variante d'*"AdaBoost"* qui a été proposée en 2005. Sa différence principale par rapport aux autres méthodes de *"boosting"* est la façon de calculer les poids des hypothèses faibles (classificateurs faibles). A chaque construction d'une nouvelle hypothèse faible le poids est calculé en prenant en compte la corrélation entre cette nouvelle hypothèse et le classificateur fort construit avant. Plus la corrélation est élevée, plus cette hypothèse sera pénalisée en diminuant son poids.

3.2.2.3 Random Subspaces

Le principe de la méthode des Random Subspaces (Ho [1998]) est assez similaire au Bagging. Le tirage aléatoire des exemples, utilisé dans le *"Bagging"* pour construire un ensemble de classificateurs est remplacé par un tirage aléatoire dans l'espace des caractéristiques. L'idée de base est d'entraîner chaque classificateur élémentaire sur un sous-ensemble aléatoire de caractéristiques. Les sous-ensembles aléatoires ont la même taille P, avec $P < N$ où N est le nombre de caractéristiques total qui représente le problème. Une construction de L sous-ensembles de caractéristiques se fait par un tirage aléatoire sans remise. Après avoir généré les L sous-ensembles, un ensemble de classificateurs h sera construit. Chaque classificateur élémentaire h_i de l'ensemble h, sera entraîné sur un des sous-ensembles de caractéristiques en utilisant la totalité des exemples d'apprentissage. Dans (Ho [1998]), l'auteur a montré que les meilleurs résultats sont obtenus pour $P \approx \frac{N}{2}$. Il a montré aussi que plus l'espace de description présente une certaine redondance d'information dispersée sur l'ensemble des caractéristiques, plus la méthode est efficace.
L'avantage de cette méthode est qu'elle permet de réduire la dimension de l'espace de description des données d'apprentissage. Par conséquence, le temps nécessaire pour la construction de l'ensemble de classificateurs sera réduit. Son architecture aussi nous permet de l'appliquer pour tout type de classificateur élémentaire.

3.2.2.4 Forêts aléatoires

Une forêt aléatoire (Breiman [2001]) est le mélange des deux techniques de *"Bagging"* et de *"Random SubSpace"* appliqués sur des arbres de décisions. A chaque itération, un échantillon *"boostrap"* est tiré aléatoirement afin de construire un arbre de décision binaire. L'espace de recherche pour la construction des nœuds de l'arbre est limité par P caractéristiques tirées aléatoirement. La performance de la méthode dépend directement du paramètre P. Une petite valeur de P risque de dégrader les performances du classificateur. Dans (Breiman [2001]), l'auteur a montré empiriquement que la valeur optimale de P est : $P = \sqrt{N}$ où N est le nombre total de caractéristiques. La technique aléatoire de l'approche a montré sa pertinence et son efficacité spécialement sur des données de haute dimensionalité (avec un nombre de caractéristiques élevé). Cette technique permet une meilleure exploration de l'espace de représentation. Les forêts aléatoires sont aussi utilisées pour résoudre des problèmes de classification dans plusieurs domaines, comme à titre d'exemple, l'imagerie biomédicale (Thomas *et al.* [2008]).

3.2.2.5 DECORATE

L'algorithme DECORATE (**D**iverse **E**nsemble **C**reation by **O**ppositional **R**elabeling of **A**rtificial **T**raining **E**xamples) a été proposé par Melville et Mooney en 2004 (Melville et Mooney [2004]). Dans cette approche, les auteurs proposent une évaluation et une utilisation explicite de la diversité des classificateurs à chaque étape de création de l'ensemble.

L'architecture globale de DECORATE est proche de celle d'*"AdaBoost"*, où l'ensemble de classificateurs est construit de manière incrémentale en modifiant l'ensemble d'apprentissage à chaque itération. DECORATE se distingue cependant d'*"AdaBoost"* au niveau des modifications apportées aux données d'entrainement de chaque classificateur. A chaque itération, un classificateur est entraîné non seulement sur l'ensemble d'apprentissage d'origine, mais sur des données artificielles (appelées données diversifiées) qui sont ajoutées. Les données diversifiées sont constituées d'exemples générés à partir de la distribution du problème mais étiquetés de manière à ce que les différences avec les prédictions de l'ensemble courant soient maximales. Le nombre d'exemples artificiels devant être générés à chaque itération est un paramètre de l'algorithme. Un nouveau classificateur est entraîné sur les données d'origine et ces données diversifiées, avec l'hypothèse selon laquelle ce classificateur augmente la diversité globale de l'ensemble de classificateurs courant. Pour maintenir un niveau de précision raisonnable, seuls les classificateurs qui améliorent la performance de l'ensemble de classificateurs courant sont ajoutés. Ce procédé est répété jusqu'à un nombre d'itérations maximal ou lorsque le nombre de classificateurs souhaité est atteint. Le risque d'avoir trop de rejets, entraîne la construction d'un ensemble de taille inférieure à celle initialement souhaitée.

3.3 Conclusion

Dans ce chapitre, nous avons présenté le concept d'apprentissage automatique et de la classification supervisée ainsi qu'un état de l'art des méthodes et algorithmes usuels en apprentissage automatique. L'approche des ensembles de classificateurs est aussi introduite ainsi qu'un état de l'art sur les principaux algorithmes ensemblistes.

Chapitre 4

Approches génétiques et sélection

Les algorithmes génétiques (AG) ont montré avec succès leur grande capacité à résoudre des problèmes d'optimisation. Ils ont aussi été utilisés dans le domaine de la sélection de caractéristiques. De nombreuses études rapportées dans la littérature ont montré que les méthodes qui utilisent les AGs comme technique de recherche ont donné des meilleurs résultats en comparaison avec les autres méthodes de sélection (Jain et Zongker [1997], Kuncheva et Jain [1999], Ishibuchi et Nakashima [2000]). Dans ce chapitre, nous commençons d'abord par décrire les concepts de base des algorithmes génétiques ainsi que les techniques d'optimisation multi-objectifs et ensuite nous aborderons la présentation des applications des algorithmes génétiques dans les domaines de la sélection de caractéristiques ainsi que de la sélection de classificateurs.

4.1 Algorithmes génétiques

Les algorithmes génétiques (AG) sont des méthodes d'optimisation stochastiques qui font partie des algorithmes évolutionnaires. Les algorithmes évolutionnaires sont des métaheuristiques qui s'inspirent des mécanismes d'évolution darwinienne des populations biologiques.
Les AGs ont été utilisés pour la première fois en 1950 par des biologistes dans le but de simuler l'évolution des organismes. La première adaptation aux problèmes d'optimisation combinatoire de ces algorithmes a été réalisée dans les années 70 par John Holland (Holland [1975]) en développant une analogie entre un individu dans une population et une solution d'un problème dans un ensemble de solutions potentielles. Des travaux de recherche ont été développés ensuite par Goldberg (Goldberg et David [1989]) et Michalewicz (Michalewicz [1992]). Ces travaux ont montré la possibilité d'appliquer les AGs à des problèmes concrets.

4.1.1 Principe

Les AGs sont des algorithmes itératifs basés sur la reproduction et l'évolution naturelle des individus en utilisant les principes de la survie des individus considérés comme les plus forts ou les mieux adaptés à l'environnement. Il s'agit alors de combiner les points forts de chaque individu pour en créer de nouveaux de manière à ce que leur efficacité soit meilleure. Avec les AGs on cherche à optimiser une fonction ($objectif$) donnée dans un espace de recherche, celui des individus. Pour l'optimiser, on définit une fonction d'évaluation ($fitness$) reliée à cette fonction $objectif$ et appliquée sur chaque individu ou chromosome.
En général, le fonctionnement d'un AG est basé sur les phases suivantes (**Figure 4.1**) :

1. **Initialisation** : Générer aléatoirement une population initiale de taille N chromosomes.
2. **Évaluation** : Évaluer chaque individu de la population par la fonction d'évaluation appropriée au problème (fonction de fitness).
3. **Reproduction** : Créer une nouvelle population de N chromosomes par l'utilisation d'une méthode de sélection appropriée et l'application d'opérateurs génétiques (croisement et mutation) sur certains chromosomes au sein de la population courante.
4. **Retour** à la phase 2 tant que la condition d'arrêt du problème n'est pas satisfaite.

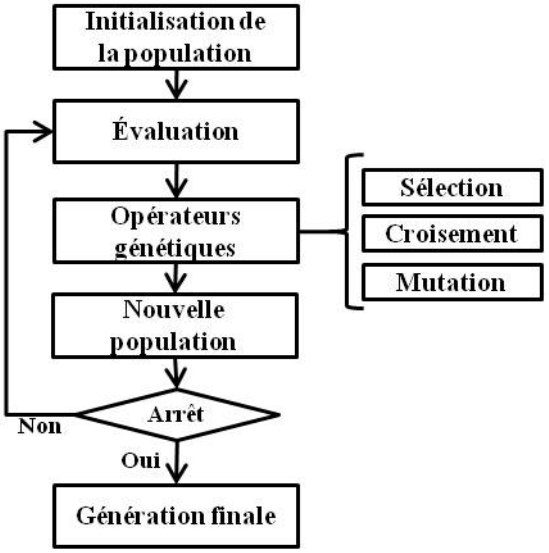

Figure 4.1 – Architecture générale d'un algorithme génétique

4.1.2 Opérateurs génétiques

La reproduction joue un rôle fondamental pour le passage d'une génération à une autre. Elle représente un cycle de l'algorithme génétique. Elle se fait par l'application d'opérations génétiques : la *sélection*, le *croisement* et la *mutation*.

4.1.2.1 Sélection

La sélection consiste à choisir les individus à partir desquels on va créer, la génération suivante, les individus qui survivront, les parents qui par croisement génèreront des enfants, ou les individus qui subiront une mutation. La sélection des individus s'effectue le plus souvent sur la base de leur fonction d'évaluation. Plusieurs opérateurs de sélection existent parmi lesquels les méthodes par probabilité (à la roulette Goldberg et David [1989]), par rang de classement dans la population (Davis [1991]) ou par tournoi (Miller et Goldberg [1995]), etc.

Sélection à la roulette
C'est la méthode de sélection la plus classique et la plus utilisée. Elle consiste à sélectionner un individu avec une probabilité proportionnelle à sa performance. Ainsi un individu x_i, dans une population de taille N, a la probabilité suivante d'être sélectionné :

$$P_{Sel}(x_i) = \frac{fitness(x_i)}{\sum_{j=1}^{N} fitness(x_j)} \tag{4.1}$$

Plus la performance d'un individu est élevée par rapport à celle des autres, plus il a une chance élevée d'être sélectionné et reproduit dans la population. Les individus ayant une grande $fitness$ relative ont donc plus de chance d'être sélectionnés.

Sélection par rang
Elle consiste à attribuer à chaque individu son classement selon l'ordre donné par la fonction d'évaluation qui quantifie l'adaptation de l'individu comme solution du problème. Le plus mauvais individu prendra 1 comme rang, par contre le meilleur aura le rang N, où N est le nombre d'individus de la population. La probabilité de sélection d'un individu x_i devient :

$$P_{Sel}(x_i) = \frac{Rang(x_i)}{\sum_{j=1}^{N} Rang(x_j)} \tag{4.2}$$

Avec cette méthode de sélection, tous les chromosomes ont une chance d'être sélectionnés. Cependant, cette méthode conduit à une convergence plus lente des individus de la population vers la bonne solution. Cela est dû au fait que les meilleurs chromosomes ne diffèrent pas toujours énormément des plus mauvais.

Sélection par tournoi
On tire deux (ou plusieurs) individus aléatoirement dans la population et on sélectionne pour la reproduction, le meilleur des deux (ou plus) pour constituer la nouvelle population.

Sélection par élitisme
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 chromo-

somes 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.

4.1.2.2 Croisement

Le croisement est chargé de construire un individu (ou une solution) qui soit le mélange de plusieurs solutions. Dans le croisement, les chromosomes échangent des séquences de gènes entre eux.
Ce processus est appliqué à chaque paire de chromosomes sélectionnés selon une des méthodes décrites précédemment avec une certaine probabilité P_{crois}. Les paires de chromosomes sont recopiées sans modification dans la génération suivante avec la probabilité $1 - P_{crois}$. Plus P_{crois} est élevée, plus il y a de nouveaux individus qui apparaissent dans la population.
Parmi les méthodes de croisement les plus utilisées, nous présentons les deux plus connues : le croisement à un point et le croisement multi-points.

Croisement à un point : il s'agit de choisir, au hasard, un point de croisement pour chaque couple de chromosomes et d'effectuer une permutation des ensembles des séquences se trouvant de part et d'autre de ce point entre les deux parents (figure 4.2)

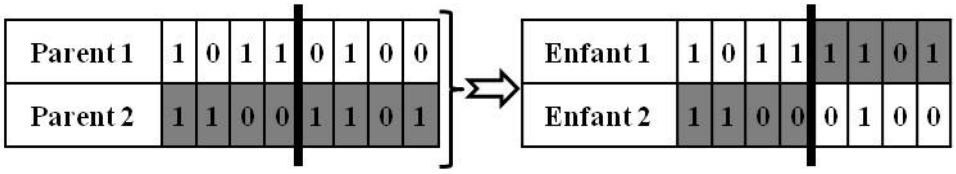

Figure 4.2 – Opérateur de croisement à un point

Croisement multi-points : dans ce cas, plusieurs points de croisement sont sélectionnés et l'échange se fait sur les différentes parties des séquences cernées par ces points, entre les gènes des parents. La figure 4.3 illustre un croisement à deux points.

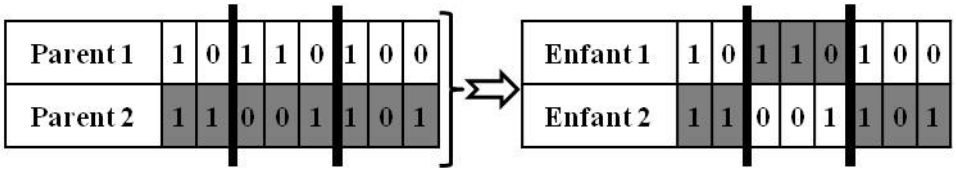

Figure 4.3 – Opérateur de croisement à deux points

4.1.2.3 Mutation

La mutation est définie comme étant la modification aléatoire de la valeur d'un gène dans un chromosome. La figure 4.4 illustre un exemple de mutation appliquée à la quatrième position d'un chromosome binaire. La mutation joue le rôle de bruit qui empêche l'évolution

de se figer. Elle permet d'étendre l'exploration de l'espace et garantit que l'optimum global puisse être atteint. Cet opérateur évite donc une convergence vers les optima locaux.

Figure 4.4 – Exemple d'une opération de mutation

4.1.3 Critère d'arrêt

Le critère d'arrêt a pour but de nous assurer une bonne qualité de la solution finale obtenue par l'algorithme génétique et associée à l'individu de meilleure qualité dans la dernière génération. Le critère d'arrêt peut prendre la forme suivante :

- Le nombre limite de générations autorisées a été atteint.
- Une stabilité de la population a été atteinte (la population cesse d'évoluer ou n'évolue plus suffisamment) de point de vue de la fonction de *fitness*.
- Le meilleur compromis dans le cas d'un problème multi-critères a été atteint.

4.1.4 Fonction de sélectivité

Un algorithme génétique nécessite généralement la définition d'une fonction rendant compte de la pertinence des solutions potentielles à partir des grandeurs à optimiser. Cette fonction est appelée fonction de sélectivité (ou fonction de ***fitness***).

4.2 Optimisation multi-objectifs

Un problème d'optimisation multi-objectifs ou multi-critères consiste à trouver un vecteur de décisions qui optimise plusieurs fonctions *objectif* simultanément et qui satisfait un ensemble de contraintes. Un vecteur de décision est composé de n variables qui représentent les variables du problème. La difficulté principale d'un problème multi-objectifs est qu'il n'existe pas de définition du terme "solution optimale" : une solution peut être préférable à une autre mais il n'existe pas une solution meilleure que toutes les autres et donc la résolution du problème multi-objectifs conduit à l'obtention d'un ensemble de solutions qui constituent des compromis entre les différents objectifs. Il existe plusieurs familles de méthodes dans ce domaine. Les plus connues sont :

Les méthodes agrégées : qui transforment un problème multi-objectifs en un problème simple objectif (Coello et Carlos [1996], Marichal [2000]).

Les méthodes Pareto : qui sont fondées sur la notion de dominance au sens de Pareto et qui privilégient une recherche satisfaisant au mieux tous les objectifs (Fonseca et Fleming [1993], Deb *et al.* [2000]).

4.2.1 Les méthodes agrégées

Ces méthodes sont basées sur la constitution d'une unique fonction *objectif* définie comme une combinaison des objectifs. Elles sont utilisées avec les modèles de combinaison tels que le modèle additif et le modèle multiplicatif.
L'application de ces modèles n'est possible que sur des objectifs commensurables (exprimés dans la même unité). En d'autres termes, il est difficile d'utiliser ces modèles avec des critères qui peuvent être quantitatifs pour certains et qualitatifs pour d'autres. Nous allons présenter quelques méthodes relevant de ces deux types d'approches.
Dans les trois sections suivantes, nous présentons quelques techniques d'optimisation basées sur l'agrégation des objectifs.

4.2.1.1 La moyenne pondérée

Un coefficient de poids est affecté à chacun des objectifs. Ce coefficient représente l'importance attribuée à l'objectif. Le problème devient un problème simple objectif de la forme : $\sum_{i=1}^{K} w_i * f_i$
avec K le nombre d'objectifs, w_i le poids affecté au *ième* objectif (f_i) et $\sum_{i=1}^{K} w_i = 1$.
Cette méthode est très efficace et elle est simple à mettre en œuvre mais les difficultés principales résident d'une part dans la détermination du poids de chaque objectif et d'autre part dans l'expression des interactions entre les objectifs.
La variation des poids dans l'utilisation d'une combinaison linéaire des objectifs peut constituer une solution pour résoudre la première difficulté. Par contre, la résolution du deuxième problème est plus difficile car il existe plusieurs types d'interactions entre les objectifs qui sont difficiles à exprimer à l'aide d'une somme pondérée. Des solutions basées sur l'intégrale de Choquet (Choquet [195]) sont proposées dans (Marichal [2000]). Cette intégrale permet de prendre en compte les différentes interactions.

4.2.1.2 Goal programming

Cette méthode est connue aussi sous le nom de *"target vector optimisation"* (Coello et Carlos [1996]). Elle cherche à minimiser la somme des écarts entre les résultats des objectifs initiaux (ou individuels) et des valeurs qui ont été fixées comme buts idéaux à atteindre

pour chaque objectif. La nouvelle fonction *objectif* devient :

$$min(\sum_{i=1}^{K} |f_i - T_i|) \tag{4.3}$$

où T_i est la valeur à atteindre comme but pour le $i^{\text{ème}}$ objectif (f_i).
Comme la somme pondérée, cette méthode est facile à mettre en œuvre mais la définition des objectifs à atteindre reste une difficulté qui détermine l'efficacité de la méthode.

4.2.1.3 Le min-max

D'une manière proche de la méthode précédente, cette méthode cherche à minimiser le maximum de l'écart relatif entre l'objectif et son but à atteindre. La nouvelle fonction *objectif* devient :

$$min\ max_i \left(\frac{w_i * (f_i - T_i)}{T_i} \right) \tag{4.4}$$

où $i = 1, ..., K$, T_i est la valeur du but associé au $i^{\text{ème}}$ objectif (f_i) et w_i est le poids associé à (f_i).
Pour plus de détails sur la méthode min-max, le lecteur pourra consulter la référence (Coello et al. [1995]) qui présente également des variantes et applications.

4.2.2 Les méthodes Pareto

La première méthode qui utilise la dominance au sens de Pareto pour résoudre des problèmes multi-objectifs a été proposée en 1989 par Goldberg (Goldberg et David [1989]). Dans ses travaux, Goldberg a suggéré d'utiliser ce concept pour assurer l'intégralité de chaque critère et pour trouver un ensemble de compromis optimaux entre les objectifs.

4.2.2.1 Optimum de Pareto

Avant de parler de l'optimum de Pareto, nous rappelons la définition de la notion de dominance de Pareto. Pour des problèmes de minimisation, si on a un ensemble F composé de n fonctions objectif $(F = \{f_1, f_2, ...f_n\})$ et deux points x et x', on dit que x domine x' si :

$\forall i, f_i(x) \leq f_i(x')$ avec au moins un indice i tel que l'inégalité $f_i(x) < f_i(x')$, soit stricte.

Les définitions utiles au problème de maximisation $(>, \geq)$ sont analogues.
La figure 4.5 montre un exemple de dominance. Dans cet exemple, il y a deux fonctions *objectif* à minimiser. La région en gris clair représente les solutions qui dominent la solution A et la région en gris foncé contient les solutions dominées par la solution A. Les deux régions blanches contiennent des solutions non comparables à A.

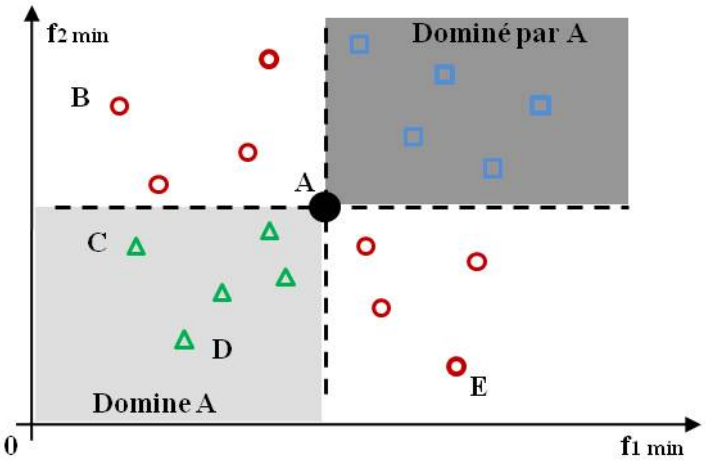

Figure 4.5 – Exemple de dominance

Un point est dit Pareto-optimal s'il n'est dominé par aucun autre point. Les points B,C,D et E de la figure 4.5 sont optimaux au sens de Pareto. Ces points sont tous incomparables les uns par rapport aux autres. Ils sont également appelés solutions non inférieures ou non dominées.

4.2.2.2 La frontière de Pareto

La frontière de Pareto est l'ensemble de tous les points optimaux au sens de Pareto. La figure 4.6 présente, pour un problème à deux objectifs, les frontières de Pareto en fonction du choix de l'utilisateur de minimiser ou de maximiser les objectifs.

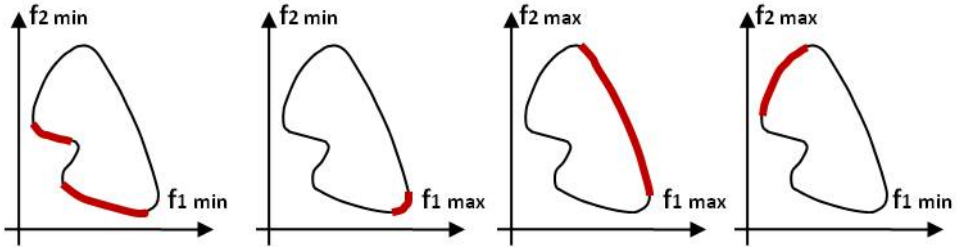

Figure 4.6 – Exemples de fronts de Pareto

Dans les deux sections suivantes, nous présentons des techniques d'optimisations fondées sur le Pareto-optimal.

4.2.2.3 Multiple Objective Genetic Algorithm (MOGA)

Cette méthode a été proposée en 1993 par Fonseca et Fleming (Fonseca et Fleming [1993]). La fonction de *fitness* pour évaluer un individu est basée sur le rang de cet individu et sur le nombre d'individus ayant le même rang. Le rang d'un individu de la population est relatif au nombre d'individus qui le dominent. Tous les individus non dominés ont le rang

1.
Cette méthode donne des solutions de bonne qualité et son implémentation est facile. En revanche, la sélection par rang risque de répartir la population autour d'un même optimum, ce qui donne à la fin une seule solution proposée qui peut ne pas être satisfaisante pour l'utilisateur. Pour éviter ce problème, les auteurs proposent d'utiliser une fonction de partage (fonction de sharing). L'utilisation de cette fonction a pour but de répartir la population sur l'ensemble de la frontière Pareto et d'éviter le regroupement des individus autour d'un optimum local. Pour cela, la fonction de partage se base sur d'autres critères pour changer la note d'un individu. Un des critères utilisés est le degré d'isolement de l'individu. Par exemple, un individu isolé qui a une note moyenne aura un score plus élevé qu'un individu qui a une très bonne note mais dans une zone très peuplée. La performance finale de la méthode dépend toujours des paramètres utilisés pour la fonction de partage.

4.2.2.4 Non dominated Sorting Genetic Algorithm (NSGA)

En 1994, Srivinas et Deb (Srinivas et Deb [1994]) ont développé une méthode basée sur la dominance au sens de Pareto appelée NSGA. Dans cette méthode, l'évaluation de la population se fait par des évaluations partielles sur des groupes d'individus, chaque groupe i représente les éléments qui constituent la $i^{\text{ème}}$ frontière de Pareto. L'évaluation commence par la première frontière en donnant une valeur factice de *fitness* et en appliquant une fonction de partage. Une fois ce groupe évalué, il sera supprimé et le processus se répète jusqu'à ce que tous les groupes aient été évalués. L'algorithme se déroule ensuite comme un algorithme génétique classique.
Le tri des solutions en différentes frontières assure d'une part une répartition plus efficace sur la frontière Pareto et par ailleurs maintient la diversité de la population. De plus, cette méthode est applicable dans des problèmes avec un nombre quelconque d'objectifs. Cependant, la complexité de calcul et le paramétrage de la fonction de partage représentent deux inconvénients de cette méthode. Un autre problème apparaît avec la méthode de tri utilisée qui conduit au ralentissement du processus de convergence de l'algorithme.
En 2000, une deuxième version de NSGA (NSGA II) a été proposée par Deb (Deb *et al.* [2000]). Dans cette version, l'auteur propose quelques solutions qui limitent les principaux problèmes de la version originale.

4.3 Algorithme génétique et sélection de caractéristiques

Les algorithmes génétiques ont montré leur efficacité pour résoudre des problèmes d'optimisation dont l'espace de recherche est de grande dimension. L'application des AGs sur des problèmes de sélection de sous-ensembles de caractéristiques a été mise en place en 1993 par Ferri et al. (Ferri *et al.* [1993]) qui ont montré que l'utilisation des AGs est bien adaptée pour la sélection sur des ensembles de caractéristiques de taille moyenne (20 à 49 caractéristiques) pour lesquels la plupart des méthodes classiques nécessitent un temps de

calcul énorme pour réaliser une sélection.
En 2000, Kudo et Sklansky (Kudo et Sklansky [2000]) ont montré la possibilité d'utiliser les AGs pour la sélection sur des ensembles de grande échelle (50 caractéristiques et plus) en ajustant les paramètres de l'AG (le nombre de générations, la taille de la population et les probabilités des opérations génétiques) d'un côté et la fonction d'évaluation de l'autre côté. Une fois que les paramètres ont été bien fixés et la fonction d'évaluation bien définie, ils ont montré que les résultats de l'AG, sont meilleurs que ceux des méthodes basées sur la recherche séquentielle. La procédure de sélection par un algorithme génétique est illustrée par la figure (4.7).

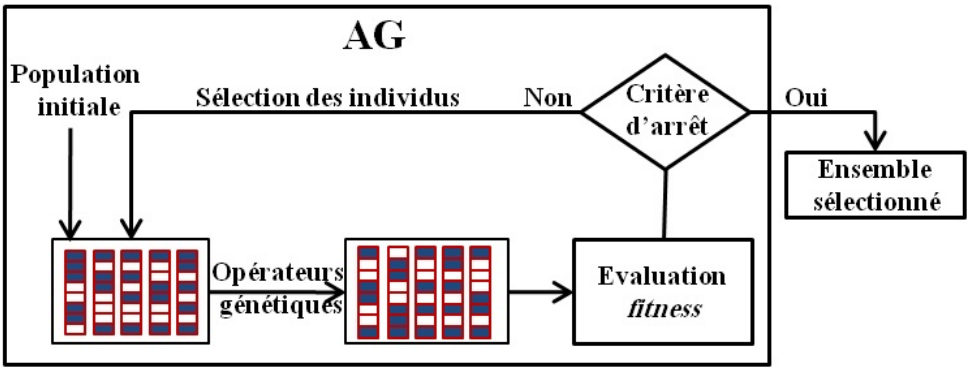

Figure 4.7 – Sélection de caractéristiques par un algorithme génétique

4.3.1 Codage et fonction de *fitness*

La représentation la plus simple pour coder un sous-ensemble de caractéristiques est le codage binaire. Chaque chromosome de la population est représenté par une chaîne binaire de taille fixe N qui représente le nombre total de caractéristiques du problème. Chaque gène du chromosome représente la présence ou l'absence de la caractéristique correspondante. Avec les AGs qui utilisent ce codage pour la sélection de caractéristiques, un des problèmes qui se pose au niveau de l'initialisation de la population initiale est le nombre de '1' dans chaque chromosome. En général, la valeur de chaque gène d'un chromosome est déterminée aléatoirement avec une probabilité de $0,5$ pour que la valeur soit '0' ou '1', ce qui signifie qu'il y a toujours environ 50% des gènes du chromosome qui ont la valeur '1' et ce qui conduit probablement à une sélection d'environ 50% des caractéristiques à la fin. Leardi a proposé (Leardi [1994]) d'autres probabilités pour avoir un nombre plus faible de '1' (la valeur par défaut est 10% des caractéristiques initiales). Une autre possibilité pour minimiser le nombre de caractéristiques sélectionnées est d'intégrer cet objectif dans la fonction d'évaluation.
La classification des méthodes de sélection par un algorithme génétique dépend de la fonction d'évaluation choisie. Le but de cette fonction d'évaluation peut être mono-objectif ou multi-objectifs. Une optimisation mono-objectif par un AG, pour un problème de sélection de caractéristiques, cherche à trouver le sous-ensemble de caractéristiques qui minimise

l'erreur de classification. Dans la littérature, les méthodes de sélection par un algorithme génétique proposées sont plutôt multi-objectifs et cherchent à optimiser plusieurs objectifs à la fois (minimiser l'erreur de classification, minimiser le nombre de caractéristiques sélectionnées, minimiser le nombre d'exemples d'apprentissage, etc.). Ces méthodes peuvent être classées en deux catégories : sélection par une agrégation des objectifs et sélection fondée sur le principe de Pareto.

Sélection par une agrégation des objectifs : L'agrégation des différents objectifs par une somme pondérée est l'approche la plus utilisée dans les méthodes de sélection de cette catégorie. Siedlecki et Sklansky ont proposé une méthode de sélection par un AG en 1993 (Siedlecki et Sklansky [1993]). Cette méthode cherche à trouver un sous-ensemble de caractéristiques de taille minimale qui conduise à une erreur de classification inférieure à un seuil prédéfini (t). La fonction de *fitness* pour un individu a_i est définie par :

$$fitness(a_i) = J(a_i) - (\mu_J - \eta\sigma_J) \tag{4.5}$$

où μ et σ représentent respectivement la moyenne et l'écart-type des valeurs de J appliquée à tous les individus de la population. η représente une valeur très petite qui assure que la valeur de $min(fitness(a_i))$ soit toujours positive. Finalement le score $J(a)$ pour un individu a est calculé par :

$$J(a) = Longueur(a) - P(e(a)) \tag{4.6}$$

où $Longueur(a)$ est le nombre des bits qui ont "1" comme valeur dans a et P(e) est une fonction de pénalité sur l'erreur de classification de l'individu $(e(a))$. Si le taux d'erreur est inférieur au seuil t alors P(e) est négatif. La valeur de P croît exponentiellement en fonction de e. La fonction P(e) est donnée par (pour simplifier la formule, nous avons remplacé $e(a)$ par e qui représente l'erreur de classification d'un individu (a) :

$$P(e) = \frac{exp(\frac{(e-t)}{\xi} - 1)}{exp(1) - 1} \tag{4.7}$$

où ξ est un paramètre d'échelonnage (Scaling parameter) de petite valeur (en général 0,01).

Yang et Honavar (Yang et Honavar [1998]) proposent une méthode de sélection qui utilise un AG avec un classificateur modélisé par un réseau de neurones. Leur fonction d'évaluation combine deux critères : le taux de classification par un réseau de neurones et une fonction de coût de la classification. La fonction de *fitness* est définie par :

$$fitness(\Omega) = Taux(\Omega) - \frac{coût(\Omega)}{Taux(\Omega) + 1} + coût_{max} \tag{4.8}$$

où Ω est le sous-ensemble courant, $Taux(\Omega)$ représente le taux de classification d'un réseau de neurones sur une base de test en utilisant le sous-ensemble Ω, $coût(\Omega)$ est le coût de classification du classificateur basé sur Ω qui est calculé en fonction d'un certain nombre de

mesures comme par exemple le temps d'évaluation ou la complexité. $coût_{max}$ est la valeur maximale des coûts calculés sur tous les individus de la population.
D'autres méthodes de cette catégorie sont proposées dans (Kuncheva et Jain [1999], Ishibuchi et Nakashima [2000]). Ces méthodes cherchent à optimiser simultanément trois objectifs (minimiser le taux d'erreur, minimiser le nombre de caractéristiques sélectionnées et minimiser le nombre d'exemples d'apprentissage).
Des travaux de recherche sur la sélection de caractéristiques dans le domaine de la bioinformatique qui utilisent cette technique ainsi que d'autres types d'algorithmes évolutionnistes et d'autre approches d'évaluations, ont été réalisés. Dans ces travaux, plusieurs objectifs sont utilisés pour sélectionner le meilleur sous-ensemble de caractéristiques. Par exemple, en plus de l'erreur de classification d'un classificateur SVM, la marge d'un classificateur SVM linéaire a été utilisée comme deuxième objectif à optimiser. Pour plus de détails, le lecteur peut consulter (Duval et Hao [2009]).

Sélection fondée sur le principe de Pareto : L'optimisation par l'approche Pareto est aussi utilisée pour la sélection des caractéristiques. Une des méthodes de sélection de caractéristiques a été proposée par Oliveira et al. en 2002 (Oliveira *et al.* [2002]). Cette méthode utilise la méthode d'optimisation NSGA (section 4.2.2.4) et cherche à optimiser les deux objectifs : maximiser le taux de classification d'un classificateur par réseau de neurones et minimiser le nombre de caractéristiques sélectionnées. Après avoir sélectionné les solutions données par le premier front de Pareto de la méthode NSGA, cet ensemble de solutions est testé sur une base de validation afin de sélectionner la meilleure.
Une autre méthode de sélection, qui utilise MOGA (section 4.2.2.3) comme technique d'optimisation, a été proposée en 2007 par Kitoogo et Baryamureeba (Kitoogo et Baryamureeba [2007]). Cette méthode cherche à minimiser l'erreur de classification d'un classificateur Knn et le temps d'évaluation pour un sous-ensemble de caractéristiques.

4.4 Algorithmes génétiques et sélection de classificateurs

Les études dans la littérature ont montré qu'un ensemble de classificateurs est plus performant qu'un classificateur simple. La performance d'un ensemble de classificateurs peut être améliorée de différentes manières :

- Sélectionner un sous-ensemble de classificateurs à partir de l'ensemble initial peut conserver ou améliorer la performance de l'ensemble de base (c'est l'équivalent d'une sélection de caractéristiques pour des classificateurs).
- Construire un nouvel ensemble de classificateurs, plus performant, à partir de l'ensemble de base (c'est l'équivalent d'une extraction de caractéristiques pour des classificateurs).

Dans les deux cas, le but est d'améliorer la performance d'un ensemble de classificateurs mais les techniques sont différentes. Dans cette section, nous présentons des techniques de

sélection des classificateurs (premier cas) qui cherchent à sélectionner un sous-ensemble des classificateurs à partir d'un ensemble de classificateurs déjà construits.
Une étude approfondie pour sélectionner des classificateurs Knn générés par la méthode RS (*"Randon SubSpace"* , section 3.2.2.3) a été réalisée par Tremblay et al. en 2004 (Tremblay *et al.* [2004]). Le but était de trouver le sous-ensemble de classificateurs Knn de taille minimale le plus performant. Comme pour le problème de la sélection des caractéristiques, celui de la sélection des classificateurs nécessite la construction d'un algorithme de recherche ainsi que la définition de critères de sélection. Dans leurs études, ils ont montré l'intérêt d'utiliser un algorithme génétique comme algorithme de recherche et étudié sa performance sur des ensembles de classificateurs de grande taille (100 classificateurs dans leur cas). En se basant sur l'idée que la diversité des classificateurs est l'un des critères importants pour construire un bon ensemble de classificateurs, les auteurs ont proposé d'optimiser deux critères de sélection en même temps : maximiser la performance et la diversité de l'ensemble et donc un algorithme génétique multi-objectifs a été utilisé. La performance d'un ensemble a été mesurée par le taux de bonne classification sur un ensemble de tests et plusieurs mesures de diversité ont été étudiées. Ces mesures seront présentées dans la section 6.2.1 du chapitre 6.
Différentes versions d'AG ont été appliquées pour trouver le meilleur sous-ensemble de classificateurs selon différents critères de sélection. Dos Santos et al. ont utilisé en 2006 (Dos Santos *et al.* [2006]) les techniques génétiques pour la sélection des classificateurs en ne considérant pas seulement la performance des sous-ensembles mais aussi leur cardinalité. Ils ont étudié aussi le phénomène de sur-apprentissage. Dans la même étude, les auteurs ont testé des algorithmes génétiques simples en optimisant à chaque fois un des critères de sélection et ils ont constaté que la performance d'un sous-ensemble est le meilleur critère simple qui peut améliorer la performance globale. Plusieurs mesures de diversité ont été utilisées comme un critère simple à optimiser et avec comme deuxième objectif la performance de l'ensemble. Les auteurs ont conclu finalement que la diversité seule ne peut pas être un critère de sélection qui peut conduire à trouver le sous-ensemble de classificateurs le plus performant. Par contre, l'intégration de la diversité avec la performance, dans la fonction d'évaluation (optimisation multi-objectifs) améliore les résultats finaux.

4.5 Conclusion

Dans ce chapitre, nous avons présenté les techniques génétiques et les techniques d'optimisation multi-objectifs ainsi que leur application dans le domaine de la sélection de classificateurs et la sélection de caractéristiques. Après avoir donné une introduction rapide sur les algorithmes génétiques et leur manière d'optimiser un objectif et après avoir présenté les différentes techniques d'optimisation mutli-objectifs par *agrégation* et par les méthodes de *Pareto*, dans le domaine de la sélection, une revue des travaux de recherches qui utilisent ces différentes méthodes d'optimisation a été présentée. L'introduction des

algorithmes génétiques a aidé à réduire le temps de calcul grâce à ces techniques aléatoires pour la génération des sous-ensembles. Mais cette réduction n'est pas suffisante pour que les méthodes de type "wrapper" soient toujours réalisables. Les techniques d'optimisation multi-objectifs ont permis de leur côté l'introduction de plusieurs objectifs. Ceci qui aide à trouver des sous-ensembles plus performants en intégrant différentes mesures à optimiser. Dans la suite, nous nous inspirons de ces techniques où nous proposons une nouvelle méthode de sélection de caractéristiques dans le but de limiter les inconvénients des méthodes existantes pour la sélection.

Deuxième partie

Une nouvelle méthode de sélection de caractéristiques

Chapitre 5

Le principe de la sélection

Comme nous avons vu dans la première partie, les méthodes de filtrage pour la sélection de caractéristiques présentent des limitations dans leur prise en compte des interactions potentielles entre les caractéristiques. Les méthodes enveloppantes (*"wrapper"*) souffrent de leur côté d'une complexité très élevée ainsi qu'une dépendance aux classificateurs utilisés pour l'évaluation. Dans le but de limiter ces inconvénients, nous proposons une nouvelle méthode de sélection de caractéristiques basée sur une sélection de classificateurs simples. Nous profitons de la rapidité des méthodes de filtrage qui se basent sur une évaluation individuelle des caractéristiques pour définir un ensemble de classificateurs simples construits à partir de chacune des caractéristiques. Pour prendre en compte l'interaction entre les caractéristiques, nous proposons d'utiliser un algorithme génétique pour sélectionner la meilleure combinaison des classificateurs. Nous utilisons une méthode de combinaison de classificateurs pour définir la fonction de *fitness*. Les caractéristiques associées au sous-ensemble final de classificateurs sélectionnés par notre méthode représentent le sous-ensemble final de caractéristiques.
Dans ce chapitre, nous présentons en détail cette nouvelle méthode de sélection, son processus et ses étapes. Nous la validons ensuite par une phase d'expérimentation et nous montrons sa capacité à sélectionner. Finalement, nous terminons par une comparaison entre notre méthode et d'autres méthodes de la littérature.

5.1 Processus de sélection

On dispose d'un ensemble $F = f_1, f_2, .., f_N$ composé de N caractéristiques et d'un ensemble d'apprentissage $B_{app} = \{X_1, X_2, .., X_M\}$ composé de M échantillons (exemples) où chaque $X_i = (f_{i1}, f_{i2}, .., f_{iN})$ représente le $i^{\text{ème}}$ exemple. Un exemple est représenté par un vecteur dont les composantes sont les valeurs des caractéristiques (f_i), de dimension N, où N est le nombre total de caractéristiques. Cet ensemble d'apprentissage est divisé en deux parties :

- Une base d'apprentissage A qui contient M_A exemples

- Une base de validation V qui contient M_V exemples

Finalement on dispose de $Y = \{y_1, y_2, .., y_M\}$ l'ensemble des étiquettes des exemples de la base. Pour un problème de classification bi-classe y_i appartient à $\{-1, 1\}$.
La figure 5.1 représente le schéma général du processus de notre méthode de sélection. Ce processus comporte trois étapes :

- La construction de N classificateurs simples H_i grâce à un apprentissage sur la base A, pour chacun n'est pris en compte que la $i^{\text{ème}}$ caractéristique f_i.
- Une présélection sur les caractéristiques initiales (f_i) pour i variant de 1 à N, notées (f_i^1) pour i variant de 1 à k et associées aux classificateurs (H_i^1) correspondants. Cette présélection est basée sur la performance des classificateurs H_i.
- Une sélection sur les classificateurs (H_i^1) par un algorithme génétique utilisant une nouvelle base d'exemples que nous avons notée V.

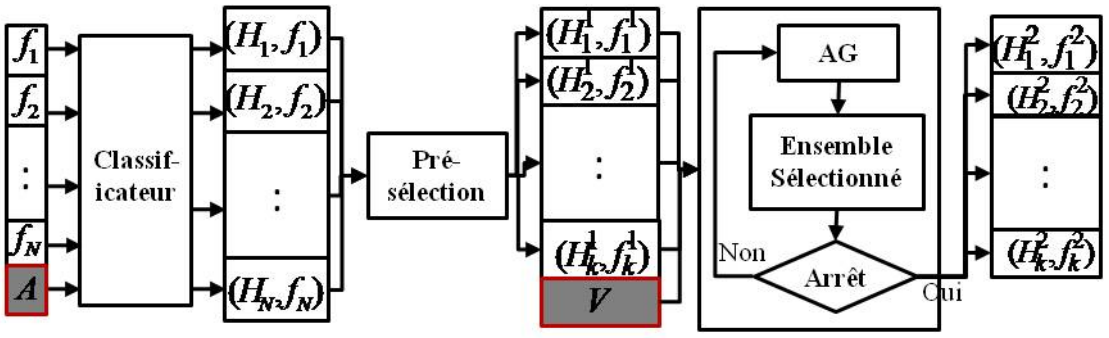

Figure 5.1 – Schéma général du processus de sélection

L'objectif de la première étape de notre processus est de constituer un ensemble de classificateurs qui représentent les caractéristiques initiales. Ils seront ensuite proposés comme entrées de l'algorithme génétique. Chaque classificateur est un modèle simple, entraîné sur une seule caractéristique. Dans la deuxième étape, une pré-sélection est appliquée afin d'éliminer les caractéristiques peu pertinentes et de réduire le nombre total de caractéristiques. Ceci aidant par ailleurs à accélérer les étapes suivantes. Dans cette étape, des méthodes classiques de filtrage de caractéristiques peuvent être utilisées. Dans notre cas, nous utilisons le taux d'erreur des classificateurs construits dans la première étape comme score de pré-sélection. Une fois cet ensemble de classificateurs construit, les plus mauvais classificateurs étant éliminés, nous appliquons dans la troisième étape un algorithme génétique pour sélectionner de proche en proche un "bon" sous-ensemble de classificateurs. Les caractéristiques associées aux modèles finalement sélectionnés représentent le sous-ensemble de caractéristiques recherché.

5.2 Construction de l'ensemble de classificateurs

Cette étape consiste à construire un ensemble de classificateurs où chaque élément est un classificateur entraîné sur une seule caractéristique. Un classificateur appris sur une seule caractéristique est un modèle simple défini en utilisant un algorithme d'apprentissage sur la base d'apprentissage A et en ne prenant en compte qu'une seule caractéristique à la fois. Ce type de classificateur doit être simple et le plus performant possible même sur une seule caractéristique. Nous allons analyser les méthodes qui existent et proposer un mode de construction original.

5.2.1 Cas d'un seul seuil de classification

Dans (Alamdari [2006]), un classificateur binaire simple de ce type a été proposé dans le but d'évaluer les caractéristiques individuellement. Il était utilisé comme un critère de sélection dans une méthode de filtrage de caractéristiques. Le seuil utilisé est le milieu du segment dont les extrémités sont les isobarycentres des valeurs de la caractéristique des données de chacune des classes. Dans la suite, nous utilisons le nom *"Classif_Alamdari"* pour désigner ce type de classificateur. Plus précisément, pour la $i^{\text{ème}}$ caractéristique, le classificateur est défini à partir d'un seuil y.
Soit $X^i = \{f_{1i}, f_{2i}, .., f_{Mi}\}$ l'ensemble associé à une caractéristique où chaque élément f_{1i} est la valeur de la $i^{\text{ème}}$ caractéristique des exemples d'apprentissage. Soit $X^{i,1} = \{f_{ki} | y_k = 1\}$ et $X^{i,-1} = \{x_{ki} | y_k = -1\}$.

$$y = signe((f_i - \frac{\mu_i^1 + \mu_i^{-1}}{2})(\mu_i^1 - \mu_i^{-1})) \tag{5.1}$$

Où μ_i^1 et μ_i^{-1} représentent les moyennes des données relatives à la *ième* caractéristique respectivement de la classe "1" et de la classe " -1".
Un autre classificateur de ce type est le *"decision stump"*. Ce classificateur est défini par un arbre de décision qui contient un seul nœud connecté directement aux nœuds terminaux de l'arbre. Ce nœud définit le "meilleur" seuil, en général celui qui minimise l'erreur de classification sur une seule caractéristique. Ce classificateur fait partie des classificateurs faibles les plus connus, ceux utilisés par l'algorithme *"AdaBoost"*.

5.2.2 Cas de plusieurs seuils de classification

Avec une seule caractéristique, nous proposons d'introduire plusieurs seuils de classification pour construire notre classificateur. Ces seuils peuvent être construits par les seuils associés aux nœuds d'un arbre de décision (section 3.1.2) ou bien en utilisant un algorithme *"AdaBoost"* (section 3.2.2.2.a) construit à partir de différents classificateurs faibles de type *"decision stump"*. A partir d'un classificateur de type "*decision stump*" initial h_{s1}, nous pouvons définir un nouveau classificateur de même type en modifiant les poids des exemples de

l'ensemble d'apprentissage qui sont mal classés par h_{s1} pour obtenir un nouveau classificateur $\alpha_1 h_{s1} + \alpha_2 h_{s2}$. La construction d'un classificateur est alors issue de l'application de ce principe de manière itérative. Nous avons fixé à 20 le nombre d'itérations afin d'éviter le problème de sur-apprentissage. Le processus de construction d'un classificateur H sur une caractéristique f est illustré dans la figure 5.2.

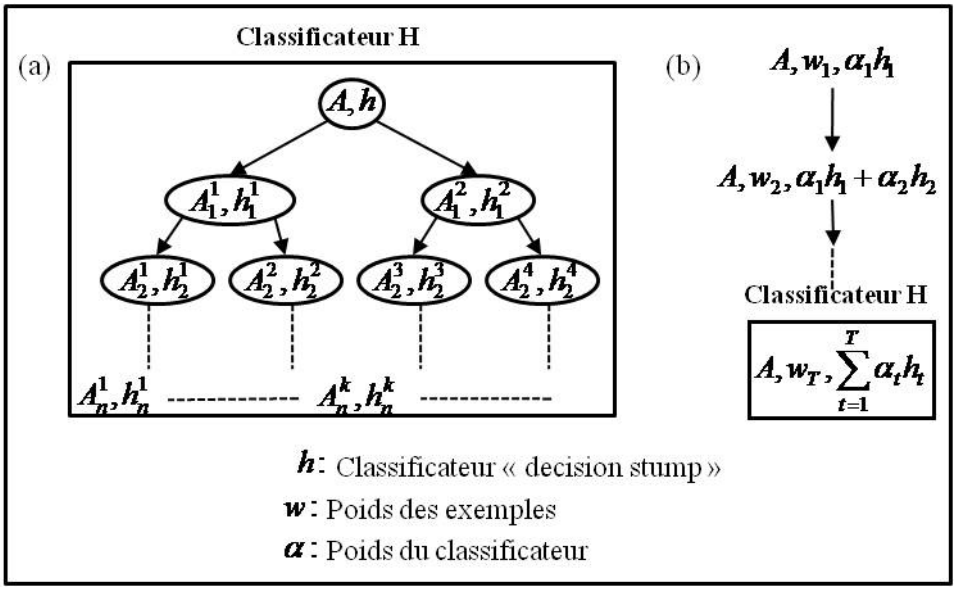

Figure 5.2 – Comment construire un classificateur H avec plusieurs seuils ? (a) Cas d'un arbre de décision (b) Cas du principe de l'algorithme d'*"AdaBoost"*

5.3 Sélection des classificateurs par algorithme génétique

Dans les sections précédentes, nous avons présenté les différents éléments de notre algorithme génétique. Nous allons les détailler et les étudier dans les sections suivantes.

5.3.1 Codage et initialisation

Dans l'algorithme génétique que nous avons développé, nous utilisons un codage binaire classique. Il consiste à coder chaque solution possible par une chaîne binaire de taille égale au nombre total de classificateurs. Un gène d'indice i a pour valeur 1 si le $i^{\text{ème}}$ classificateur de l'ensemble de départ est présent dans le sous-ensemble courant et 0 dans le cas contraire. Ce type de codage évite de sélectionner un classificateur plusieurs fois dans un même individu mais son inconvénient majeur est qu'il ne permet pas de contrôler le nombre de classificateurs sélectionnés. Nous notons par $C = (c_1, c_2, .., c_N)$ un chromosome où $c_i \in \{0, 1\}$ et S_c est un ensemble $S_c = \{i/c_i = 1\}$.

5.3.2 Fonction de $fitness$

L'objectif est de trouver un sous-ensemble de classificateurs le plus performant possible ayant un nombre réduit d'éléments. La fonction de $fitness$ définie dans notre algorithme génétique consiste à évaluer un individu de la population qui représente un sous-ensemble de classificateurs. Dans les méthodes *wrapper*, la fonction de *fitness* est liée à la construction d'un nouveau classificateur basé sur les caractéristiques impliquées dans l'individu. Pour contourner la lourdeur de cette approche, nous avons défini la fonction à minimiser comme l'erreur de classification d'une combinaison de classificateurs présents dans un individu de la population. Cette combinaison introduit un nouveau classificateur qui sera utilisé pour évaluer la qualité d'un sous-ensemble de classificateurs. Ce classificateur est défini par :

$$H^c = \boldsymbol{Comb}_{i \in S_c}(H_i) \tag{5.2}$$

où $\boldsymbol{Comb}_{i \in S_c}(H_i)$ est la combinaison de $|S_c|$ classificateurs présents dans un individu. Finalement la fonction de *fitness* est définie par l'erreur de classification de ce nouveau classificateur (H^c) :

$$fitness = error(H^c) \tag{5.3}$$

Nous présentons ci-dessous un exemple qui montre comment calculer la $fitness$ d'un individu (sous-ensemble de classificateurs *"AdaBoost"* appris pour chacune des caractéristiques initiales).
Soit un ensemble de classificateurs *"AdaBoost"* $H = \{H_1, H_2, .., H_{12}\}$ qui contient douze classificateurs où $H_i = \sum_{t=1}^{T} \alpha_{it} h_{it}$, $i = \{1, 2, 3, .., 12\}$ et T est le nombre d'itérations d'*"AdaBoost"*. Soit $I =$ "100110010110" un individu de la population. Pour cet individu, six classificateurs sur douze sont présents. Si on utilise la moyenne comme méthode de combinaison de classificateurs ($\boldsymbol{Comb}_{i \in S_c}(H_i) = \frac{1}{6} \sum_k (\sum_{t=1}^{T} \alpha_{kt} h_{kt})$, la fonction de $fitness$ de I sera calculée de la manière suivante :

$$fitness(I) = err(signe(\tfrac{1}{6} \textstyle\sum_k (\sum_{t=1}^{T} \alpha_{kt} h_{kt}))) \text{ où } k \in \{1, 4, 5, 8, 10, 11\}$$

L'opérateur de combinaison ($Comb$) peut prendre plusieurs formes que nous présentons et discutons dans la section suivante.

5.3.3 Combinaison de classificateurs

Pour la combinaison des classificateurs, nous avons utilisé les différentes méthodes de combinaison décrites dans la section 3.2.1 (La moyenne, la moyenne pondérée, le vote majoritaire, vote majoritaire pondéré, le médian et AWFO). Pour la méthode de combinaison par une moyenne pondérée, nous avons pondéré les réponses en utilisant le poids utilisé par l'algorithme d'*"AdaBoost"* pour pondérer les classificateurs faibles. Ce poids est donné par la

formule suivante :

$$W(H_i) = \frac{1}{2} ln \left(\frac{1 - erreur(H_i)}{erreur(H_i)} \right) \tag{5.4}$$

où $W(H_i)$ représente le poids de la $i^{\text{ème}}$ réponse d'un classificateur H et $erreur(H_i)$ est son erreur de classification.

En revanche, pour la méthode *AWFO* (Dujet et Vincent [1998]) nous proposons une adaptation pour qu'elle soit applicable dans notre cas. Dans la version initiale, il était toujours sous-entendu que l'ensemble de valeurs à agréger appartenait à un intervalle sur lequel la qualité des valeurs relativement à un objectif était monotone. Dans notre cas d'une classification à deux classes, les deux classes ont un statut équivalent, ne permettant pas de définir une valeur distinguée globale significative. Si nous cherchons à combiner des réponses de classificateurs, réponses comprises entre -1 et 1, -1 étant associé à un élément de la première classe et +1 caractérisant un élément appartenant à la seconde classe, nous disposons de deux valeurs "optimales". Les valeurs positives ou négatives n'ont pas le même propos, les unes indiquent avec plus ou moins de confiance une appartenance à une classe et les autres à une autre classe. Nous avons donc choisi d'agréger séparément les réponses positives et les réponses négatives. Ainsi, nous avons deux valeurs distinguées, -1 pour les valeurs négatives et +1 pour les valeurs positives (figure 5.3).

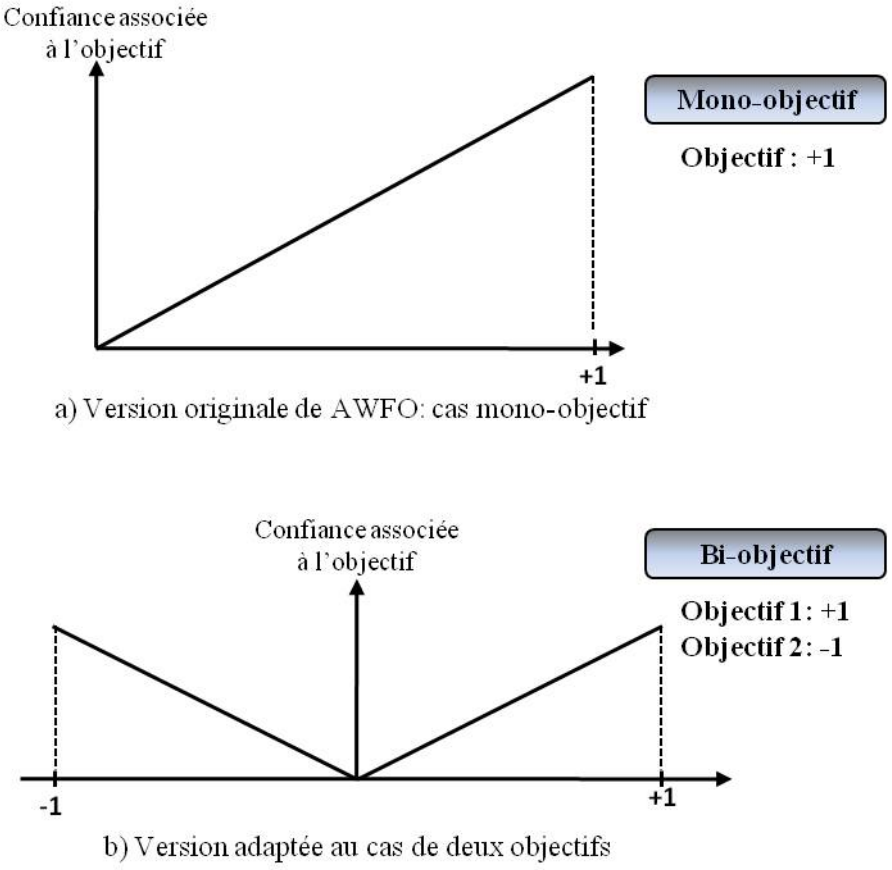

Figure 5.3 – Contexte (a)mono et (b) bi-objectif dans la méthode AWFO

La méthode *AWFO* propose de ne pas considérer exclusivement la réponse du classificateur mais aussi la distribution de toutes les réponses pour réaliser l'agrégation. En d'autres termes et si nous avons par exemple les réponses de dix classificateurs avec cinq réponses positives et les autres négatives, nous proposons de calculer un poids pour chaque réponse positive en prenant en compte les autres réponses positives et de calculer un poids pour chaque réponse négative en prenant en compte les autres réponses négatives. Le poids de chaque réponse est calculé par la formule suivante :

$$W(x_i) = \frac{d_{cum}(x_i)}{\sum\limits_{signe(x_j)=signe(x_i)} d_j} \tag{5.5}$$

$$\text{où} \begin{cases} d_{cum}(x_i) = \sum\limits_{j \in E_i} d_j \text{ avec } E_i = \{j/(d_j \geq d_i) \ \& \ (signe(x_j) = signe(x_i))\} \\ et \\ d_i = 1 - |x_i| \end{cases}$$

Le tableau 5.1 montre les détails d'un exemple décrivant la combinaison par la méthode *AWFO*. La première ligne représente les réponses brutes des classificateurs pour un unique exemple (échantillon). Après avoir trié les valeurs (deuxième ligne), nous calculons la distance de chaque valeur négative, respectivement positive, par rapport à la valeur optimale utilisée comme valeur distinguée -1, respectivement la valeur 1 (troisième ligne). Le poids de chaque réponse est égal à la somme cumulée (quatrième ligne) des distances des réponses qui ont une valeur moins favorable que la valeur courante, normalisée par la somme des distances à la valeur distinguée de l'ensemble des réponses de même signe. L'agrégation finale sera la moyenne des réponses pondérées par les poids calculés avec cette méthode (cinquième ligne).

Réponses initiales	-0.2	0.4	-0.5	-0.7	0.8	0.1	-0.9	0.6	0.5	-0.3
Réponses triées (x_i)	-0.9	-0.7	-0.5	-0.3	-0.2	0.1	0.4	0.5	0.6	0.8
Distance (d)	0.1	0.3	0.5	0.7	0.8	0.9	0.6	0.5	0.4	0.2
Dist cumulée (d_{cum})	2.4	2.3	2	1.5	0.8	0.9	1.5	2	2.4	2.6
Poids Final $(W(x_i))$	1	0.95	0.83	0.62	0.33	0.34	0.57	0.77	0.92	1

Table 5.1 – Exemple de combinaison par la méthode AWFO

Finalement, pour une réponse négative très proche de -1 et si nous avons beaucoup de réponses négatives la somme cumulée des distances devient plus élevée et son poids sera élevé. Pour une réponse très proche de 1, le même processus se produit.
Avant de présenter les études expérimentales faites pour valider notre méthode et pour analyser l'effet des choix des différents paramètres définis précédemment, nous avons voulu analyser l'apport que peut présenter l'utilisation des classificateurs associés au caractéristiques par rapport aux caractéristiques elles-mêmes

5.4 Classificateur *vs.* caractéristique

Dans la méthode proposée, nous utilisons les réponses d'un classificateur, appris sur chacune des caractéristiques, dans l'algorithme génétique. En d'autres termes, nous pouvons dire que nous remplaçons une caractéristique par la réponse d'un classificateur appris sur cette caractéristique. Des questions peuvent naturellement se poser : est ce que la réponse d'un classificateur construit sur une caractéristique peut remplacer cette caractéristique ? comment ce remplacement peut améliorer les résultats ?
Pour répondre à ces questions, nous avons fait des expérimentations en utilisant trois bases de données. Les trois bases de données utilisées sont décrites dans le tableau 5.2. Ces bases proviennent toutes de l'UCI Repository (Asuncion et Newman [2007]) et constituent des données relatives à des problèmes de classification supervisée fréquemment considérés dans la littérature du domaine de la classification.

	Nb carac	***Nb d'exemples***	***Apprentissage***	***Test***
Sonar	60	208	104	104
Ionosphere	34	702	351	351
Heart	22	267	150	117

Table 5.2 – Description de trois bases

Les deux premières colonnes du tableau 5.2 désignent respectivement le nombre de caractéristiques caractérisant chaque exemple et le nombre d'exemples disponibles. Nous donnons ensuite la taille respective des ensembles d'apprentissage et de test que nous utilisons.
Nous avons construit une base des réponses de classificateurs où chaque caractéristique est remplacée par la réponse d'un classificateur *"AdaBoost"*. Le tableau 5.3 montre le taux d'erreur d'un classificateur SVM pour les trois bases en utilisant comme descripteur d'une part les caractéristiques initiales et d'autre part les réponses du classificateur. Nous pouvons constater que nous avons eu les mêmes taux d'erreur pour les deux descripteurs (caractéristiques et réponses d'un classificateur).

	Sonar	***Ionosphere***	***Heart***
Caractéristiques	12.5	5.12	22.46
Réponses	12.5	5.12	22.46

Table 5.3 – Taux d'erreur obtenus à l'aide d'un SVM

Nous avons ensuite tiré aléatoirement des sous-ensembles de caractéristiques de différentes tailles et nous avons testé leur qualité à l'aide d'un classificateur SVM. Les réponses du classificateur qui correspondent aux caractéristiques de chaque sous-ensemble, représentent de leur côté des sous-ensembles de caractéristiques.
La figure 5.4 montre une comparaison entre l'histogramme des erreurs sur 100 sous-ensembles

aléatoires de caractéristiques de taille 40 de la base **Sonar** et celui des 100 sous-ensembles de réponses de classificateurs.

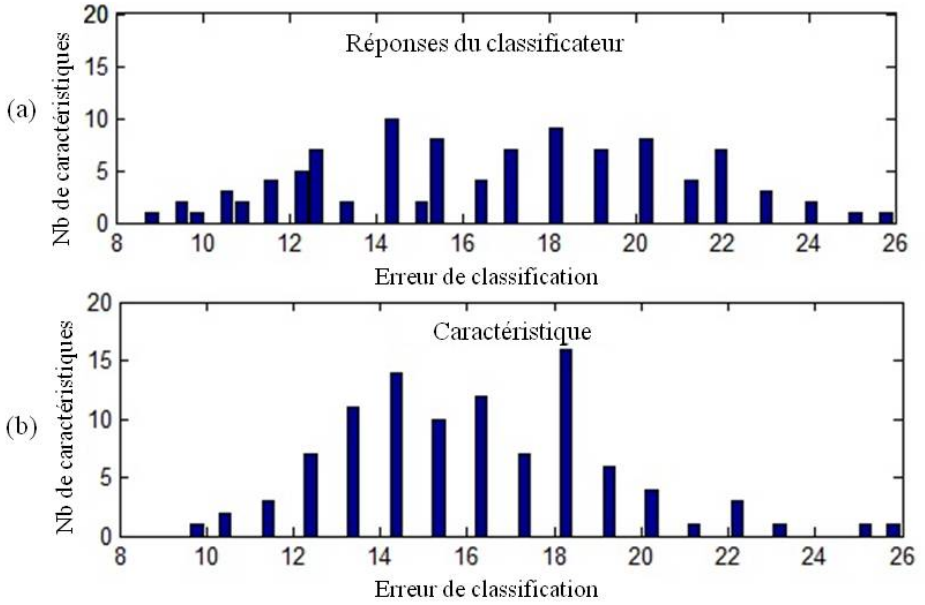

Figure 5.4 – Histogramme des erreurs (a) Réponses des classificateurs (b) Caractéristiques

A partir de ces histogrammes, nous pouvons constater que le meilleur sous-ensemble de réponses est meilleur que celui de caractéristiques. Par contre l'histogramme cumulé (figure 5.5) montre que généralement pour les sous-ensembles de bonne qualité, les réponses des classificateurs fournissent de meilleurs résultats que ceux obtenus avec les caractéristiques mais le phénomène s'inverse dans le cas des sous-ensembles plus mauvais.

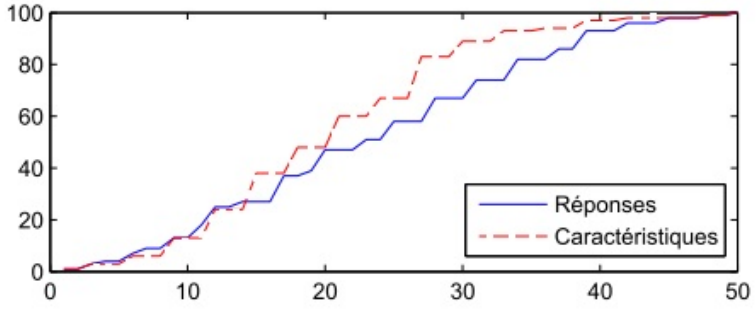

Figure 5.5 – Histogramme cumulé

Le tableau 5.4 résume les résultats d'un SVM pour les trois bases. Les résultats sont calculés sur des sous-ensembles tirés aléatoirement. Les tailles des sous-ensembles sont égales respectivement à 25%, 50%, 75% et 100% de la taille initiale de l'ensemble de caractéristiques.

Base	*Nb de carac*		*Taille des sous-ensembles*			
			25%	50%	75%	100%
Sonar	60	Réponses	10.57	7.69	**8.6**	12.5
		Caractéristiques	10.57	7.69	9.8	12.5
Ionosphere	34	Réponses	4.27	**3.41**	**3.41**	5.12
		Caractéristiques	4.27	4.27	4.27	5.12
Heart	22	Réponses	**26.7**	**23.5**	**22.46**	22.46
		Caractéristiques	27.27	26.73	22.99	22.46

Table 5.4 – Résultats obtenus sur les caractéristiques et sur les réponses des classificateurs

A partir de ce tableau, nous pouvons constater que les réponses des classificateurs fournissent toujours de meilleurs résultats que les caractéristiques initiales et cela pour les trois bases. Mais peut-on considérer ces résultats comme significatifs ? Est-ce qu'une généralisation de ce principe est possible ?
Pour répondre à ces questions nous avons étudié dans quel cas la réponse d'un classificateur peut être meilleure que la caractéristique elle même tout en travaillant dans un espace à une dimension. Dans ce but, et pour un problème bi-classe, nous avons distingué deux cas :

- Le cas idéal où les deux classes sont bien séparées
- Le cas où il y a des chevauchements entre les classes

Pour le cas idéal, et dans la figure 5.6, nous présentons deux exemples d'une caractéristique qui est capable de discriminer entre les deux classes. Pour le premier exemple, un seul seuil est suffisant pour résoudre le problème. Par contre pour le deuxième exemple il faut deux seuils. Dans les deux exemples, nous pouvons remarquer que les deux classes sont bien séparées.

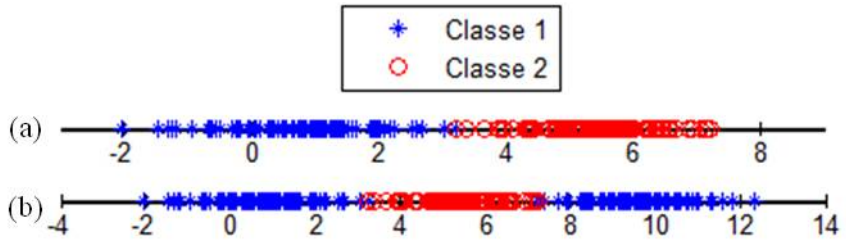

Figure 5.6 – Cas idéal : (a) Un seuil (b) Deux seuils

En remplaçant la caractéristique par la réponse du classificateur associé, nous avons constaté qu'une simplification au niveau des données est apparue. La figure 5.7 montre la variation des réponses de classificateur en fonction des valeurs initiales de la caractéristique. Nous pouvons remarquer sur cette figure, pour les deux cas définis précédemment, que la

résolution du problème est rendue plus simple puisque rendue possible avec un seul seuil. Le classificateur joue le rôle d'une transformation non-linéaire comme illustré sur cette figure ainsi que sur la figure 5.8 dans le cas où les classes se chevauchent.

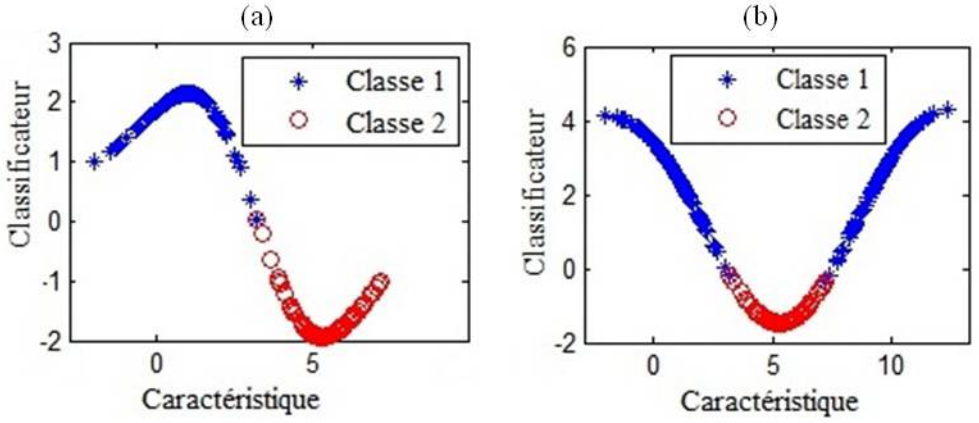

Figure 5.7 – Transformation non-linéaire de données à l'aide d'un classificateur (a) Un seuil (b) Deux seuils

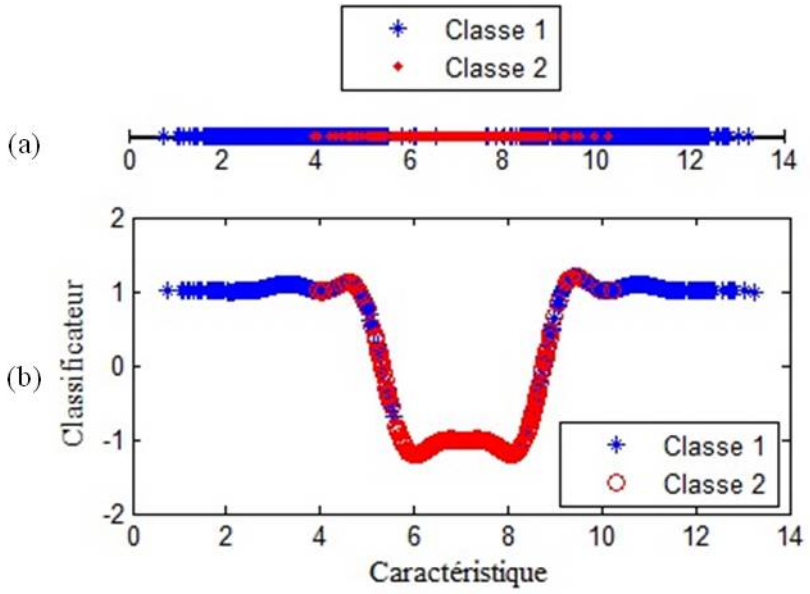

Figure 5.8 – Cas de chevauchement des classes (a) Caractéristique (b) Fonction de transformation par un classificateur

Afin de comparer objectivement une caractéristique f et la caractéristique associée f^c définie comme la réponse d'un classificateur simple, nous avons retenu un des critères utilisés dans les méthodes de type *"filter"*. Nous avons calculé le score de *Fisher* de f et f^c des exemples présentés dans les figures 5.6, 5.7 et 5.8. Les résultats figurent dans le tableau 5.5. Nous pouvons constater que les réponses du classificateur ont un score de *Fisher* plus élevé et cela dans tous les cas. Cela reflète une nette amélioration du pouvoir discriminatoire de la caractéristique.

	Cas ideal		***Cas de chevauchement***
	Un seuil	Deux seuils	
Caractéristiques	2.4	2.22	0.35
Réponses	3.58	2.85	1.15

Table 5.5 – Score de *Fisher* calculé sur les caractéristiques et sur les réponses des classificateurs

Cette transformation que nous introduisons n'est pas sans rappeler la transformation réalisée dans les méthodes d'extraction de caractéristiques où elle se fait au niveau de l'espace total des caractéristiques alors que nous nous limitons à une transformation dans un espace à une dimension. La transformation est liée aux données connues qui apportent une structuration particulière à la dimension considérée dans la caractéristique. L'espace se trouve alors étiré, aplati et replié selon la nature des données et des classes.

5.5 Expérimentations et validation

Dans cette section, nous présentons les expérimentations que nous avons réalisées pour valider notre méthode de sélection. Nous commençons par décrire les bases de données que nous avons utilisées ainsi que les descripteurs. Nous passons après à l'étude des paramètres de la méthode et surtout des paramètres de l'algorithme génétique. Nous présentons finalement les résultats de sélection par notre méthode et leur comparaison avec d'autres méthodes de la littérature.

5.5.1 Base de données

Pour nos expérimentations, nous avons utilisé la base *MNIST*. C'est une base de chiffres manuscrits (de 0 à 9) isolés, construite en 1998 (Lecun *et al.* [1998]). Chaque chiffre est associé à une image de taille 28 x 28 en 256 niveaux de gris (exemple de la figure 5.9). La base *MNIST* est divisée en deux sous-ensembles, un ensemble d'apprentissage de 60 000 exemples et un ensemble de tests de 10 000 exemples.

Figure 5.9 – Exemples d'images extraites de la base *MNIST*

5.5.1.1 Protocole d'expérimentation

Pour nos expérimentations, nous avons construit des bases des bases d'apprentissage, de validation et de test à partir de deux ensembles initiaux.

Nous traitons *a priori* des problèmes à deux classes. Dans le cas plus général de n classes il est nécessaire de construire des sous-ensembles dans l'ensemble d'exemples étiquetés qui permettent d'utiliser une approche *"un contre tous"*. Chaque sous-ensemble est associé à une classe. Soit $A = \{A_i\}$, $V = \{V_i\}$ et $T = \{T_i\}$ qui représentent respectivement les bases d'apprentissage, les bases de validation et les bases de test. A_i, V_i et T_i sont construits pour l'utilisation d'une méthode *"un contre tous"*. Chacune des bases A_i et T_i contient 2*N exemples, N exemples de la classe i et N exemples de toutes les autres classes. Par contre les V_i ne contiennent que N éléments ($\frac{N}{2}$ exemples de la classe i et $\frac{N}{2}$ exemples de toutes les autres classes). Pour la base *MNIST* nous avons $i \in \{0, 1, 2, .., 9\}$ et $N = 1000$.

5.5.1.2 Descripteurs

Nous avons utilisé plusieurs descripteurs de formes parmi lesquels, le descripteur de Fourier générique (GFD) (Zhang et Lu [2002]), la R-signature (Tabbone et Wendling [2003]) et le descripteur de Zernike (Kim *et al.* [2000]).

Le descripteur générique de Fourier (Generic Fourier Descriptor GFD) : ce descripteur a été proposé en 2002 (Zhang et Lu [2002]). Afin de garantir l'invariance à la rotation, les transformations de Fourier ont été construites en exprimant l'image et la transformée de Fourier dans le domaine polaire. La transformée de Fourier dans le domaine polaire est calculée par la formule suivante :

$$PF(\rho, \psi) = \sum_{r=0}^{R} \sum_{l=0}^{T} F(r, \theta_l) exp(2\pi(\frac{r}{R}\rho) + \frac{2\pi l}{T}\psi)$$

(ρ, ψ) (resp. (r, θ)) sont les coordonnées polaires de l'image (resp. du spectre de Fourier), $\theta_l = \frac{2\pi l}{T}$, R et T sont respectivement la résolution radiale et la résolution angulaire et $F(r, \theta_l)$ est le niveau de gris du pixel de coordonnées polaire (r, θ_l) dans l'image initiale.
L'ensemble de caractéristiques réelles associées à ce descripteur est défini par :

$$GFD = \left\{ \frac{|PF(0,0)|}{aire}, \quad \frac{|PF(0,1)|}{|PF(0,0)|}, \quad .. \quad , \frac{|PF(m,n)|}{|PF(0,0)|} \right\}$$

où aire est la mesure de la surface du disque englobant l'objet d'intérêt, m et n sont les fréquences radiale et angulaire maximum considérées.
Cet ensemble est transformé en un vecteur de caractéristiques qui constitue une représentation de chaque image de la base de données.

la *R*-signature : elle a été introduite en 2003 (Tabbone et Wendling [2003]). Ce descripteur est fondé sur la transformée de Radon. La transformée de Radon d'une fonction f, T_{Rf} est définie par :

$$T_{Rf}(\rho, \theta) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) \delta(xcos(\theta) + ysin(\theta) - \rho) dxdy$$

où $-\infty < \rho < \infty$, $0 \leq \theta < \pi$ et δ correspond à la distribution de Dirac.
A partir de la transformée de Radon, la R-signature est définie par :

$$R_f(\theta) = \int_{-\infty}^{\infty} T^2_{R^f}(\rho, \theta) d\rho$$

Le descripteur de Zernike : est un descripteur "pixels" basé sur les moments de Zernike. Ces moments sont construits (équation 5.6) à partir de polynômes complexes V_{mn} qui forment un ensemble orthogonal complet défini sur le disque unité,

$$A_{mn} = \frac{m+1}{\pi} \int_x \int_y I(x,y)[V_{mn}(x,y)]dxdy \tag{5.6}$$

Où m et n définissent l'ordre du moment, V_{mn} est un polynôme complexe (Zernike [1934]) dit polynôme de Zernike. I(x, y) est le niveau de gris d'un pixel de l'image I.
Les polynômes de Zernike V_{mn} sont exprimés en coordonnées polaires :

$$V_{mn}(x,y) = W(r,\theta) = R_{mn}(r)e^{-jn\theta} \tag{5.7}$$

Où R_{mn} est le polynôme radial orthogonal :

$$R_{mn}(r) = \sum_{s=0}^{\frac{m-|n|}{2}} (-1)^s \frac{(m-s)!}{s!\left(\frac{m+|n|}{2}-s\right)!\left(\frac{m-|n|}{2}-s\right)!} \tag{5.8}$$

$R_{mn}(r)$ est égale à 0 dans le cas où m et n n'ont pas la même parité.

5.5.2 Paramétrage de l'AG

Nous avons étudié plusieurs paramètres pour notre algorithme génétique parmi lesquels les paramètres liés à la population (initialisation, nombre d'individus, stabilité d'optimisation), le nombre de générations et la méthode de combinaison utilisée dans la fonction de $fitness$. Pour ce dernier aspect, l'étude de l'influence de la méthode de combinaison sur la fonction de *fitness* a été considérée à deux niveaux, d'une part au niveau de la qualité des caractéristiques sélectionnées, d'autre part au niveau des résultats de reconnaissance. Les résultats présentés dans la suite sont obtenus sur la base *MNIST* en utilisant la luminance des pixels comme descripteur (ceci permet d'envisager un grand nombre de caractéristiques) et des classificateurs *"AdaBoost"* comme classificateurs de base.

L'initialisation de la population consiste à générer N_{pop} individus. Chaque individu représente un sous-ensemble de classificateurs. En codage binaire, un individu est un vecteur binaire. Soient (a) la probabilité qu'un gène soit à '1' et $(1-a)$ la probabilité qu'un gène soit à '0'. Si a est faible, nous aurons une forte sélection et si a est important, peu de caractéristiques seront supprimées au bout d'un nombre raisonnable de générations.
Nous avons vérifié l'influence de la probabilité (a) sur les résultats de sélection (nombre

et qualité des classificateurs sélectionnés). La probabilité classique considérée dans les algorithmes génétiques utilisés pour la sélection dans les méthodes de type *"wrapper"* est de 0,5 ce qui signifie qu'il y a toujours environ 50% des gènes du chromosome à '1'. Dans nos expérimentations, nous avons étudié l'influence de cette probabilité en faisant varier (a). Les valeurs testées sont 0,25, 0,5 et 0,75.
Les trois courbes de la figure 5.10 montrent l'évolution du nombre de classificateurs présents dans le meilleur individu en fonction du nombre de générations.

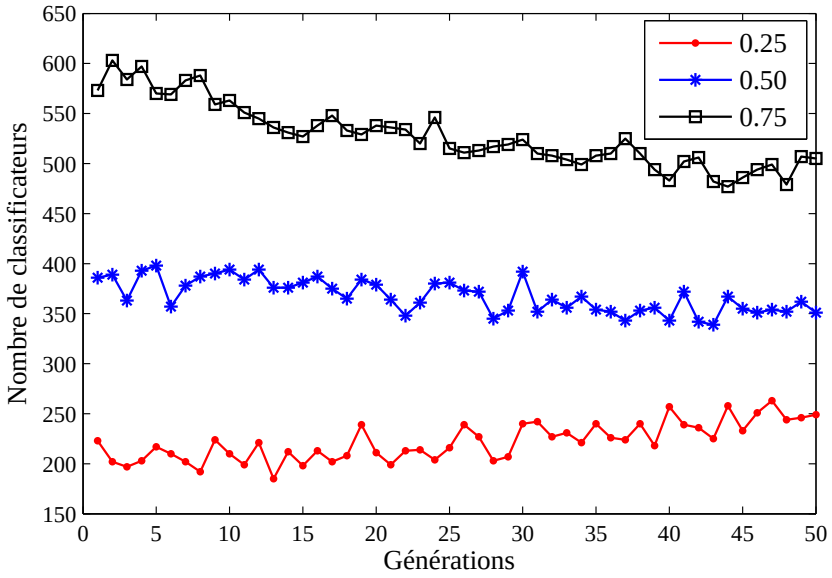

Figure 5.10 – Nombre de classificateurs à chaque génération pour différentes valeurs de a, paramètre utilisé pour l'initialisation de la première génération

Les résultats montrent l'influence de cette probabilité sur le nombre de classificateurs sélectionnés. Pour une probabilité de 0,25 pour '1', le nombre de classificateurs de la première génération est 200 mais il passe à 250 pour la dernière. Pour les deux autres probabilités, le nombre de classificateurs présents dans le meilleur individu de la population est toujours plus élevé que si nous utilisons une probabilité de 0,25, mais il décroit au cours des générations pendant le processus de stabilisation plus lent et donc les risques de sur-apprentissage est plus important.
La figure 5.11 montre l'erreur moyenne de tous les individus à chaque génération. Les trois courbes représentent les résultats en utilisant les différentes valeurs de probabilité. Cette figure montre que l'AG est capable de converger vers une solution optimale même si le nombre des gènes à '1' au départ est petit. De plus, nous voyons que l'erreur moyenne est plus faible, du moins au bout de 50 générations.
Nous avons bien montré que si le nombre de classificateurs dans chaque chromosome de la population initiale est faible, on pourra sélectionner un nombre réduit de classificateurs. Par ailleurs en combinant moins de classificateurs pour chaque individu, la fonction de $fitness$ sera plus rapide à calculer. Le tableau 5.6 montre le temps relatif de sélection en faisant

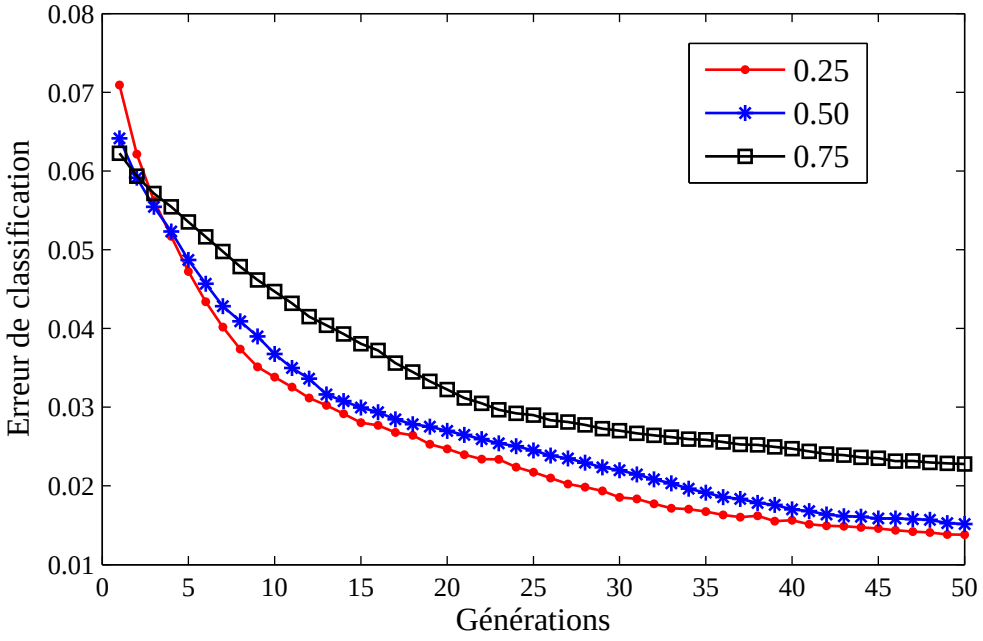

Figure 5.11 – Erreur moyenne par génération pour différentes valeurs de a

Probabilité de "1"	0.25	0.5	0.75
Temps relatif de sélection	0.19	0.38	1
Taux de reco	98.55	98.6	98.40

Table 5.6 – L'influence de a sur le temps de sélection

varier cette probabilité. Les résultats montrent que le processus de sélection devient plus de 80% plus rapide en diminuant le nombre de classificateurs présents dans un individu de $\frac{3N}{4}$ à $\frac{N}{4}$ où N est le nombre total de gènes d'un chromosome (nombre total de classificateurs).

Nombre d'individus : Le temps de traitement dépend du nombre d'individus considérés à chaque génération. Nous avons donc étudié l'influence de ce paramètre sur notre méthode. La figure 5.12 présente des résultats du taux de reconnaissance en prenant respectivement 50, 100, 200 et 500 comme taille de la population. Les résultats obtenus montrent que, pour les valeurs 100,200 et 500 l'AG converge vers des solutions d'une qualité similaire, indépendamment du nombre d'individus de la population. Par contre, si l'on diminue encore le nombre d'individus, l'efficacité de la méthode est moindre. Néanmoins, il est connu qu'il faut un grand nombre d'individus au départ pour assurer une bonne couverture de l'espace et trouver une solution acceptable. Le tableau 5.7 montre aussi l'influence de ce paramètre sur le temps de sélection. Il montre le temps relatif de sélection par l'algorithme génétique en faisant varier le nombre d'individus et nous pouvons remarquer qu'en passant de 500 individus à 100 individus, le processus de sélection est 75% plus rapide tout en gardant la même performance.

Stabilité de la sélection : Nous avons aussi étudié la stabilité du processus proposé en exécutant la sélection plusieurs fois avec les mêmes valeurs de paramètre. Il ne s'agit

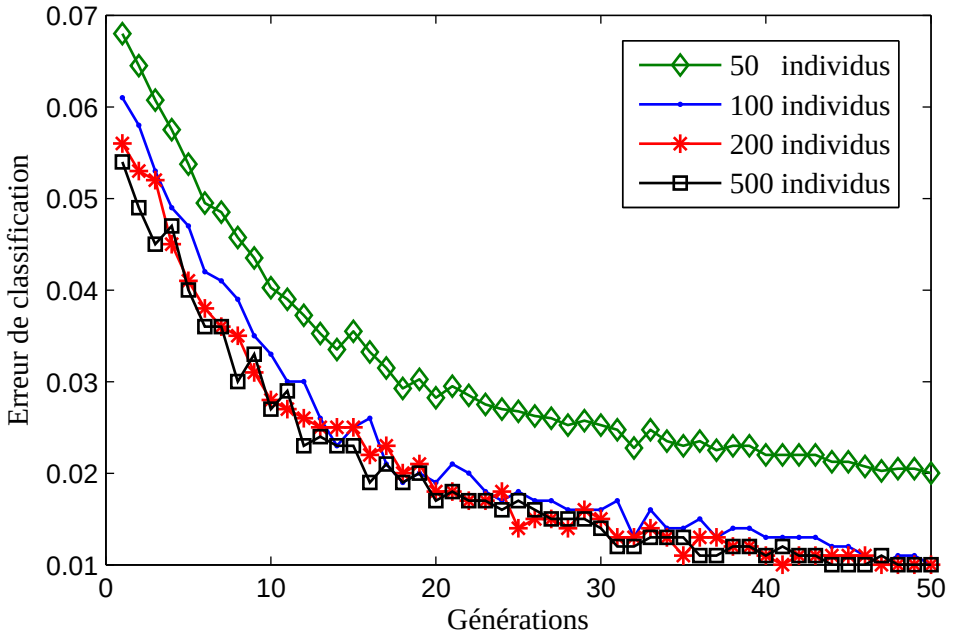

Figure 5.12 – L'influence du nombre d'individus sur l'erreur de classification du meilleur individu

Nombre d'individus	100	200	500
Temps relatif de sélection	0.23	0.44	1
Taux de reco	98.50	98.45	98.55

Table 5.7 – Influence du nombre d'individus sur le temps de sélection

pas de vérifier que les mêmes classificateurs ont été sélectionnés, mais de vérifier que l'algorithme génétique est capable de retrouver des sous-ensembles de classificateurs de qualité équivalente. La figure 5.13 montre les résultats de trois exécutions de la méthode de sélection. Les trois courbes représentent la qualité du meilleur individu de la population à chaque génération et ceci pour chaque exécution. Nous pouvons remarquer que les sous-ensembles finalement sélectionnés par l'algorithme génétique, dans les trois expériences, ont des qualités équivalentes.

La méthode de combinaison : la fonction de *fitness* que nous proposons est basée sur la combinaison d'un sous-ensemble de classificateurs. Nous avons étudié l'influence que peuvent avoir les méthodes de combinaison présentées dans la section 3.2.1, sur la fonction de *fitness* d'un côté et sur la qualité des caractéristiques sélectionnées de l'autre côté. Dans ce paragraphe, nous présentons les résultats de différentes méthodes de combinaison utilisées dans l'algorithme génétique et ses influences sur la fonction de *fitness*. Dans la section 5.5.3, nous aborderons le sujet de l'influence de ce choix sur la qualité des caractéristiques sélectionnées.

Nous avons bien vérifié qu'il n'est pas efficace d'utiliser le **min** et le **max** comme méthode de combinaison. En effet, il suffit qu'un seul classificateur, celui qui représente la valeur minimale (respectivement maximale), se trompe pour que le résultat de la combinaison

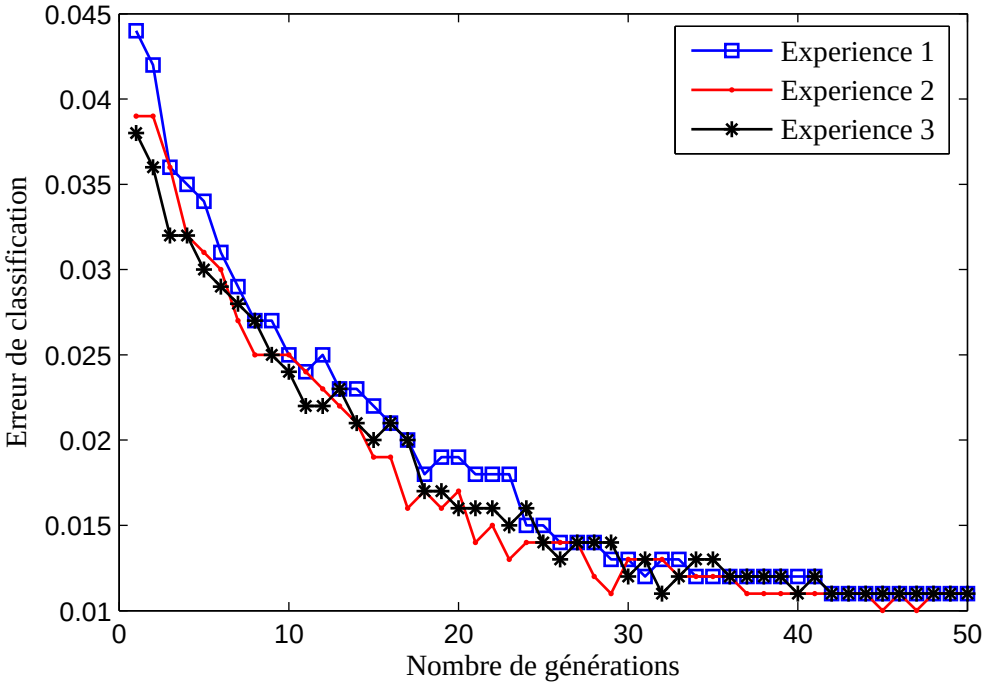

Figure 5.13 – Stabilité de l'AG

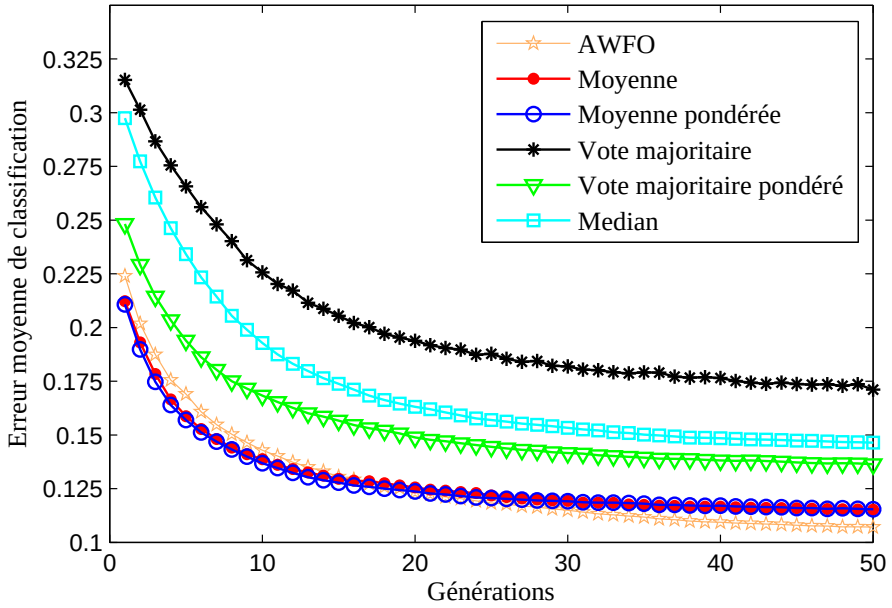

Figure 5.14 – Influence de la méthode de combinaison sur la *fitness* (66 caractéristiques du descripteur de*Zernike*)

devienne faux.

Les figures 5.14 et 5.15 montrent la moyenne de l'erreur moyenne de classification de la population au niveau de chaque génération sur les dix classes de la base *MNIST* en utilisant différentes méthodes de combinaison et différents descripteurs. Nous pouvons remarquer

que la moyenne et la moyenne pondérée sont plus performantes que le vote majoritaire, le vote majoritaire pondéré et le médian. Par contre, la méthode de combinaison *AWFO* avec l'adaptation proposée est la plus performante sur les différents types de descripteurs.

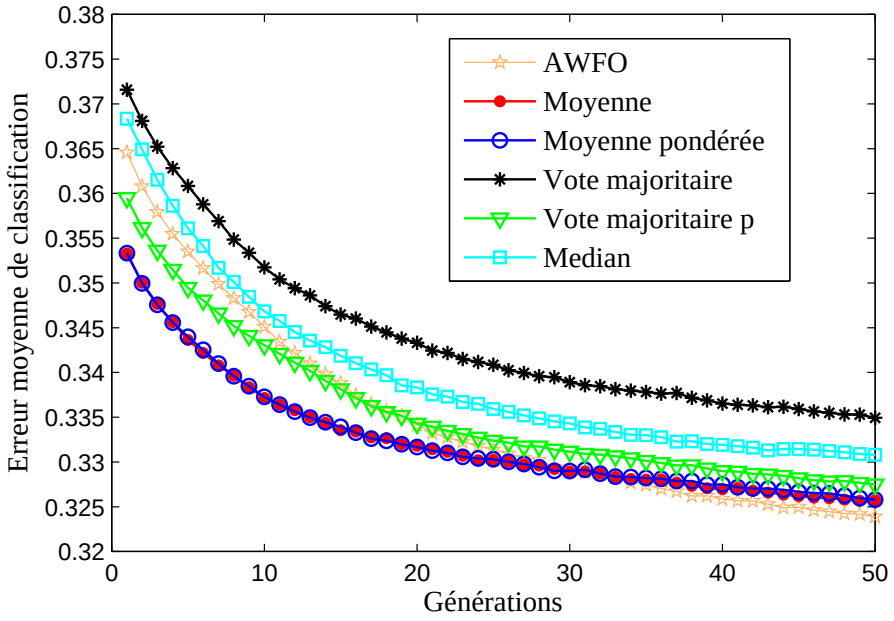

Figure 5.15 – Influence de la méthode de combinaison sur la *fitness* (180 caractéristiques du descripteur *R-signature*)

Choix du sous-ensemble final : pour sélectionner le sous-ensemble final de caractéristiques, nous avons défini trois stratégies :
- La première est la plus classique et consiste à prendre le meilleur individu de la population de la dernière génération. Le sous-ensemble final est donc composé des classificateurs présents dans cet individu.
- La deuxième est basée sur des statistiques calculées sur la population de la génération finale. Pour chaque classificateur, nous calculons sa fréquence d'apparition dans les individus de la population. Les S premiers classificateurs les plus fréquents représentent le sous-ensemble final où S est un nombre prédéfini par l'utilisateur.
- La troisième stratégie consiste à exécuter l'algorithme génétique *K-fois* et à sélectionner les S classificateurs les plus fréquents dans les K individus. Les K individus représentent les meilleurs individus obtenus par K applications de l'algorithme génétique.

Dans la suite, nous utiliserons respectivement les notations suivantes : *"Strat_MI"*, *"Strat_Popf"* et *"Strat_K-exec"* pour ces trois stratégies.

Critère d'arrêt : Le critère d'arrêt classique utilisé pour mettre fin à l'évolution de l'algorithme génétique est le nombre de générations. Nous l'avons utilisé comme critère d'arrêt ainsi qu'un autre critère basé sur la stabilité de la fonction de *fitness*. En d'autres termes, le processus s'arrête quand k itérations successives ont à peu près la même valeur de *fitness* sur le meilleur individu de la population. Le nombre de générations choisi est assez

faible (50 générations) pour que l'évolution n'ait pas le temps d'atteindre un apprentissage par coeur sur l'ensemble des exemples considérés.

5.5.3 Résultats

Dans cette section, nous présentons les résultats de sélection obtenus sur la base *MNIST* en utilisant les différents descripteurs définis dans la section 5.5.1.2. Le tableau 5.8 regroupe les tailles du vecteur de caractéristiques associé à chacun des descripteurs.

		Nombre de caractéristiques
GFD	R=8,T=12	96
	R=10,T=15	150
R-signature		180
Zernike		66
Pixels		784

Table 5.8 – Nombre de caractéristiques pour chaque descripteur utilisé pour la représentation de la base *MNIST*

Dans un premier temps, nous montrons les résultats obtenus par application de notre méthode de sélection sans procéder au préalable à une phase de présélection, puis nous présentons ensuite l'apport d'une phase de présélection et nous terminons par une comparaison entre les résultats de notre méthode et ceux obtenus par d'autres méthodes de sélection.

5.5.3.1 Résultat de la méthode sans présélection

Dans cette section, nous présentons les résultats obtenus sans passer par une phase de présélection des classificateurs. Nous avons appliqué notre méthode de sélection en gardant la totalité des caractéristiques initiales. Nous montrons les résultats obtenus sur la base *MNIST* en appliquant les trois stratégies de sélection que nous venons de définir ainsi que des classificateurs de différents types. La construction de l'ensemble initial des classificateurs simples est faite à l'aide d'un des classificateurs décrits dans la section 5.2. Le premier classificateur considéré est un classificateur *"AdaBoost"*. D'une part, il nous permet de trouver plusieurs seuils adaptés selon les exemples d'apprentissage, ceci constitue un avantage par rapport aux classificateurs basés sur un seul seuil. D'autre part, la réponse de ce classificateur est numérique et le signe indiquant la classe et le module constituant une sorte de confiance comprise entre 0 et 1. Ce format permet de mettre en œuvre différentes méthodes de combinaison ce qui n'est pas le cas pour les arbres de décision qui sont capables de trouver plusieurs seuils mais qui fournissent les étiquettes des classes comme réponse finale sans les accompagner d'une grandeur numérique, ce qui limite l'utilisation des méthodes de combinaison au vote majoritaire ou aux votes pondérés. Ensuite, nous

comparons les résultats obtenus avec l'algorithme d'AdaBoost et ceux obtenus avec les autres classificateurs.

5.5.3.1.a Cas d'un classificateur *"AdaBoost"*

Après la construction de l'ensemble de classificateurs simples à l'aide de l'algorithme d'*"AdaBoost"*, et après avoir sélectionné le meilleur sous-ensemble de classificateurs pour les différents descripteurs de la base MNIST, en utilisant les différentes stratégies de sélection, nous avons fait une étude expérimentale de l'influence de la méthode de combinaison sur la qualité des sous-ensembles sélectionnés ainsi que sur l'apport de notre méthode de sélection sur le temps d'apprentissage.

Cas *"Strat_MI"* : Dans ce cas, le meilleur individu de la dernière génération représente le sous-ensemble final de classificateurs sélectionnés et les caractéristiques associées à cet ensemble représenteront le sous-ensemble que nous utiliserons dans le système de classification finale. Le tableau 5.9 montre le nombre de caractéristiques sélectionnées pour chaque descripteur et pour chacune des dix classes de la base *MNIST*. Nous remarquons que les sous-ensembles finaux sont en moyenne 68,5% plus petit que l'ensemble initial.

	Zernike	***GFD_8×12***	***GFD_10×15***	***R-signature***	***Pixels***
Classe 0	18	33	43	40	278
Classe 1	28	36	47	30	217
Classe 2	18	27	53	34	261
Classe 3	27	26	43	32	240
Classe 4	27	31	46	52	240
Classe 5	24	23	39	50	250
Classe 6	28	29	44	62	258
Classe 7	24	37	53	46	239
Classe 8	26	27	38	29	228
Classe 9	25	33	51	38	239
Moyenne	25	30	46	42	245
Initial	66	96	150	180	784
% de réduction	62.1	68.75	69.33	76.66	68.75

Table 5.9 – Nombre de caractéristiques sélectionnées pour chaque descripteur utilisé pour la reconnaissance de chiffres

Pour évaluer la qualité des sous-ensembles trouvés par notre méthode, nous avons utilisé un classificateur SVM entraîné sur les bases d'apprentissage A_i et testé sur les bases T_i.
Le tableau 5.10 montre le taux de classification pour chaque descripteur et pour chaque

classe avant et après la sélection. Nous pouvons remarquer que les taux avant et après la sélection sont relativement proches et ceci quel que soit le descripteur.

	Zernike		***GFD_8×12***		***GFD_10×15***		***R-signature***		***Pixels***	
	Sans	Avec	Sans	Avec	Sans	Avec	Sans	Avec	Sans	Avec
Classe 0	98.5	98.55	97.5	97.4	97.5	97.6	65.5	75.25	98.8	98.6
Classe 1	98.25	98.55	98.25	98.25	98.1	98.3	85.6	88.35	98.7	98.55
Classe 2	85.8	85.35	92.45	92.55	91.8	92.25	77.7	77.8	97.8	97.85
Classe 3	91.1	90.8	88.4	89.25	88.5	88	73.25	73.85	96.8	96.65
Classe 4	94.45	94.55	92.2	91.4	91.7	91.6	83.4	86.95	97.55	97.5
Classe 5	89.6	89.15	92	91.9	90.7	91.25	77.25	82.7	98.25	98
Classe 6	91.15	90.85	93.9	93.2	94	93.2	65.7	66.35	98.6	98.55
Classe 7	93.75	93.05	91.9	92.15	91.2	91.2	85.35	84.35	97.8	97.9
Classe 8	93	92.65	88	87.75	87.4	87.25	77.7	80.35	95.4	95.2
Classe 9	89.1	88.95	89.15	89.5	88.8	89.1	68	75.9	97.6	97.2
Moyenne	92.47	92.25	92.38	92.34	91.97	92.08	75.95	79.19	97.73	97.6

Table 5.10 – Résultats d'un classificateur SVM sans et avec la sélection pour chaque descripteur utilisé pour la reconnaissance de chiffres

En conclusion, nous pouvons dire que nous avons réussi à sélectionner des sous-ensembles de caractéristiques 68,5% plus petits que la taille des ensembles initiaux mais ayant à peu près la même qualité. Nous avons étudié l'apport de la sélection sur le temps d'apprentissage en utilisant toujours un classificateur SVM. Le tableau 5.11 compare les temps d'apprentissage en utilisant la totalité des caractéristiques d'un côté et le sous-ensemble sélectionné par notre méthode. Nous remarquons que le processus d'apprentissage devient entre deux et cinq fois plus rapide.

	Zernike	***GFD_8×12***	***GFD_10×15***	***R-signature***	***Pixels***
Sans sélection	0.11	0.15	0.12	0.25	1
Avec sélection	0.05	0.055	0.06	0.075	0.2
Facteur d'amélioration	2.2	2.8	2	3.3	5

Table 5.11 – Temps relatif d'apprentissage d'un *SVM* avec et sans sélection

Finalement nous avons comparé les résultats de sélection en utilisant les différentes méthodes de combinaison. Le tableau 5.12 montre les résultats de comparaison en faisant la moyenne sur les dix classes. Les résultats confirment la conclusion que nous avons tirée sur les méthodes de combinaison et ses apports sur la fonction de *fitness*. Cette conclusion est valable sur la qualité des caractéristiques sélectionnées et dans ce tableau, nous remarquons que les résultats pour les trois méthodes de combinaison *AWFO, moyenne et moyenne pondérée* sont proches pour les différents descripteurs utilisés.

	Zernike	*GFD_8×12*	*GFD_10×15*	*R-signature*	*Pixels*	*Moyenne*
AWFO	92.25	92.34	92.08	79.19	97.6	90.74
Moyenne	92.42	91.9	91.95	79.04	97.4	90.54
Moyenne pondérée	92.28	92.55	92.1	79.55	97.55	90.76
Vote majoritaire	91.71	90.55	91.65	77.38	96.8	89.61
Vote pondéré	91.95	91.95	90.98	79.05	97.2	90.22
Médian	92.14	91.06	92.05	78.25	97.5	90.20
Sans sélection	92.47	92.38	91.97	75.95	97.73	90.1

Table 5.12 – Comparaison de différentes méthodes de combinaison (*"Strat_MI"*)

Cas *"Strat_Popf"* : Pour que nous puissions comparer les résultats de ce cas avec les résultats du cas précédent, nous avons sélectionné le même nombre de classificateurs. Mais cette fois-ci en prenant les classificateurs les plus fréquents dans les individus de la population de la génération finale.
Le tableau 5.13 montre les résultats d'un classificateur *SVM* sur les sous-ensembles sélectionnés à partir de différents descripteurs en utilisant cette stratégie et pour les différentes méthodes de combinaison. Les résultats montrent que le sous-ensemble associé au meilleur individu est plus performant.

	Zernike	*GFD_8×12*	*GFD_10×15*	*R-signature*	*Pixels*	*Moyenne*
AWFO	85.75	86.75	86.65	72.65	96.9	85.74
Moyenne	86.22	86.34	85.12	73.7	96.69	85.61
Moy pondérée	86.12	87.84	86.45	74.3	97.1	86.36
Vote majoritaire	83.28	79.54	80.5	72.05	94.8	82.03
Vote pondérée	86.81	86.95	86.6	73.24	96.2	85.91
Médian	85.24	83.9	81.86	73.66	95.86	84.10
Sans sélection	92.47	92.38	91.97	75.95	97.73	90.10

Table 5.13 – Comparaison entre les différentes méthodes de combinaison (*"Strat_popf"*)

Cas *"Strat_K-exec"* : Comme dans le cas précédent, nous avons sélectionné le même nombre de classificateurs que dans le cas *"Stra_MI"*. Mais cette fois-ci en prenant les classificateurs les plus fréquents dans les K individus qui regroupent les meilleurs individus de la dernière génération obtenue dans K exécutions de l'algorithme génétique.
Le tableau 5.14 montre les résultats d'un classificateur *SVM* sur les sous-ensembles sélectionnés de différents descripteurs en utilisant cette stratégie et pour les différentes méthodes de combinaison. Les résultats sont proches des résultats obtenus en ne considérant qu'une seule exécution de l'algorithme génétique. Ceci montre la stabilité de la méthode.

	Zernike	*GFD_8×12*	*GFD_10×15*	*R-signature*	*Pixels*	*Moyenne*
AWFO	91.88	91.46	91.43	77.11	97.45	89.86
Moyenne	92.02	91.33	91.48	77.22	97.1	89.83
Moy pondérée	91.83	91.14	90.55	78.08	97.55	89.83
Vote majoritaire	91.01	90.88	89.84	75.69	96.15	88.51
Vote pondérée	91.64	91.05	89.12	76.4	97.3	89.10
Médian	91.72	90.95	90.81	75.45	97.16	89.21
Sans sélection	92.47	92.38	91.97	75.95	97.73	90.10

Table 5.14 – Comparaison de différentes méthodes de combinaison (*"Strat_K-exec"*)

Les résultats présentés jusqu'à présent représente les meilleurs résultats obtenus sur cinq exécutions de notre méthode de sélection. Le tableau 5.15 montre la moyenne des résultats de cinq exécutions ainsi que l'écart-type en utilisant le *AWFO* comme méthode de combinaison. Nous pouvons constater que l'écart type est relativement bas et ce qui montre la stabilité de notre méthode.

	Zernike	*GFD_8×12*	*GFD_10×15*	*R-signature*	*Pixels*
Meilleur	92.25	92.34	92.08	79.19	97.6
Moyenne	92.08	92.21	91.98	78.89	97.45
Écart-type	0.14	0.11	0.11	0.19	0.09
Sans sélection	92.47	92.38	91.97	75.95	97.73

Table 5.15 – Stabilité de la sélection

5.5.3.1.b Cas d'autres classificateurs

Nous avons comparé les résultats obtenus à partir de l'ensemble de classificateurs *"AdaBoost"* avec ceux basés sur les mêmes descripteurs mais des ensembles de classificateurs d'un autre type. Pour les deux classificateurs simples *"decision stump"* et *"Classif_Alamdari"*, nous avons testé les différentes méthodes de combinaison. Pour les arbres de décisions, nous n'avons utilisé que le vote majoritaire et le vote majoritaire pondéré comme méthode de combinaison (*cf* section 5.5.3.1).

	Zernike	*GFD_8×12*	*GFD_10×15*	*R-signature*	*Pixels*
AdaBoost	92.42	92.55	92.1	79.55	97.6
classif_Alamdari	91.5	90.15	90.85	74.15	93.85
decison_stump	92.05	92.05	91.95	79.25	97.45
Arbre de décision	91.95	92.17	91.85	79.35	97.5

Table 5.16 – Comparaison des résultats basés sur différents ensembles de classificateurs

Le tableau 5.16 montre les meilleurs résultats obtenus pour chacun des ensembles de

classificateurs et chacun des descripteurs. Les résultats montrent que la sélection à partir d'un ensemble de classificateurs *"AdaBoost"* est meilleur que d'autres ensembles de classificateurs. Les résultats sur un ensemble de classificateurs de type *"arbres de décisions"* sont proches de ceux d'*"AdaBoost"* mais la possibilité d'utiliser plusieurs méthodes de combinaison avec *"AdaBoost"* rend ces classificateurs plus avantageux.

5.5.3.2 Résultat de la méthode avec pré-sélection

Dans cette section, nous présentons les résultats de notre méthode de sélection après une phase de pré-sélection. Cette phase permet d'éliminer les caractéristiques qui sont supposées très mauvaises avant de lancer notre algorithme génétique. N'importe quelle méthode de filtrage peut être appliquée dans cette phase. La figure 5.16 montre un exemple de comparaison de trois scores calculés sur les vingt premières caractéristiques du descripteur de *Zernike*. Les trois scores utilisés sont le score de Fisher (2.3), le coefficient de corrélation (2.2) et le taux de classification d'un classificateur appris sur une seule caractéristique.

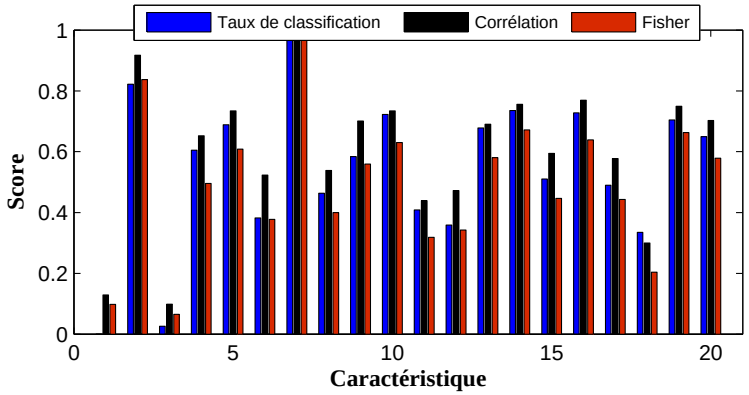

Figure 5.16 – Exemple de scores des caractéristiques

Nous pouvons remarquer que les trois scores sont à peu près équivalents. Mais puisque nous utilisons une base des classificateurs appris sur chacune des caractéristiques dans la phase suivante, nous avons utilisé la performance de chacun des classificateurs comme score de présélection.

		Zernike	*GFD_8×12*	*GFD_10×15*	*R-signature*	*Pixels*
10%	***AWFO***	91.25	91.71	92.08	79.98	97.25
	Moyenne	91.37	92.02	92.01	79.82	97.35
	Moyenne pondérée	91.05	91.34	91.85	79.75	97.15
20%	***AWFO***	91.65	91.79	91.99	79.28	97.05
	Moyenne	90.75	91.66	91.57	78.36	97.25
	Moyenne pondérée	90.35	90.34	91.34	79.24	97.15
50%	***AWFO***	90.05	89.94	89.19	76.15	96.8
	Moyenne	89.07	89.28	89.45	76.55	96.55
	Moyenne pondérée	89.89	89.15	88.25	76.3	96.35
Meilleurs résultats sans présélection		92.42	92.34	92.1	79.55	97.6

Table 5.17 – Résultats de sélection après une pré-sélection

Le tableau 5.17 montre les résultats de sélection pour tous les descripteurs après une phase de pré-sélection. Nous avons étudié l'influence de la pré-sélection en éliminant 10, 20 et 50 % des plus mauvaises caractéristiques selon leur taux d'erreur individuel. Nous comparons donc les résultats de chaque cas de pré-sélection en utilisant différentes méthodes de combinaison dans la phase de sélection. Pour la plupart des descripteurs nous pouvons remarquer que plus on élimine de caractéristiques avant la sélection, plus le taux de classification se dégrade. Même avec un taux de pré-sélection très bas nous avons eu des résultats très proches de ceux que nous avions eu sans pré-sélection. Néanmoins, les résultats montrent que de mauvaises caractéristiques peuvent parfois interagir positivement et peuvent avoir une influence sur la qualité finale du sous-ensemble de caractéristiques. Nous pouvons remarquer aussi qu'avec une pré-sélection de 50% pour le descripteur *Pixel*, les résultats ne sont pas loin des meilleurs résultats que nous avons eu sans la pré-sélection ce qui montre que la pré-sélection devient de plus en plus utile quand nous travaillons avec des descripteurs de très grande taille.

5.5.3.3 Comparaison avec d'autres méthodes

Nous avons comparé notre méthode de sélection avec trois autres méthodes existantes. Ces méthodes reposent sur différentes approches d'évaluation (*"filter"* et *"wrapper"*). Les trois méthodes considérées sont *Relief* (section 2.4), *SAC* (section 2.7) et la troisième est une méthode de *"wrapper"* classique basée sur une recherche aléatoire en utilisant le même algorithme génétique que notre méthode, avec les mêmes paramètres mais la fonction de *fitness* est définie par l'erreur d'un classificateur *SVM*. Dans les paragraphes suivants, nous utilisons le nom *"Wrapper"* pour designer cette méthode.
Dans la suite, nous présentons les résultats détaillés pour les dix classes de la base *MNIST* et le descripteur *Zernike*. Pour le reste des descripteurs nous présentons uniquement la

moyenne des résultats sur les dix classes.
Le tableau 5.18 montre les taux de classification d'un classifcateur SVM sur les sous-ensembles de caractéristiques sélectionnés par les différentes méthodes de sélection citées ci-dessous ainsi que par notre méthode sur le descripteur *Zernike*. Pour les deux méthodes *Relief*, nous avons sélectionné le même nombre de caractéristiques que celui déterminé par notre méthode. Pour la méthode *Wrapper*, nous avons laissé l'algorithme génétique sélectionner le meilleur sous-ensemble et finalement pour SAC le nombre de caractéristiques est contrôlé par l'équation 2.8 (*cf.* section 2.7).

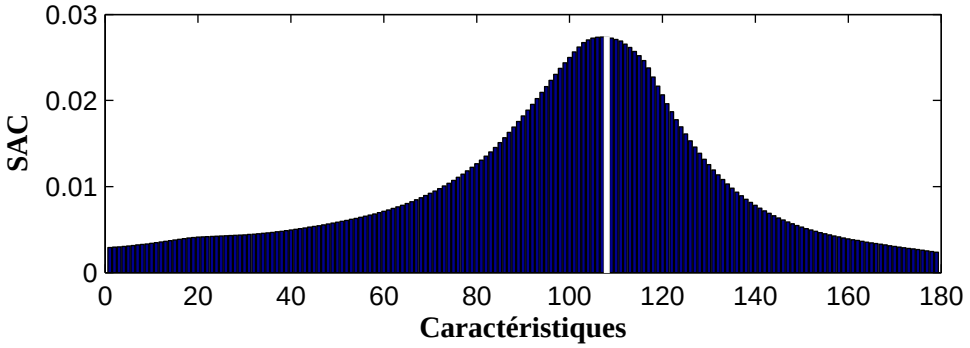

Figure 5.17 – Exemple de sélection par la méthode SAC

La figure 5.17 montre un exemple de sélection sur la *classe 0* de la base MNIST en utilisant le descripteur *R-signature* par la méthode *SAC*. Cette méthode consiste à classer les caractéristiques selon leur performance (évaluée par un classificateur). Le score de Fisher est calculé sur les taux de classification des caractéristiques en ajoutant à chaque fois la meilleure caractéristique au sous-ensemble courant.

	Relief	***Wrapper***	***SAC***	***Notre méthode***
Classe 0	96.8	98.6	98.2	98.55
Classe 1	95.4	98.2	97.9	98.55
Classe 2	81.15	86.3	83.05	85.35
Classe 3	88.95	91.35	89.85	90.8
Classe 4	91.85	94.65	93.55	94.55
Classe 5	87.95	90.2	88.05	89.15
Classe 6	89.8	91.05	89.85	90.85
Classe 7	90.15	93.25	92.45	92.65
Classe 8	91.25	93.45	90.25	93.05
Classe 9	85.15	88.75	87.65	88.15
Moyenne	89.85	92.61	91.11	92.25

Table 5.18 – Comparaison détaillée avec d'autres méthodes de sélection à partir du descripteur *Zernike*

Le sous-ensemble final est représenté par celui qui maximise le score de Fisher. Sur la figure, nous pouvons remarquer que le score de *Fisher* maximum est atteint quand le nombre de caractéristiques est égal à 108 (la barre blanche) ce qui correspond au nombre de caractéristiques à sélectionner.
Pour les autres descripteurs, les résultats sont présentés dans le tableau 5.19. Nous pouvons remarquer que notre méthode est nettement meilleure que les méthodes *Relief* et *SAC*. Par contre, nos résultats sont très proches de ceux obtenus par la méthode *Wrapper* et ceci pour tous les descripteurs (tableau 5.19)

	Relief	***Wrapper***	***SAC***	***Notre méthode***
GFD_8×12	89.95	92.55	91.11	92.55
GFD_10×15	90.15	92.01	91.45	92.1
R-signature	73.55	79.95	75.88	79.55
Pixels	95.85	97.68	96.35	97.6

Table 5.19 – Comparaison entre différentes méthodes pour chaque descripteur

Le tableau 5.20 compare la taille moyenne des sous-ensembles sélectionnés par notre méthode et les deux méthodes *SAC* et *Wrapper* sur les dix classes. Nous pouvons remarquer que notre méthode est capable de sélectionner des sous-ensembles de caractéristiques de taille plus petite mais néanmoins avec une qualité meilleure que la méthode *SAC* et une qualité très proche de la méthode *Wrapper*.

	Zernike	***GFD_8×12***	***GFD_10×15***	***R-signature***	***Pixels***
Notre méthode	25	30	46	42	245
SAC	30	31	48	96	280
Wrapper	36	52	65	79	299

Table 5.20 – Comparaison des tailles des sous-ensembles finaux de caractéristiques

Par ailleurs, nous avons calculé le temps de sélection de notre méthode et de cette dernière méthode *Wrapper* et ceci pour tous les descripteurs. Le tableau 5.21 montre les résultats et nous pouvons remarquer que notre méthode est 127 fois plus rapide dans le pire des cas et 250 fois dans le meilleur des cas pour le descripteur *Pixels*.

	Zernike	***GFD_8×12***	***GFD_10×15***	***R-signature***	***Pixels***
Notre méthode	0.001	0.0015	0.0022	0.0026	0.004
Wrapper	0.13	0.22	0.28	0.36	1
Facteur d'amélioration	130	146	127	139	250

Table 5.21 – Comparaison des temps de sélection relatifs de notre méthode et la méthode *Wrapper*

5.6 Conclusion

Nous avons présenté dans ce chapitre une nouvelle méthode rapide de sélection de caractéristiques. Cette méthode est basée sur une sélection de classificateurs simples construits sur chacune des caractéristiques. Plusieurs classificateurs simples ont été étudiés afin de trouver le classificateur le plus performant sur une seule caractéristique. Une optimisation par un algorithme génétique afin de trouver la meilleure combinaison des classificateurs est proposée et différentes méthodes de combinaison sont utilisées et adaptées à notre problème. Des expérimentations sur la base de chiffres manuscrits MNIST ont été faites en utilisant différents descripteurs de différentes tailles. Une large évaluation de la nouvelle méthode a donc été effectuée ainsi que des comparaisons avec d'autres méthodes de la littérature. Cette évaluation confirme la performance de la méthode ainsi que sa rapidité par rapport aux méthodes de *Wrapper*.

Chapitre 6

Analyse des choix des différents éléments de l'AG et conséquences

Dans ce chapitre, nous présentons une étude de plusieurs paramètres qui définissent l'algorithme génétique auquel nous avons eu recours dans la méthode développée dans le chapitre précédent. Dans le chapitre précédent, le nombre de caractéristiques sélectionnées fait partie des résultats de l'algorithme. Dans ce chapitre, nous proposons une méthode où le contrôle du nombre de caractéristiques à sélectionner est laissé à l'utilisateur, ceci grâce à l'introduction d'un codage entier des caractéristiques. Ensuite, nous abordons le sujet de la redondance des caractéristiques en étudiant la diversité d'un ensemble de caractéristiques et nous proposons d'intégrer cette mesure dans la fonction de *fitness* afin de limiter la redondance. Cette intégration nécessite d'utiliser des méthodes d'optimisation multi-objectifs (méthodes d'agrégation et méthodes de *Pareto*). Une base artificielle est utilisée pour mesurer la pertinence de l'intégration de la mesure de diversité ainsi que la base MNIST. Finalement, nous présentons une méthodologie pour la hiérarchisation des caractéristiques aussi bien au niveau des descripteurs qu'au niveau des caractéristiques.

6.1 Changement de codage

Le codage binaire utilisé dans le chapitre précédent ne permet pas de contrôler le nombre de classificateurs à sélectionner. Pour limiter ce problème, nous proposons d'utiliser un codage entier. Dans le codage binaire, chaque chromosome est de taille N où N est un nombre total de classificateurs. Nous adoptons cette représentation à l'aide d'une chaîne non binaire de taille prédéfinie notée L qui représente la taille de l'ensemble de classificateurs à construire par sélection. Chaque gène porte une valeur comprise entre 1 et N correspondant au numéro du classificateur dans l'ensemble de départ. Cette représentation garantit une taille constante des solutions durant le processus d'évolution. En revanche, elle ne permet pas de gérer directement les apparitions multiples d'un même classificateur dans la construction des sous-ensembles. Pour résoudre ce problème, pour un individu, nous

remplaçons la répétition de classificateurs par un classificateur tiré aléatoirement dans l'ensemble des classificateurs absents de l'individu pour assurer finalement la sélection de L classificateurs distincts. A titre d'exemple, si nous avons un ensemble de six classificateurs $H = \{H_1, H_2, .., H_6\}$ et un individu $I =$ "100110" de la population codé en binaire. Cet individu devient $I =$ "1 4 5" en passant en codage entier.
Nous avons utilisé les mêmes paramètres que dans l'algorithme génétique de base. Évidemment, en changeant le codage binaire nous ne considérons plus des algorithmes génétiques mais des algorithmes évolutionnaire en général. Pour le croisement, le principe est identique au principe appliqué dans le codage binaire et la figure 6.1 illustre un exemple de croisement de deux individus codés avec le nouveau codage.

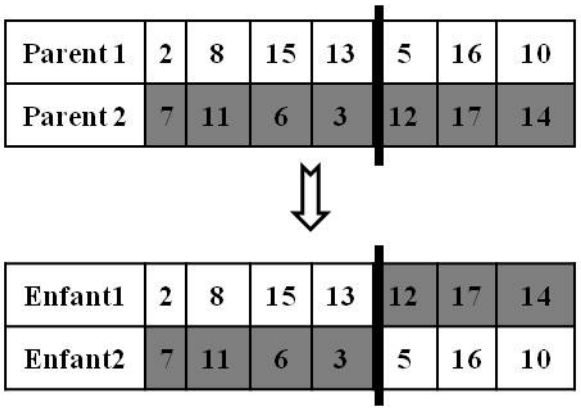

Figure 6.1 – Exemple de croisement pour le codage entier

Pour la mutation, un gène sera remplacé par une valeur aléatoire entre 1 et le nombre total de classificateurs (N). Dans les deux cas avant la validation, le cas d'éventuels doublons est traité.
Pour des problèmes bien précis, l'utilisateur peut, en utilisant ce type de codage, sélectionner un nombre défini de caractéristiques. Un autre avantage de ce type de codage est qu'il nous permet de tester des sous-ensembles de toute taille et ainsi d'appliquer une méthode plus systématique afin de trouver un sous-ensemble optimal.
Nous avons testé ce codage sur quelques descripteurs présentés précédemment pour la reconnaissance des chiffres de la base MNIST avec plusieurs valeurs de L. Nous avons donc sélectionné pour chaque descripteur des sous-ensembles de taille $L = 1, 2, .., N$ où N représente le nombre de caractéristiques du descripteur.
La figure 6.2 montre le taux de reconnaissance d'un classificateur SVM de la deuxième classe de la base MNIST en fonction du nombre de caractéristiques retenues en utilisant les sous-ensembles de caractéristiques du descripteur *R-signature*. La sélection est réalisée par notre méthode en utilisant le codage entier. Sur la même figure, nous indiquons le résultat du meilleur sous-ensemble sélectionné en utilisant le codage binaire (étoile rouge) et le codage entier (rectangle bleu). La courbe présente des irrégularités et globalement une vaste zone où les résultats sont assez stables, entre 25 et 110 caractéristiques.

Le tableau 6.1 montre les résultats moyens sur les dix classes de la base MNIST calculés sur les trois descripteurs (*Zernike, GFD_8×12 et R-signature*). Pour le codage entier, nous présentons le résultat du meilleur sous-ensemble parmi les $|Desc|$ sélectionnés par notre méthode en utilisant différentes méthodes de combinaison. Nous comparons aussi ces résultats avec ceux obtenus en utilisant un codage binaire (résultats présentés dans le chapitre précédent) ainsi que les résultats sans sélection.

	Méthode de combinaison	***Zernike***		***GFD_8×12***		***R-signature***	
		Nb_carac	Taux	Nb_carac	Taux	Nb_carac	Taux
Codage entier	AWFO	29	**92.53**	39	92.46	38	79.98
	Moyenne	27	92.49	36	**92.56**	39	80.01
	Moy_pondérée	31	92.46	35	92.52	40	**80.41**
Codage binaire	Meilleure	25	92.42	30	92.34	42	79.55
Sans sélection	-	66	92.47	96	92.38	180	75.95

Table 6.1 – Résultats après le changement de codage

Les résultats du codage entier sont très proches de ceux du codage binaire et cela pour toutes les méthodes de combinaison. Ils sont un peu meilleurs mais en sélectionnant un plus grand nombre de caractéristiques.

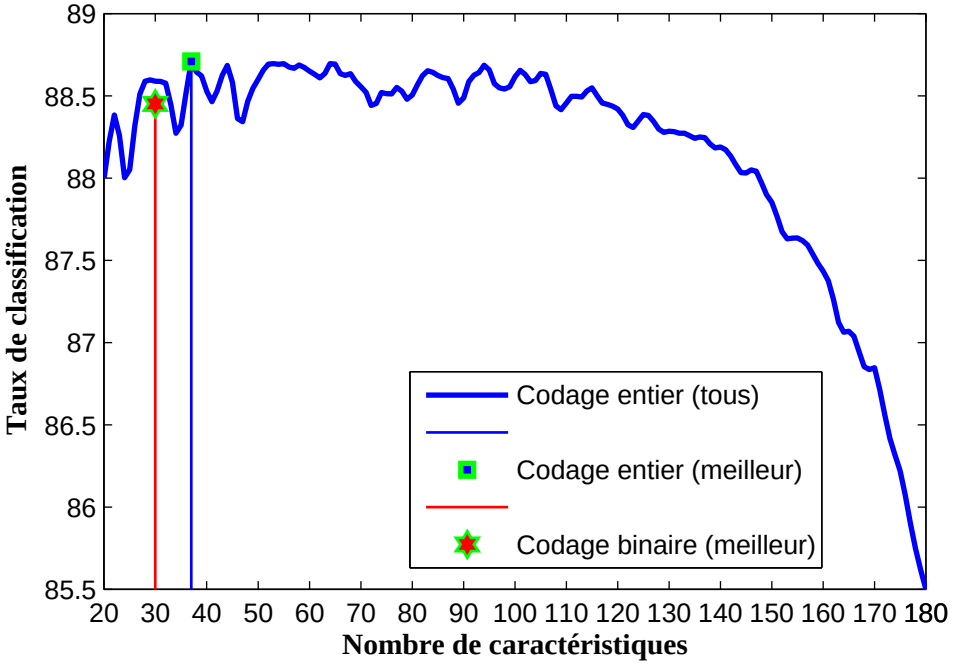

Figure 6.2 – Comparaison des deux types de codage

Nous avons comparé la qualité des sous-ensembles de caractéristiques obtenus par notre

méthode en utilisant le codage binaire et celle des sous-ensembles construits en utilisant la méthode classique *SFS* (section 2.1.5.1). Nous nous sommes pour cela basé sur une évaluation individuelle des caractéristiques par le score de Fisher (équation 2.3) ou la corrélation (équation 2.2). Après avoir calculé le score de chacune des caractéristiques, les sous-ensembles de caractéristiques sont construits en prenant la meilleure caractéristique, les deux meilleures, les trois meilleures, etc.

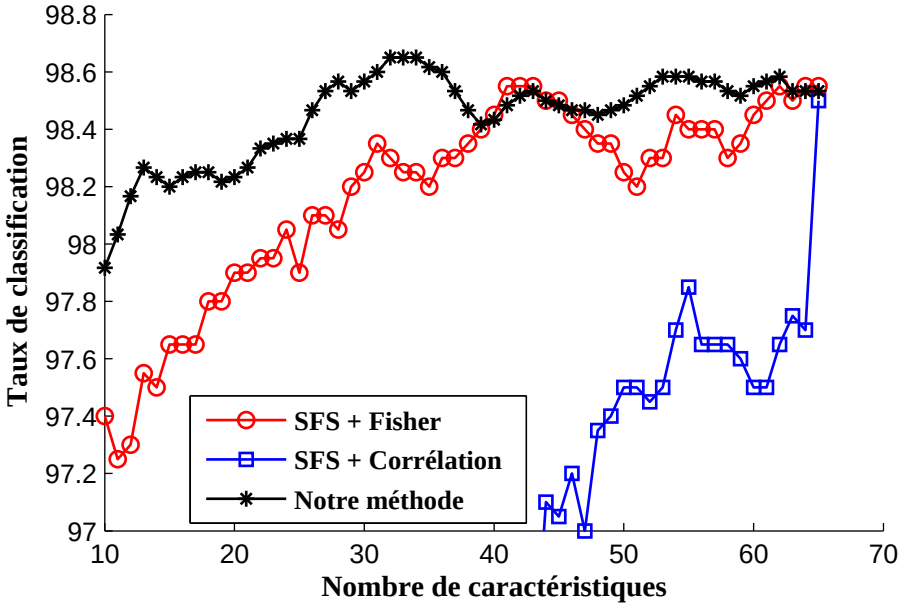

Figure 6.3 – Comparaison avec la méthode SFS

La figure 6.3 montre un exemple de comparaison de résultats calculés avec un ***SVM*** sur le descripteur de ***Zernike*** d'une des classes de la base *MNIST*. Cette comparaison montre que notre méthode réussit, même avec des sous-ensembles de petite taille, à trouver des sous-ensembles de caractéristiques qui sont plus pertinents que ceux construits par les deux autres méthodes ***SFS_Fisher*** et ***SFS_Corrélation***. La comparaison détaillée est montrée dans le tableau 6.2. Dans ce tableau, nous remarquons que notre méthode est meilleure que les deux autres et aussi qu'elle est capable de sélectionner des sous-ensembles de taille 35% plus petite en moyenne.

	Zernike		***GFD_8×12***		***R-signature***	
	Nb_carac	Taux	Nb_carac	Taux	Nb_carac	Taux
Codage entier	29	92.53	36	92.56	40	80.41
SFS_Fisher	58	92.47	74	92.38	79	78.88
SFS_Corrélation	55	92.47	71	92.4	75	77.54
Sans sélection	66	92.47	96	92.38	180	75.95

Table 6.2 – Comparaison des taux de reconnaissance avec ceux obtenus par la méthode *SFS*

6.2 Redondance de classificateurs

La fonction de *fitness* que nous avons utilisée, a pour objectif de trouver la combinaison de classificateurs qui minimise l'erreur sur une base de validation. En effectuant cette minimisation, nous risquons de sélectionner des classificateurs redondants qui ont la même qualité et qui réussissent à bien classer les mêmes exemples. Pour limiter la redondance dans le sous-ensemble sélectionné, nous avons étudié la diversité entre les classificateurs présents dans chacun des sous-ensembles de la population de l'algorithme génétique.

6.2.1 Diversité de classificateurs

Les mesures qui quantifient la diversité d'un ensemble de classificateurs sont classées en deux groupes : le premier consiste à évaluer la différence entre toutes les paires de classificateurs en évaluant la distance entre deux classificateurs en fonction de leur désaccord sur les mêmes exemples. Les mesures de ce groupe sont appelées *pairwise*. Le deuxième groupe est constitué des mesures globales qui consistent à quantifier la diversité d'une manière globale. Ce type de mesure risque d'ignorer les disparités internes importantes que peuvent présenter deux classificateurs.
De manière à quantifier le désaccord entre deux classificateurs et ainsi définir différentes mesures de diversité, une abstraction des sorties des classificateurs peut être utilisée, en se ramenant à un niveau binaire appelé niveau oracle. Pour un exemple X_j donné, nous cherchons uniquement à savoir si le classificateur Cl_i détermine la bonne classe ou si l'étiquette attribuée est incorrecte :

$$Cl_i(X_j) = \begin{cases} 1 & si \quad Cl_i(X_j) = Y_j \\ 0 & si \quad Cl_i(X_j) \neq Y_j \end{cases} \tag{6.1}$$

Il est toujours possible de se ramener au niveau oracle quel que soit le nombre de classes. A ce niveau, une matrice d'incidence peut être définie (tableau 6.3). Cette matrice permet d'évaluer les probabilités de désaccord entre deux classificateurs C_i et C_j.

	$Cl_j(X)$ est correcte	$Cl_j(X)$ est incorrecte
$Cl_i(X)$ est correcte	a	b
$Cl_i(X) = 0$ est incorrecte	c	d

Table 6.3 – Matrice d'incidence définie pour deux classificateurs au niveau oracle

Les termes de la matrice correspondent aux probabilités pour lesquelles Cl_i et Cl_j sont en accord ou en désaccord avec la classe attribuée aux mêmes exemples, avec $a+b+c+d = 1$. Ainsi, les entrées a et d représentent les exemples pour lesquels les deux classificateurs répondent par la même étiquette. Plusieurs mesures peuvent être calculées sur cette matrice pour deux classificateurs donnés.
Une mesure appelée *Q Statistic* a été définie dans (Yule [1900]). Pour deux classificateurs

Cl_i et Cl_j, cette mesure est calculée de la manière suivante :

$$Q_{ij} = \frac{ad - bc}{ad + bc} \tag{6.2}$$

En général, Q varie entre -1 et 1. Deux classificateurs qui réussissent à classer correctement les mêmes exemples, ont une valeur Q positive. Finalement, pour deux classificateurs statistiquement indépendants, $Q_{i,j}$ est égale à 0.
Une autre mesure qui permet d'évaluer le désaccord entre deux classificateurs et qui se base sur la matrice d'incidence a été proposée par (Fleiss *et al.* [2003]). Cette mesure est appelée κ. Pour deux classificateurs, l'expression de κ est donnée par :

$$\kappa_{i,j} = \frac{2(ac - bd)}{\sqrt{(a+b)(c+d) + (a+c)(b+d)}} \tag{6.3}$$

Une valeur faible de $\kappa_{i,j}$ signifie un grand désaccord entre Cl_i et Cl_j, et par conséquent une grande diversité.
Des mesures globales sur l'ensemble de classificateurs ont été définies. Parmi ces mesures est celle définie dans (Kuncheva [2004]). Cette mesure, notée KW, est définie par :

$$KW = \frac{1}{ML^2} \sum_{j=1}^{M} H(X_j) \times (L - H(X_j)) \tag{6.4}$$

où M, L représentent respectivement le nombre d'exemple de la base et le nombre total de classificateurs. $H(X_j)$ représente le nombre de classificateurs ayant correctement étiqueté l'exemple X_j. En se basant sur KW, une autre définition de κ a été proposée pour calculer globalement la diversité d'un ensemble. Soit p_{moy} la précision individuelle moyenne des classificateurs de l'ensemble. Le κ s'exprime alors :

$$\kappa = 1 - \frac{\frac{1}{L} \sum_{j=1}^{M} H(X_j) \times (L - H(X_j))}{N(L-1) \times p_{moy} \times (1 - p_{moy})} \tag{6.5}$$

Partridge et Krzanowskion ont proposé une mesure appelée **"DG"** (**D**iversité **G**énéralisée) (Krzanowski et Partridge [1996]), cette mesure est définie par :

$$DG = 1 - \frac{p(2)}{p(1)} \tag{6.6}$$

avec $p(1) = \sum_{i=1}^{L} \frac{i}{L} p_i$ et $p(2) = \sum_{i=1}^{L} \frac{i}{L} \frac{(i-1)}{L-1} p_i$
où le terme p_i correspond à la probabilité tel que i classificateurs choisis aléatoirement dans l'ensemble classent de manière incorrecte un exemple donné.
Finalement, la mesure de diversité globale d'un ensemble de classificateurs peut être toujours calculée par la moyenne des mesures de l'ensemble des paires de classificateurs possibles. Notons $DIV_{i,j}$ la mesure de diversité calculée pour deux classificateurs Cl_i et Cl_j

et L la taille de l'ensemble de classificateurs. La diversité globale de l'ensemble est définie par :

$$Div_{glob} = \frac{2}{L(L-1)} \sum_{1=1}^{L-1} \sum_{j=i+1}^{L} Div(Cl_i, Cl_j) \tag{6.7}$$

D'autres mesures de diversité sont disponibles dans (Kuncheva et Whitaker [2001],Kuncheva [2004])

6.2.2 Expérimentations

Dans le but d'éliminer la redondance, nous avons intégré une mesure de diversité dans notre méthode. Deux mesures ont été utilisées, la première est une mesure globale de diversité (équation 6.4) et l'autre est une moyenne des mesures calculées sur toutes les paires possibles de classificateurs (équation 6.3). La maximisation de la diversité dans le but de diminuer la redondance, doit être intégrée comme un objectif à optimiser par notre algorithme génétique. Pour optimiser ce dernier objectif en même temps que notre objectif de base qui est la minimisation de l'erreur de classification, nous avons utilisé un algorithme génétique multi-objectifs. Nous avons testé l'influence de la mesure de diversité sur la redondance des classificateurs et sa capacité à éviter de choisir un ensemble contenant des classificateurs par une phase d'expérimentations. Dans ces expérimentations, nous avons utilisé une base artificielle construite dans le but de mesurer la capacité de notre approche à enlever la redondance. Finalement, nous avons retesté notre méthode sur la base *MNIST* après les changements proposés.

6.2.2.1 Base artificielle

Dans le but de mesurer l'influence de l'intégration de la mesure de diversité sur l'élimination de la redondance, nous avons utilisé une base artificielle dans un problème à deux classes (Krizek *et al.* [2007]). Cette base est composée de 48 caractéristiques. Les caractéristiques sont construites comme suivant :
Parmi les 48 caractéristiques, les 13 premières sont authentiques, les 7 suivantes sont du bruit et les autres se déduisent des précédentes.
Pour les 20 premières, les échantillons (exemples de la base) sont calculés en suivant une distribution normale de probabilité caractérisée par une matrice de covariance Σ et un vecteur moyen μ. Les éléments de la première classe sont calculés selon la loi $N(\Sigma, \mu)$ par contre, les éléments de l'autre classe sont calculés selon la loi $N(\Sigma, -\mu)$. Le vecteur moyen, de dimension 48, et la matrice de covariance sont décomposés en plusieurs blocs :

Le premier bloc contient des caractéristiques statistiquement indépendantes qui ont le même pouvoir discriminant où :

$$\Sigma^{1,..,3} = I_3 \text{ et } \mu^{1,..,3} = \begin{bmatrix} 0.635 & 0.635 & 0.635 \end{bmatrix}^\top$$

avec I_n est la matrice identité de taille $n \times n$ où $n \in \mathbb{N}$.

Le deuxième bloc est constitué d'une paire de caractéristiques non indépendantes qui sont intégrées dans un bloc de taille 3.

$$\Sigma^{4,..,6} = \begin{bmatrix} 1.05 & 0.48 & 0.95 \\ 0.48 & 1.0 & 0.20 \\ 0.95 & 0.20 & 1.05 \end{bmatrix} \text{ et } \mu^{4,..,6} = \begin{bmatrix} 0.5 \\ 0.4 \\ 0 \end{bmatrix}$$

Le troisième bloc contient des caractéristiques statistiquement indépendantes avec une dégradation du pouvoir discriminant de chacune des caractéristiques avec :

$$\Sigma^{7,..,13} = I_7 \text{ et } \mu^{7,..,13} = \begin{bmatrix} 0.636 & 0.546 & 0.455 & 0.364 & 0.273 & 0.182 & 0.091 \end{bmatrix}^\top$$

Le dernier bloc ne contient que du bruit avec les paramètres :

$$\Sigma^{14,..,20} = I_7 \text{ et } \mu^{14,..,20} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^\top$$

La figure 6.4 résume la matrice de covariance pour les trois blocs de caractéristiques.

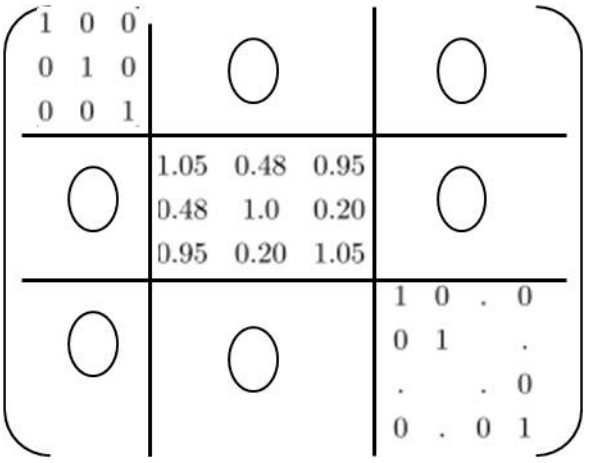

Figure 6.4 – Matrice générale de covariance pour les trois blocs

Le reste des caractéristiques(21 .. 48) sont des duplications des autres caractéristiques et certaines sont dupliquées plus qu'une fois.

6.2.2.2 Optimisation multi-objectifs

Pour optimiser nos deux objectifs (l'erreur de classification et la diversité), nous avons utilisé deux méthodes d'optimisation par un algorithme génétique multiobjetcifs. La fonction de *fitness* de ce nouvel algorithme génétique peut être définie comme une agrégation des objectifs (section 4.2.1) ou par les méthodes de *pareto* (section 4.2.2). Nous envisageons les deux approches.

6.2.2.3 Agrégation des objectifs

Comme nous l'avons vu dans la section 4.2.1, et avec deux objectifs, la fonction de *fitness* peut être définie comme une combinaison linéaire des objectifs. Chaque objectif est pondéré par un poids. La nouvelle fonction de *fitness* sera :

$$fitness = \alpha * (erreur(comb(H))) + (1 - \alpha) * (1 - diversité) \tag{6.8}$$

où α ($0 \leq \alpha \leq 1$) est un paramètre de pondération qui permet d'attribuer un facteur d'importance à chacun des deux objectifs. Une valeur de α supérieure à 0,5 poussera la recherche génétique vers des solutions de haute précision. Inversement, utiliser de petites valeurs de α donne plus d'importance à la maximisation de la diversité. Le tableau 6.4 montre les résultats calculés sur la base artificielle. Nous comparons dans ce tableau les résultats de la sélection par notre méthode avec et sans intégration de la diversité. Plusieurs valeurs de α ont été testées afin de trouver empiriquement la meilleure valeur. Nous pouvons noter que plus nous donnons d'importance à la diversité, plus nous risquons de détériorer la qualité des caractéristiques sélectionnées. Pour une valeur de α égale à 0,5, la méthode a permis de trouver un ensemble de caractéristiques de taille minimale mais en gardant un peu de bruit, la qualité du sous-ensemble sélectionné n'est pas satisfaisante. Une augmentation de la valeur de α améliore la qualité mais il faut toujours donner un peu de poids à la diversité pour éliminer la redondance.

				Caractéristiques				
				Initiale	Bruit	Redondance	Totale	Taux de classif
Optimale				13	0	0	13	90.7
Sans sélection				13	7	28	48	85.05
Avec sélection	Sans diversité			11	0	15	26	89.85
	Div_globale	α	0.5	4	3	5	12	82.5
			0.75	8	2	6	16	83.75
			0.85	7	1	9	17	85.5
			0.95	8	0	12	20	**89.85**
	Div_pairwise	α	0.5	5	1	5	11	80.5
			0.75	7	1	9	17	85.75
			0.85	9	0	10	19	88.5
			0.95	8	0	13	21	**89.95**

Table 6.4 – Influence de la diversité sur les résultats de sélection pour différentes valeur de α

Finalement, pour trouver un compromis entre ces deux objectifs sans passer par une combinaison linéaire pondérée des objectifs, nous avons utilisé les techniques de *Pareto*.

6.2.2.4 Pareto-optimal

Pour la méthode d'optimisation par la technique de *Pareto*, nous avons utilisé l'algorithme NSGAII (section 4.2.2.4). La figure 6.5 montre un exemple de *front Pareto* calculé sur la base artificielle. Cette figure montre les solutions de la dernière génération proposées par l'algorithme génétique NSGAII ainsi que la meilleure solution choisie (rectangle rouge).

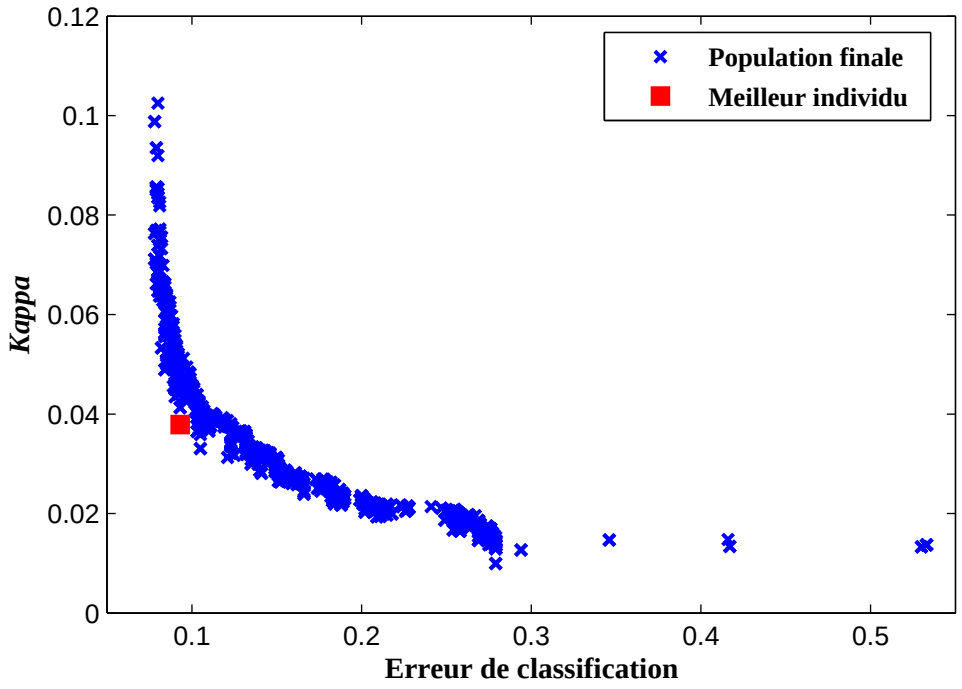

Figure 6.5 – Font *Pareto* avec les deux objectifs : erreur de classification et diversité

Nous avons donc utilisé cette technique pour sélectionner les meilleurs sous-ensembles de caractéristiques de la base artificielle en optimisant nos deux objectifs et nous montrons les résultats obtenus dans le tableau 6.5.

		Nombre de caractéristiques				
		Initiale	Bruit	Redondance	Totale	*Taux de classif*
Optimale		13	0	0	13	90.70
Sans sélection		13	7	28	48	85.05
Avec sélection	Sans diversité	11	0	15	26	89.85
	Avec *div_globale*	10	0	7	17	89.90
	Avec *div_pairewise*	9	0	6	15	90.20

Table 6.5 – Influence de la diversité sur les résultats de sélection en considérant une méthode multi-objectifs résolue par la méthode de Pareto

Dans ce tableau, les résultats de sélection sans intégrer les mesures de diversité montrent que notre méthode de base est capable d'enlever les bruits mais garde des informations du-

pliquées et redondantes (13 caractéristiques). En comparant ces résultats avec ceux obtenus en intégrant les deux mesures de diversité, nous pouvons remarquer qu'avec une mesure de diversité globale (*div_globale*), la méthode a réussi à sélectionner 17 caractéristiques au total. Dix caractéristiques parmi les 13 de base ont été sélectionnées et les trois qui restent ont été sélectionnées parmi les caractéristiques redondantes et donc il n'y a que quatre caractéristiques redondantes qui sont présentes dans le sous-ensemble final. Pour l'autre mesure de diversité $\kappa_{i,j}$ (*div_pairwise*), la méthode a réussi à enlever deux caractéristiques de plus et finalement à ne garder que deux caractéristiques redondantes. Finalement, nous pouvons remarquer que l'introduction de la diversité peut aider à réduire la redondance.

6.2.3 Résultats sur la Base MNIST

Nous avons retesté notre méthode après l'intégration de la mesure de diversité sur la base MNIST en utilisant la technique de *Pareto*. Le tableau 6.6 montre ces résultats. Dans ce tableau, nous pouvons constater que l'influence de la mesure de la diversité que nous avons ajoutée, a été minime pour les trois descripteurs *Zernike, GFD et R-signature*. Par contre sur le descripteur *pixel*, cette mesure a réussi à enlever 6% de caractéristiques de plus. La définition de chacun de descripteurs peut expliquer cette influence, décomposition d'un signal dans une base. Pour le descripteur *pixel*, la redondance des caractéristiques est toujours présente et ce n'est pas le cas des autres descripteurs. Quand la modélisation d'une image est réalisée par des chaînes ou des champs de Markov, l'hypothèse d'ordre 1 est généralement admise.

	Zernike		***GFD_8×12***		***R-signature***		***Pixels***	
	Nb_carac	Taux	Nb_carac	Taux	Nb_carac	Taux	Nb_carac	Taux
Sans diversité	29	92.53	36	92.56	40	80.41	245	97.6
Avec div_globale	25	92.55	32	92.53	39	80.31	207	97.55
Avec div_pairwise	23	92.58	29	92.56	40	80.45	201	97.6
Sans sélection	66	92.47	96	92.38	180	75.95	784	97.73

Table 6.6 – Résultats de notre méthode sur la base MNIST après l'intégration de la diversité

6.3 Sélection hiérarchique

La sélection hiérarchique de caractéristiques peut être définie comme une sélection sur plusieurs niveaux. Chaque niveau fournit un sous-ensemble de caractéristiques sélectionné à partir du sous-ensemble du niveau supérieur. Cette selection pourra être appliquée sur un ensemble de descripteurs ou sur les caractéristiques d'un seul descripteur.

6.3.1 Sélection hiérarchique sur un seul descripteur

Pour réaliser des sous-ensembles emboîtés de caractéristiques ou, en d'autres termes, pour réaliser une hiérarchisation des caractéristiques en fonction de l'efficacité de l'ensemble construit, nous proposons d'appliquer notre méthode de sélection plusieurs fois successivement. La sélection, à chaque niveau, se fait sur le sous-ensemble sélectionné au niveau précédent. La probabilité du nombre de "1" (nombre de classificateurs) présents dans un individu joue un rôle primordial pour contrôler la vitesse de convergence vers un sous-ensemble optimal de taille minimale. Jusqu'à présent cette probabilité est fixée à 0,25 parce que notre objectif est la sélection d'un côté et l'accélération de notre système de sélection. Par contre, si l'objectif est la hiérarchisation nous pouvons partir de 75% de "1" et ainsi nous pouvons faire une suite de sélections. La figure 6.6 illustre ce processus de sélection.

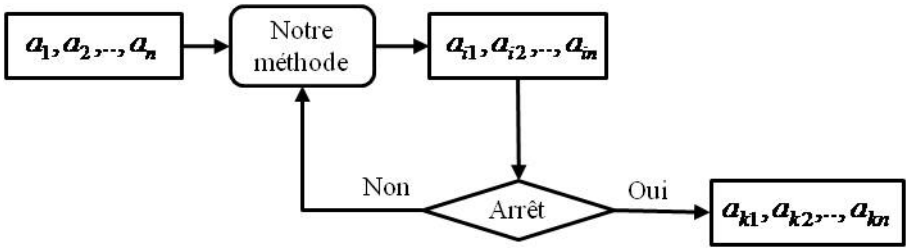

Figure 6.6 – Processus d'une sélection hiérarchique au niveau des caractéristiques

Le tableau 6.7 montre les résultats de ce type de sélection sur différents descripteurs de la base MNIST. Nous pouvons constater qu'avec une probabilité de 0,75, nous sommes capables de retrouver approximativement les mêmes résultats qu'avec une probabilité de 0,25 en faisant la sélection sur plusieurs itérations. Le processus de sélection continue jusqu'à une dégradation de la qualité. Pour les deux descripteurs *Zernike* et *GFD_8×12*, la sélection s'arrête au deuxième niveau. Par contre, pour la *R-signature* après le troisième, niveau nous avons augmenté la probabilité afin d'aller plus doucement dans le but de retrouver le sous-ensemble optimal sélectionné par notre méthode avec une probabilité de 0,25. Nous pouvons remarquer qu'au quatrième, niveau nous avons retrouvé un sous-ensemble de la même qualité que le sous-ensemble sélectionné par notre méthode de base.

		Zernike		*GFD_8×12*		*R-signature*	
	Proba de "1"	Nb_carac	Taux	Nb_carac	Taux	Nb_carac	Taux
Sans Sélection	-	66	92.47	96	92.38	180	75.95
Avec sélection	0.25	29	92.42	36	92.55	40	79.55
Sélection niveau 1	0.75	39	92.54	66	92.45	101	76.08
Sélection niveau 2	0.75	28	92.35	48	92.58	71	76.9
Sélection niveau 3	0.75	23	90.85	28	92.33	45	79.43
Sélection niveau 4	0.9	-	-	-	-	40	79.45
Sélection niveau 5	0.9	-	-	-	-	34	79.25

Table 6.7 – Résultat de la sélection hiérarchique (cas d'un seul descripteur)

Finalement un critère d'arrêt peut être défini en se basant sur cette stratégie. La procédure de sélection commence avec une probabilité de 0,75 et la sélection continue jusqu'à ce que la qualité du sous-ensemble sélectionné diminue (niveau n). Après cette diminution, nous reprenons le sous-ensemble sélectionné au niveau n-1, nous changeons la probabilité à 0,90 pour aller moins vite et nous appliquons la sélection jusqu'à une diminution à nouveau (niveau n-1+k) et le sous-ensemble final est celui qui correspond au niveau n-2+k.

6.3.2 Sélection hiérarchique sur plusieurs descripteurs

Dans le cas de plusieurs descripteurs, il existe deux manières pour appliquer la méthode de sélection. La première consiste à mettre toutes les caractéristiques de tous les descripteurs ensemble et à appliquer la méthode de sélection sur l'ensemble total. La deuxième réalise une sélection sur chacun des descripteurs indépendamment des autres et le sous-ensemble final est représenté par la réunion des sous-ensembles sélectionnés pour chacun des descripteurs. Une sélection peut être appliquée sur ce dernier sous-ensemble (figure 6.8) pour aller plus loin et pour sélectionner des sous-ensembles de taille plus petite.

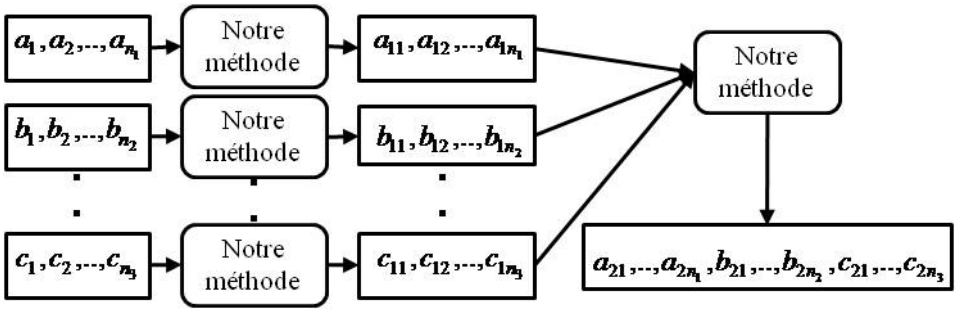

Figure 6.7 – Processus d'une sélection hiérarchique sur plusieurs descripteurs

Les résultats de ces deux techniques sont présentés ci-dessous dans le tableau 6.8. Sur ce tableau, nous pouvons constater que la sélection sur chacun des descripteurs donne de meilleurs résultats par rapport à une sélection sur l'ensemble. Par contre, une sélection sur

l'ensemble est capable de sélectionner un sous-ensemble de taille 12% plus petite mais avec une perte de 0.12% dans le taux de reconnaissance. Finalement, si le but est de trouver un sous-ensemble de taille minimale et si nous sacrifions un peu de taux de reconnaissance, une sélection sur l'ensemble de descripteurs peut être meilleure.

	Zernike + GFD_8×12 + R-signature	
	Nombre de caractéristiques	Taux de classification
Sans sélection	66+96+180=342	94.65
Sélection sur tous	20+8+32=60	94.51
Sélection sur chacun	25+30+46=101	94.63
Sélection sur la sélection	12+5+15=32	93.97

Table 6.8 – Résultat de la sélection hiérarchique (cas de plusieurs descripteurs)

6.4 Conclusion

Nous avons introduit dans ce chapitre un codage entier qui permet de contrôler le nombre de caractéristiques à sélectionner. La mesure de la diversité des classificateurs est aussi étudiée et l'intégration de cette mesure dans le processus d'optimisation en utilisant différentes techniques d'optimisation multi-objectifs, a aidé à éliminer la redondance des classificateurs et par conséquent les caractéristiques redondantes. Une phase d'expérimentations sur une base artificielle ainsi que sur la base MNIST a validé l'intérêt de la mesure de la diversité pour éliminer la redondance. Finalement, le sujet de la hiérarchisation au niveau de descripteur ainsi qu'au niveau de caractéristiques a été étudié.

Troisième partie

Applications

Chapitre 7

Applications

Les domaines où la sélection de caractéristiques peut être appliquée, sont très nombreux. Des collaborations avec d'autres équipes dans le cadre du projet ANR NaviDoMass (NAVIDOMASS [2010]) ainsi qu'avec des équipes de recherche, nous ont amené à appliquer notre méthode de sélection sur plusieurs types de problèmes. Dans ce chapitre, nous présentons deux applications. La première concerne les lettrines extraites de documents anciens et leur indexation selon des styles définis par les historiens. La deuxième consiste à appliquer notre méthode de sélection sur des données biologiques et plus précisément de sélectionner les caractéristiques les plus pertinentes pour résoudre un problème de classification de molécules.

7.1 Lettrines

Dans le cadre du projet ANR NaviDoMass (NAVIDOMASS [2010]), un de nos centres d'intérêt était la recherche par le contenu d'images graphiques de documents des $XV^{\text{ème}}$ et $XVI^{\text{ème}}$ siècles. Ces documents sont fournis par le Centre d'Études Supérieures de la Renaissance (CESR) à Tours. Nous avons voulu évaluer nos approches de sélection de caractéristiques sur plusieurs problèmes liés au document mais nous n'en développerons qu'un dans le cadre de la thèse. Nous présentons ici le travail réalisé pour la caractérisation et l'indexation des lettrines déjà extraites des documents anciens. Avant d'aborder ce sujet et pour contextualiser notre travail, nous donnons la définition d'une lettrine.
Une lettrine est une lettre majuscule commençant un paragraphe ou un texte ayant une hauteur supérieure à celle de la ligne. Celle-ci est souvent ornée d'enluminures, terme issu du latin illuminare. C'est en fait à cause d'une erreur de vocabulaire entre le terme illuminare et miniature, venant du latin minimum que la lettrine est devenue plus qu'une simple lettre posée sur une série d'entrelacs et s'est transformée au cours des siècles en la représentation d'une scène en miniature.

7.1.1 Problématique

Les lettrines que nous présentons viennent comme nous l'avons précisé de la base de données du CESR et ont été extraites à l'aide du logiciel Agora. Celui-ci a été conçu par l'équipe Reconnaissance des Formes et Analyse d'Images du Laboratoire d'Informatique de l'université de Tours. Il permet d'extraire les différentes parties d'un document ancien numérisé. Ainsi il peut séparer dans une page les bandeaux des lettrines, les lettrines du texte, etc. Il fonctionne à partir de scénarios élaborés par les experts du document. Il permet donc une extraction des lettrines dans une masse de documents. Par contre, cette extraction de masse rend alors nécessaire l'indexation des images extraites de manière à les rendre utilisables, c'est à dire permettre de retrouver l'information. Vu le grand nombre des lettrines extraites, il n'est pas envisageable de les étiqueter à la main. Indexer les milliers de lettrines extraites se révélerait un travail gigantesque. Cette indexation peut être considérée sur plusieurs niveaux (la lettre de la lettrine, le fond, le décor, etc.) car elle doit répondre aux besoins de différents types d'utilisateurs. Particulièrement, nous avons travaillé sur l'indexation liée au type de décor et au fond sur lequel sont placés la lettre et le décor.

7.1.2 Style de lettrines

Il semble que les lettrines aient, en fonction de leur nature, de l'enlumineur qui les a réalisées ou de l'imprimeur, des caractéristiques communes. Dans un même ouvrage, les lettrines ne portent pas la même majuscule et ne représentent pas la même scène mais elles ont été réalisées par une même personne et souvent illustrent une même histoire. Un même ouvrage peut contenir deux fontes de lettrines pour exprimer deux niveaux d'information différents. Ainsi nous pouvons définir ce que l'on appellera le style d'une lettrine. En effet même si deux lettrines ne représentent pas la même lettre, elles peuvent être illustrées par des scènes composées des mêmes personnages dans des situations différentes. Les types de végétation ou d'entrelacs sur lesquels sont posées les lettres, peuvent être sensiblement proches. La figure 7.1 illustre ce concept.

Figure 7.1 – Trois styles de lettrines

Ces exemples illustrent la notion de style, tel que nous l'envisageons dans la suite. Nous constatons qu'il n'y a pas de correspondance précise entre les lettrines d'un même

style mais on voit se dégager une ressemblance générale. Finalement, l'idée est de proposer un système d'étiquetage automatique des lettrines selon ces trois styles présents dans les ouvrages considérés dans le projet.

7.1.2.1 Indexation des lettrines

Le descripteur que nous avons utilisé pour la classification automatique des lettrines est basé sur les données considérées quand on veut modéliser une image par une loi de Zipf. Mais avant d'aborder les détails de la méthode nous allons rappeler l'énoncé de la loi de Zipf.

La loi de Zipf (Zipf [1949]) est une loi empirique qui a été énoncée il y a une cinquantaine d'années. Elle repose sur une loi puissance qui affirme que dans un ensemble de symboles organisés de façon topologique, la distribution des fréquences des n-uplets, nommés motifs, n'est pas aléatoire. Notons les fréquences d'apparition $N_1, N_2, .., N_n$ de ces symboles, ici des motifs dans le cas de l'étude d'image, $M_1, M_2, .., M_n$. Quand ils sont triés par ordre décroissant de leur fréquence d'apparition, les fréquences sont liées aux rangs de ces fréquences. Cette relation peut être formulée ainsi :

$$N_{\sigma(i)} = k \times i^a \tag{7.1}$$

$N_{\sigma(i)}$ représente le nombre d'apparitions d'un motif de rang i, k et a sont des constantes. Cette loi est caractérisée par l'exposant a. k, lui, est plus lié à la longueur de la séquence de symboles étudiée ou ici à la taille de l'image considérée. La complexité de la loi tient au fait qu'elle n'est pas linéaire. Néanmoins l'application d'un opérateur logarithmique conduit à une relation linéaire caractérisée elle aussi par deux paramètres. De plus, la valeur de l'exposant a peut être estimée expérimentalement par le coefficient directeur de la droite de régression approximant le graphe $(log_{10}(i), log_{10}(N_\sigma(i)))$ où $i = 1$ à n. Par la suite, nous appellerons ce graphe la courbe de Zipf.

Des études ont montré que la loi de Zipf était vérifiée dans le cas des images avec différents processus de codage (Caron *et al.* [2003]). Cette loi peut être considérée pour une image en prenant comme motifs des imagettes de l'image et en calculant leur fréquence d'apparition dans l'image totale et leur rang.

Les lettrines que nous étudions sont en niveaux de gris où chaque pixel est encodé sur 8 bits (256 niveaux de gris différents). Néanmoins pour 256 niveaux de gris il y a théoriquement 256^9 motifs possibles (motif de taille 3×3). Ce nombre n'est pas raisonnable car souvent supérieur au nombre de pixels des images. Cela entraînerait que chaque motif serait rare et de ce fait sa fréquence aurait peu de sens d'un point de vue statistique. Par exemple, dans une image de 640×480 pixels on ne trouve que 304964 motifs. Il devient indispensable de réduire le nombre de motifs possibles pour donner un sens au modèle. Des méthodes de ré-échantillonnage de l'échelle des niveaux de gris de l'image peuvent être utilisées afin de réduire le nombre de motifs totaux.

Une méthode d'indexation de lettrines selon les trois styles définis précédemment, a été

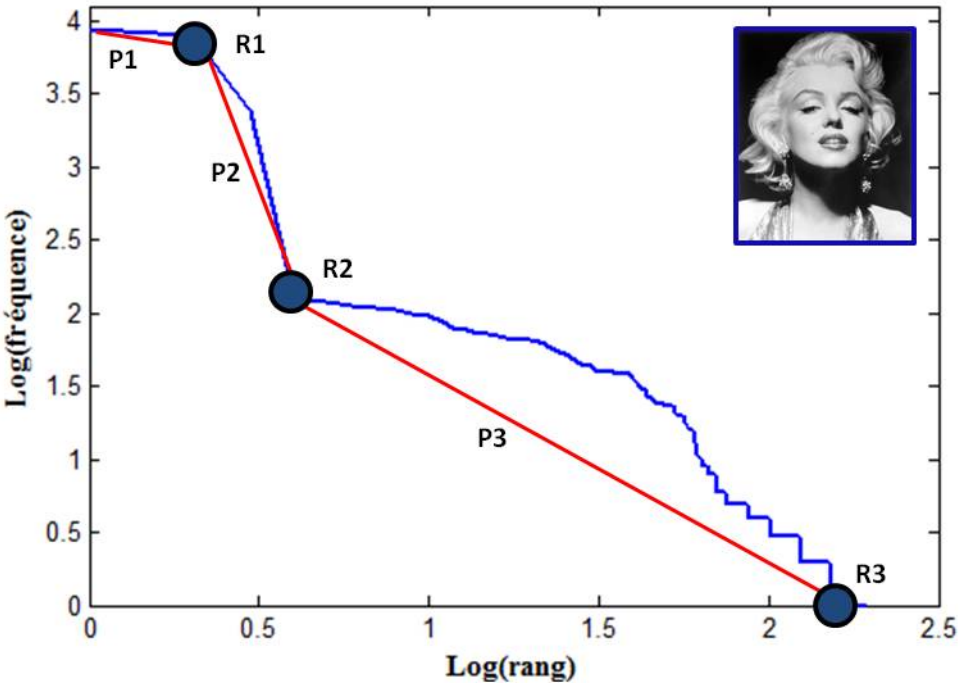

Figure 7.2 – Caractéristiques extraites sur le graphe de *Zipf*

proposée en 2006 (Pareti et Vincent [2006]). Cette méthode consiste à extraire sept caractéristiques du graphe de *Zipf* pour une image afin de les utiliser pour classer automatiquement les lettrines. Les caractéristiques sont extraites en extrayant trois zones linéaires dans chaque courbe. Elles sont extraites automatiquement par un processus récursif. Un premier point de coupure est obtenu comme le point le plus éloigné au sens de la distance euclidienne usuelle de la droite joignant les points de fréquence maximum et minimum. Ensuite, le processus est itéré sur la portion droite de la courbe. Un exemple de résultat représentatif est présenté sur la figure 7.2 qui montre un graphe de *Zipf* d'une image en niveaux de gris.

Les six caractéristiques extraites du graphe de *Zipf* sont respectivement les pentes (P_1, P_2, P_3) de trois droites et les abscisses (R_1, R_2, R_3) de trois points de rupture. La septième caractéristique est extraite de la courbe de *Zipf* inverse qui met l'accent sur les motifs rares plutôt que sur les motifs fréquents, elle permet de relire la partie droite de la courbe (figure 7.2). Dans la suite, nous utilisons le nom *"PRZ"* (**P**ente, **R**upture et **Z**ipf) pour désigner cette méthode.

Dans notre cas, nous disposons de plusieurs descripteurs, la fréquence et le rang des motifs de taille (2×3 ou 3×2). La taille des descripteurs est égale au nombre de motifs possibles. Un ré-échantillonnage en trois niveaux de gris est appliqué sur les images avant de calculer le descripteur. Le ré-échantillonnage s'appuie sur l'algorithme K-means. Avec des motifs de taille (2×3 ou 3×2) et trois niveaux de gris, nous avons 729 motifs possibles. Sur les 729 caractéristiques nous avons appliqué notre méthode de sélection de caractéristiques afin de

trouver les motifs les plus discriminants, capables de résoudre le problème de caractérisation du style et nous présentons les résultats dans la section suivante.

7.1.2.2 Résultats

La première base de lettrines fournie par le CESR est composée de 300 lettrines qui sont reparties équitablement entre les styles, 100 lettrines pour chacun des styles. Chaque lettrine peut être représentée par un vecteur dont les composantes sont les rangs ou les fréquences des différents motifs. Avec un tel modèle vectoriel et un nombre d'exemples faibles, le classificateur le plus simple qui peut être utilisé est le Knn (K plus proches voisins).
Le tableau 7.1 montre les résultats de classification en utilisant les trois descripteurs définis précédemment (la fréquence, le rang et le descripteur à 7 caractéristiques utilisé dans la méthode *PRZ*) pour les trois styles. Dans ce tableau, nous pouvons remarquer que les deux descripteurs (fréquence, rang) sont meilleurs que celui de *PRZ* mais il faut reconnaître que cette dernière méthode utilise beaucoup moins de caractéristiques ce qui la rend plus avantageuse même si le calcul des caractéristiques est un peu sophistiqué.

	Style1	***Style2***	***Style3***
Fréquence	100	100	100
Rang	100	100	100
Méthode PRZ	100	93	100

Table 7.1 – Résultats de classification en utilisant différents descripteurs pour les trois styles

Pour diminuer le nombre de caractéristiques utilisé, nous avons appliqué notre méthode de sélection et le tableau 7.2 montre les résultats avant et après la sélection. La sélection est faite en utilisant le processus de sélection présenté dans la section (6.3.1) appliqué sur le descripteur de rang. Le sous-ensemble final est trouvé en appliquant notre méthode plusieurs fois (niveaux).

		Taux de classification				
Méthode	***Taille de motif***	***Style1***	***Style2***	***Style3***	***Nb de carac***	***Sélection***
PRZ	***3×3***	100	93	100	6	-
Notre méthode	***2×3***	100	100	100	729	-
		100	100	100	224	Niveau 1
		100	100	100	88	Niveau 2
		100	100	100	19	Niveau 3
		100	100	100	6	Niveau 4

Table 7.2 – Résultats de sélection sur le descripteur Rang

Chaque fois, la sélection est appliquée sur le sous-ensemble de caractéristiques sélec-

tionné à l'itération précédente. Nous constatons que nous avons réussi à sélectionner six caractéristiques parmi les 729 et qui sont capables de résoudre ce problème tout en gardant 100% de réussite pour les trois styles. Signalons que les taux de classification sont obtenus par un classificateur Knn avec $K = 1$.
La méthode d'extraction des caractéristiques à partir du graphe de Zipf permet une interprétation des différences entre styles assez grossière, sur les tailles des zones qui ont pu être mises en évidence, mais aucune information précise ne peut être formulée sur les différents types. Au contraire, l'utilisation de la fréquence ou du rang des motifs comme des caractéristiques permet de conserver l'information, en particulier les motifs utilisés. Les six motifs sélectionnés par notre méthode sont montrés dans la figure 7.3. Sur cette figure, nous pouvons constater que tirer des conclusions ou des interprétations sur les motifs sélectionnés est difficile. Ainsi, pour obtenir une interprétation du processus de sélection, nous avons analysé ces six motifs. Pour l'analyse, les moyennes des rangs pour chacun des motifs sélectionnés sont calculées pour chaque style.

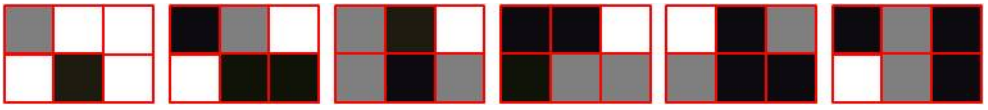

Figure 7.3 – Les six motifs sélectionnés

Le tableau 7.3 montre des statistiques sur les rangs pour chacun des motifs sélectionnés et cela pour chaque style. Ces résultats nous permettent de localiser la zone des rangs des motifs sélectionnés sur le graphe de Zipf afin d'en tirer une conclusion. Nous pouvons constater que tous les motifs ont un rang élevé et cela pour les trois styles. Le logarithme décimal du rang varie entre 2,15 et 2,855. A l'aide de ces deux valeurs nous pouvons localiser et placer les motifs sélectionnés sur le graphe de Zipf.

	Style 1			***Style 2***			***Style 3***		
Motif	Moyenne	Max	Min	Moyenne	Max	Min	Moyenne	Max	Min
M_1	446	250	515	479	271	585	405	278	487
M_2	605	450	670	664	496	701	634	456	687
M_3	482	295	555	573	418	624	513	253	602
M_4	555	365	689	669	489	710	680	493	711
M_5	465	301	621	411	283	519	368	149	421
M_6	642	510	693	691	580	705	671	498	703

Table 7.3 – Statistiques sur les motifs sélectionnés

La figure 7.4 montre un exemple de trois graphes de Zipf où chacun représente le graphe d'une des trois styles de lettrine. La région en gris représente la zone des six motifs sélectionnés ($2.15 < log_{10}(rang) < 2.85$). Nous concluons finalement, à partir de cette figure, que les motifs les plus pertinents sont relativement rares.

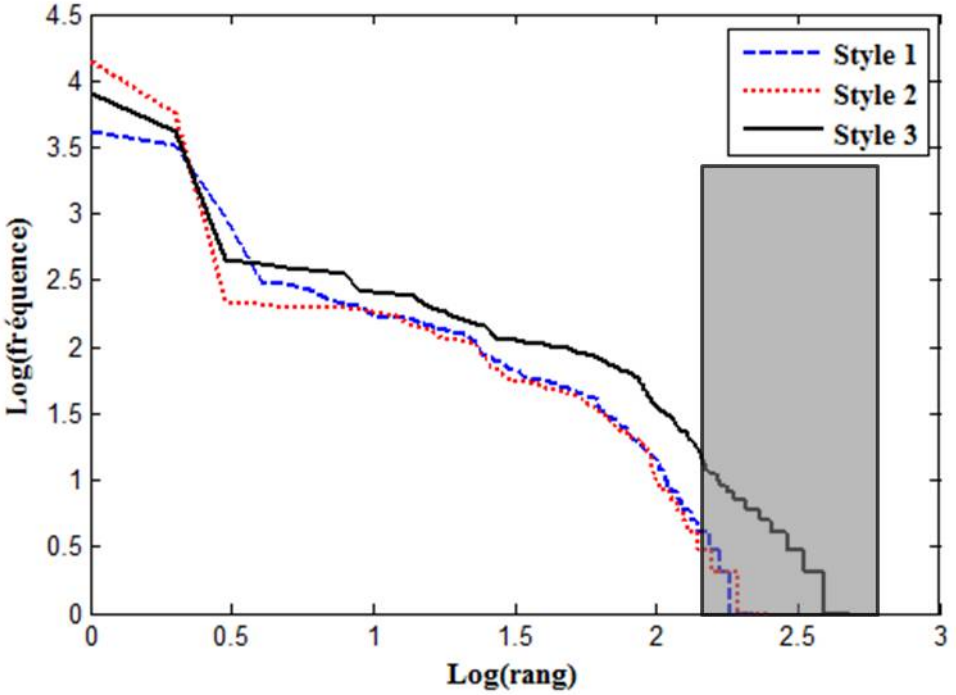

Figure 7.4 – Zone des six motifs sélectionnés dans la courbe de Zipf

Nous pouvons également signaler que les résultats présentés en considérant le descripteur des rangs peuvent être faits de manière similaire avec le descripteur des fréquences. Pour ce problème en particulier, les résultats sont sensiblement les mêmes.

7.1.3 Redéfinition des styles

A la suite de ces premiers résultats, les historiens du CESR ont défini d'autres styles en basant cette nouvelle classification sur le fond de la lettrine. Cela conduirait à une autre indexation des lettrines. Ces styles sont nommés ***Criblé, Hachuré, Noir*** et ***Blanc*** et ils sont illustrés dans la figure 7.5.

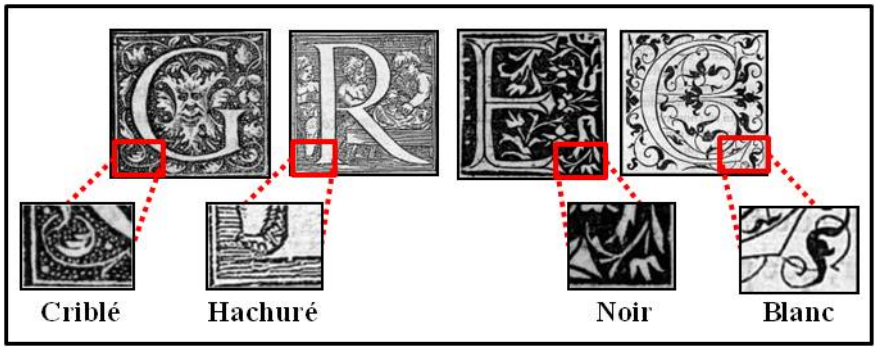

Figure 7.5 – Les quatre nouveaux styles de lettrines

Le style **Criblé** a pour spécificité que le fond contient des points blancs, de petites taches blanches. De petites lignes horizontales parallèles dans le fond caractérisent le style

Hachuré. Pour les deux autres styles, la scène de la lettrine est construite respectivement sur un fond noir ou blanc.
Des difficultés pour distinguer ce genre de lettrines sont présentes à différents niveaux. Le premier est la qualité des lettrines considérées. Il y a un certain nombre d'images où la résolution et la qualité ne sont pas suffisantes. Le deuxième est que la définition des styles par les historiens est basée sur une vision humaine qui intègre d'autres informations que celles de l'image seule. Par exemple, nous pouvons trouver une scène en noir dans une lettrine qui remplit la totalité du fond et qui ne laisse que quelques petites zones blanches. Pour les historiens, cette lettrine est considérée comme ayant un fond blanc (qui occupe une portion ultra minoritaire de l'image) et cela ne peut pas être le cas du point de vue informatique.
La figure 7.6 illustre pour chacun des nouveaux styles quelques exemples de lettrines qui posent problèmes au niveau de la classification.

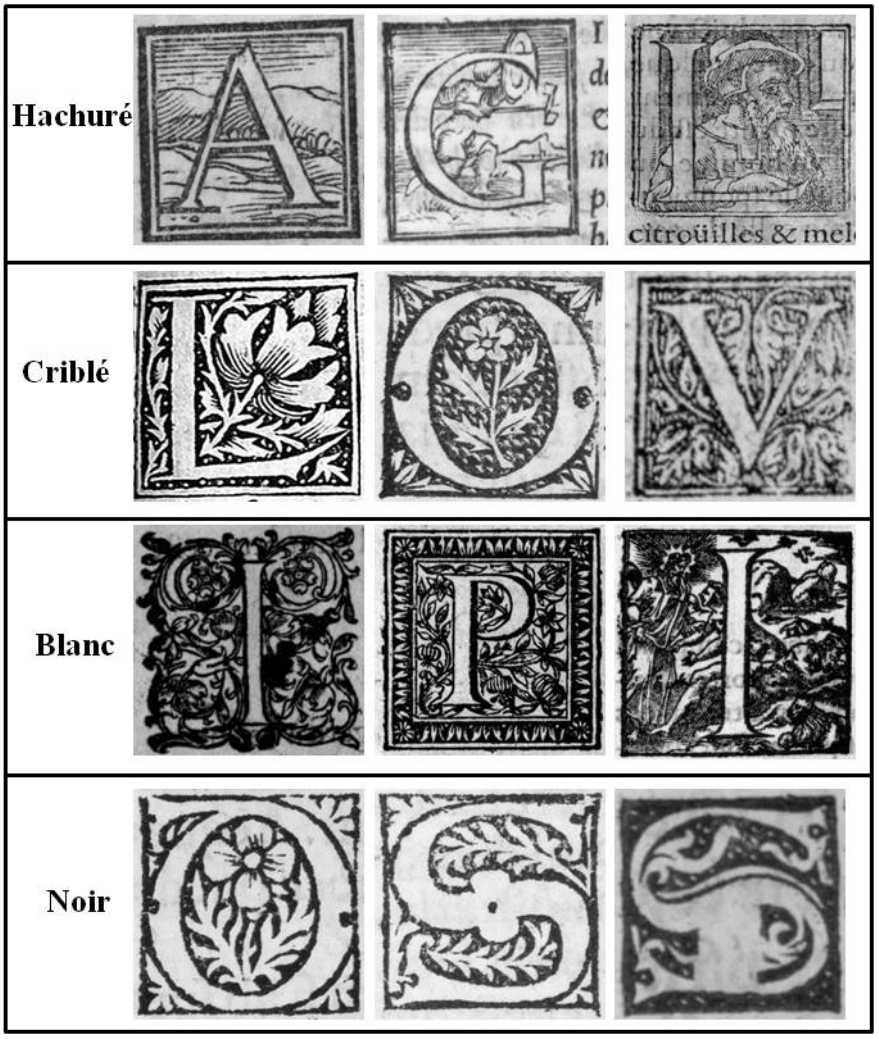

Figure 7.6 – Exemples de lettrines posant des difficultés pour la classification selon les nouveaux styles

7.1.3.1 Résultats

Pour évaluer les méthodes de reconnaissance nous disposons d'une base de lettrines contenant 2670 lettrines. Les lettrines ne sont pas réparties de manière équitable entre les styles. Le tableau 7.4 montre la répartition des lettrines dans chacun des nouveaux styles.

Criblé	**Hachuré**	**Noir**	**Blanc**	**Total**
258	674	258	1480	2670

Table 7.4 – Nombre de lettrines de chacun des styles

Comme dans le cas du problème précédent, nous avons utilisé un classificateur Knn pour la classification. Cela évite la difficulté d'une phase d'apprentissage, spécialement lorsque le nombre d'exemples de la base est relativement faible. Plusieurs valeurs de k sont testées ($k = \{1, 2, 5, 10\}$). Comme descripteurs, nous avons considéré les mêmes descripteurs (le rang et la fréquence de motifs ainsi que celui de la méthode *PRZ*) que dans le problème précédent.
Le tableau 7.5 montre les taux de classification obtenus à l'aide d'un classificateur Knn sur chacun des descripteurs où k appartient à $\{1, 3, 5, 10\}$.

	Descripteur		
K	Fréquence	Rang	Méthode PRZ
1	79.84	71.9	74
3	80.75	73.7	77.27
5	**80.85**	75.73	**77.95**
10	79.65	**77.27**	77.44

Table 7.5 – Résultats de reconnaissance des types de fond pour différents descripteurs

Nous pouvons constater que les résultats obtenus en utilisant la fréquence des motifs comme descripteur sont meilleurs que ceux obtenus en utilisant les rangs. Cela prouve que certaines informations sont perdues lorsque nous considérons le rang seul sans prendre en compte la fréquence. En effet, il a été montré dans (Pareti et Vincent [2006], Coustaty *et al.* [2009]) que la distribution des motifs dans une image, dans le cas des lettrines, ne peut pas être modélisée par une seule loi de Zipf, mais plutôt par un mélange de lois de Zipf. Ce qui prouve la différence entre le rang et la fréquence. Une comparaison avec la méthode *PRZ* est aussi présentée dans ce tableau et nous constatons que la fréquence des motifs donne de meilleurs résultats mais il faut reconnaître que le nombre de caractéristiques utilisées est beaucoup plus important.
Nous avons appliqué notre méthode de sélection comme nous l'avons fait dans la section précédente. Les résultats sont présentés dans le tableau 7.6 où nous pouvons conclure que notre méthode est capable de réduire le nombre de caractéristiques sans perdre pour autant

en qualité.

Criblé			
	Niveau de sélection		
	-	1	2
K	729	209	58
1	90.38	90.88	88
3	91.33	90.90	89.48
5	**91.04**	**91.15**	89.57
10	90.96	90.76	**89.98**

Hachuré			
	Niveau de sélection		
	-	1	2
K	729	215	68
1	94.86	94.36	93.95
3	94.98	94.89	94.65
5	**95.15**	**95.05**	**94.95**
10	94.36	94.25	94.55

Noir			
	Niveau de sélection		
	-	1	2
K	729	213	87
1	91.17	90.3	89.06
3	**92.33**	91.72	**89.52**
5	92.12	**91.85**	89.35
10	91.67	90.65	89.35

Blanc			
	Niveau de sélection		
	-	1	2
K	729	221	71
1	96.97	96.52	96.27
3	97.14	96.85	96.23
5	**97.47**	**97.35**	**96.47**
10	97.18	96.95	96.31

Table 7.6 – Résultats de reconnaissance des styles après sélection

Nous pouvons constater aussi qu'à partir du deuxième niveau le taux de classification commence à chuter ce qui montre que nous avons besoin d'un nombre élevé de motifs pour caractériser ce problème ce qui reflète la difficulté de classer les nouveaux styles.

7.1.3.2 Pondération des motifs

Dans le but d'améliorer les résultats et si nous supposons qu'un motif est comme un terme dans un document textuel et qu'une lettrine peut jouer le rôle du document, nous pouvons donc appliquer des modèles utilisés dans les techniques de recherche d'information. Généralement, les termes sont pondérés afin d'améliorer leur pertinence. Un des modèles de pondération le plus connu dans ce domaine est le Tf-Idf.

7.1.3.2.a Tf-Idf

Le TF-IDF (***T**erm **F**requency-**I**nverse **D**ocument **F**requency*) est une méthode de pondération souvent utilisée en recherche d'information et en particulier dans la fouille de textes (Salton et McGill [1986]). Cette mesure statistique permet d'évaluer l'importance d'un terme contenu dans un document, relativement à une collection ou un corpus. Le poids augmente proportionnellement au nombre d'occurrences du mot dans le document. Il varie également en fonction de la fréquence globale du mot dans le corpus.

La fréquence du terme (term frequency) est simplement le nombre d'occurrences de ce terme dans le document considéré. Le principe est que plus un terme est fréquent dans ce document, plus il est important dans la description de celui-ci. Soit le document d_j et le terme t_i , alors la fréquence Tf_{ij} du terme dans le document est calculée par :

$$Tf_{ij} = \frac{f_{ij}}{\sum_k f_{kj}} \tag{7.2}$$

où f_{ij} est le nombre d'occurrences du terme t_i dans d_j. Le dénominateur est le nombre d'occurrences de tous les termes dans le document d_j.
L'inverse de la fréquence de documents (inverse document frequency) est une mesure de l'importance du terme dans l'ensemble du corpus. Elle consiste à calculer le logarithme de l'inverse de la proportion de documents du corpus qui contiennent le terme. Cette mesure est calculée par la formule suivante :

$$Idf_i = log\left(\frac{N}{n}\right) \tag{7.3}$$

où n est la proportion des documents contenant le terme t_i et N le nombre total de documents dans la collection.
Finalement le poids W_{ij} d'un terme t_i du document d_j est calculé par le produit de deux mesures Tf_{ij} et Idf_i ($W_{ij} = Tf_{ij} * Idf_i$).
Pour quantifier la similarité entre un document d et une requête q, plusieurs mesures de similarité peuvent être utilisées. Les plus connues sont : la similarité du Cosinus et celle de Jaccard, ces deux mesures sont présentées dans le tableau 7.7. En notant que $|d|^2 = \sum_{i=1}^{n} d_i^2$ et $d.q = \sum_{i=1}^{n} d_i \times q_i$.

Cosinus	**Jaccard**
$Sim(d,q) = \frac{d.q}{\|d\|\|q\|}$	$Sim(d,q) = \frac{d.q}{\|d\| + \|q\| - d.q}$

Table 7.7 – Quelques mesures de similarité

7.1.3.2.b Résultats

Nous avons utilisé le modèle Tf-Idf pour pondérer les motifs. Les résultats sont affichés dans le tableau 7.8. Après la pondération, nous avons utilisé un classificateur Knn en utilisant la distance euclidienne ainsi que deux autres distances qui se basent sur les mesures de similarité de Cosinus et de Jaccard.

K	Fréquence	Rang	Méthode *PRZ*	Tf-Idf		
				Euclidienne	Jaccard	Cosinus
1	79.84	71.90	74.00	**83.00**	**83.03**	83.30
3	80.75	73.70	77.27	82.84	82.58	84.31
5	**80.85**	75.73	**77.95**	82.43	82.32	**84.89**
10	79.65	**77.27**	77.44	81.57	82.13	83.70

Table 7.8 – Résultats sans et avec la pondération par le modèle Tf-Idf

Les résultats après la pondération montrent une amélioration par rapport aux résultats initiaux et pour les trois distances. Une amélioration de 4% est obtenue en utilisant la mesure de similarité de Cosinus.
Finalement, nous présentons dans le tableau 7.9 les résultats détaillés de classification de chacun des styles en comparant les résultats sans et avec la pondération par le modèle Tf-Idf. Nous pouvons remarquer que le taux de reconnaissance est amélioré pour tous les styles et surtout pour le style noir. Cette amélioration montre la capacité du modèle Tf-Idf à trouver les motifs les plus représentatifs de chaque style. L'amélioration est d'autant plus sensible que le nombre d'exemples est faible. Le fait que les deux styles **Hachuré** et **Noir** aient peu d'exemples risque de conduire à ce que les autres les dominent en utilisant un classificateur Knn. Par contre, la pondération par Tf-Idf réduit l'effet de disproportion des nombres d'exemples de chacun des styles en supprimant les motifs vides (les motifs qui sont très fréquents dans la lettrine et dans toutes les lettrines).

	Criblé	**Hachuré**	**Noir**	**Blanc**
Fréquence	60.85	81.3	43.58	90.61
Tf-Idf	75.58	81.68	72.37	91.22
Taux d'amélioration	14.73	0.38	28.79	0.61

Table 7.9 – Amélioration obtenue par le modèle Tf-Idf

7.1.4 Conclusion

Pour cette application, notre méthode de sélection a aidé à trouver les motifs les plus pertinents pour un problème de classification automatique de lettrines selon des styles définis par les historiens ainsi à mieux comprendre et interpréter les résultats. L'application du modèle Tf-Idf utilisé dans le domaine de la recherche d'information a permis la pondération des motifs et par conséquent l'amélioration des résultats en pénalisant les motifs très fréquents dans la lettrine et en même temps très fréquents dans toute la base.

7.2 Données biologiques

Dans le cadre d'une collaboration avec l'équipe "Qgar" du "LORIA" qui a amené à d'autres collaborations avec l'équipe "ORPAILLEUR" du même laboratoire, des bases de données biologiques ont été fournies dans le but de sélectionner des caractéristiques pertinentes pour un problème de classification de molécules. Ce problème consiste à identifier de nouvelles molécules susceptibles de devenir des médicaments. La conception de nouveaux médicaments est un enjeu scientifique et économique très important de nos jours. La première question posée lors de la considération d'un nouveau médicament est : "comment affecter la cible du processus moléculaire causant la maladie ?". Durant des décennies, une méthode d'essais-erreurs a été utilisée. Il s'avère qu'étant donné les contraintes économiques et temporelles, cette méthode n'est certainement pas la meilleure. Ainsi, la recherche pharmaceutique a de plus en plus eu recours à des technologies permettant de synthétiser un très grand nombre de molécules simultanément et de tester leur action sur une cible thérapeutique donnée. De récentes évolutions concernent la création d'outils informatiques adaptés au haut débit pour le criblage de bases de données chimiques réelles et virtuelles. Le criblage virtuel sert ainsi à réduire des bases qui contiennent un nombre trop important de composants (caractéristiques) en un ensemble d'éléments prometteurs, par rapport à une cible (ou une famille de cibles) à travers l'application de méthodes informatiques. Ce criblage s'appuie sur l'utilisation des descripteurs moléculaires qui vont permettre de caractériser les propriétés des molécules. Ces propriétés ne dépendent pas seulement des atomes qui composent la molécule mais aussi des liaisons entres ces atomes.

7.2.1 Descripteurs moléculaires

Les descripteurs sont utilisés pour caractériser les molécules à analyser. Ils peuvent être calculés à partir de la structure (constitution, configuration et conformation moléculaires) ou des propriétés (physiques, chimiques, biologiques) appartenant aux molécules (Brown [1997], Todeschini et Consonni [2000]). Ils doivent vérifier un certain nombre de propriétés : invariance par rapport à l'étiquetage et la numérotations des atomes, invariance par rapport à la rotation-translation de la molécule, calcul avec une méthode algorithmique non ambiguë, valeurs dans un intervalle de valeurs acceptables(Randić [1996]).
Les descripteurs constitutionnels (voir figure 7.7) incluent de l'information au niveau de l'ordre des atomes ($1D$) et de leurs liaisons ainsi que sur la présence ou l'absence de fragments, et un certain nombre d'autres caractéristiques 2D. Les descripteurs configurationnels concernent l'arrangement en 3D des atomes et les descripteurs conformationnels représentent l'arrangement spatial thermodynamique stable des atomes dans une molécule.

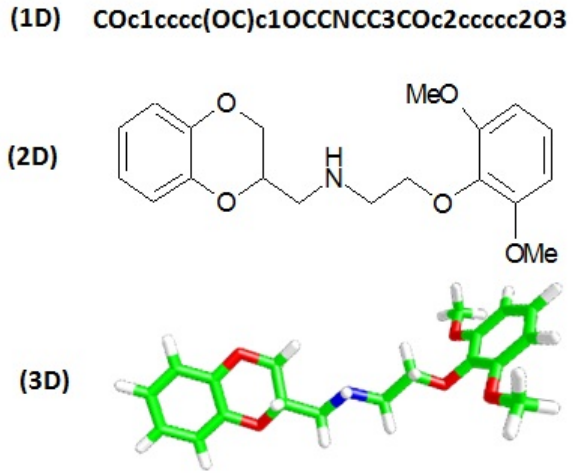

Figure 7.7 – Représentation 1D, 2D et 3D pour la formule chimique $C_{19}H_{23}NO_5$

Idéalement, les descripteurs utilisés pour le développement des modèles moléculaires devraient être rapidement calculables et facilement interprétables. Ils devraient représenter la réalité chimique du système et optimiser ainsi la structuration de l'espace chimique.
Les descripteurs moléculaires ont augmenté dernièrement, en nombre et en complexité. La plupart sont obtenus, soit à travers des définitions spécifiques, soit par des combinaisons d'autres descripteurs. Souvent, ils sont composés de valeurs numériques qui correspondent généralement à des propriétés physico-chimiques.

7.2.2 Description des bases de données

Les objectifs du projet "ORPAILLEUR" sont l'extraction de connaissances dans les bases de données, la gestion de connaissances dans les bases de données et le web sémantique. ces trois points sont importants et ont pour but commun de passer des données aux connaissances. Ils partagent comme processus commun la classification.
Dans le cadre du notre collaboration, nous avons étudié la possibilité d'améliorer le processus de classification tout en réduisant le nombre de caractéristiques moléculaires utilisées qui caractérisent des bases de données d'éléments chimiques.
Ces bases sont composées de six classes qui correspondent à six cibles thérapeutiques qui peuvent influencer positivement quatre types de maladies (syndromes) (tableau 7.10). Ces cibles (ou composantes de ces cibles) peuvent être représentées par un polysaccharide, un lipide, une protéine ou un acide nucléique. Dans le cadre des bases de données utilisées, les cibles sont toutes représentées par des protéines (plus particulièrement par des enzymes). De manière générale, dans 40% des cas de cibles thérapeutiques, la cible constitue une enzyme jouant un rôle direct ou indirect sur un ou plusieurs facteurs d'un sentier métabolique influençant positivement ou négativement une maladie.

	Cibles thérapeutiques	***Syndrome***
Classe 1	Adénosine kinase	Anti inflammatoire
Classe 2	Factor Xa	Cardiovasculaire
Classe 3	Thrombine	Cardiovasculaire
Classe 4	Méthyltransférase	MDS (myelodysplastic syndrome)
Classe 5	HIV reverse transcriptase	sida
Classe 6	HIV protéase	sida

Table 7.10 – Les cibles du problème

Chaque molécule de la base est caractérisée par 199 caractéristiques. Pour les biologistes, les caractéristiques proposées sont très corrélées et une réduction de la dimensionalité devrait conduire probablement à une perte dommageable d'information. Cela a motivé le test de notre méthode afin d'étudier la possibilité de diminuer le nombre de caractéristiques utilisées. Les deux bases de données sont décrites ci-dessous.

Base de deux-classes : Cette base est composée de 4474 molécules. 1152 molécules appartiennent à la classe "***adénosine***" et le reste appartient à une deuxième classe appelée "***Factor Xa***". Nous avons construit une base d'apprentissage de 1400 exemples (700 exemples de chacune des classes choisis aléatoirement). Le reste des exemples représente notre base de test.

Base de six-classes : Avec six classes, la base de données fournie est plus large. Elle contient 20101 molécules. Le tableau 7.11 résume le nombre de molécules de chacune des six classes. Nous avons utilisé le même protocole défini pour la base MNIST (section 5.5.1.1) qui s'appuie sur l'approche "*un contre tous*" pour construire des sous-ensembles de molécules associés à chacune des classes. Pour l'apprentissage, chaque sous-ensemble est composé de 1000 exemples de la classe et 1000 de toutes les autres classes. Chaque base de validation contient 500 exemples de la classe et 500 de toutes les autres et à partir des exemples restants nous avons construit nos bases de tests.

Classe 1	**Classe 2**	**Classe 3**	**Classe 4**	**Classe 5**	**Classe 6**
3066	3173	3653	5195	3323	2691

Table 7.11 – Nombre de molécules pour chacune des classes

7.2.3 Résultats

Pour les deux problèmes de classification décrits précédemment, nous avons appliqué notre méthode de sélection afin de trouver les caractéristiques les plus pertinentes. La technique de sélection utilisée est celle utilisée dans la section 6.7 avec une probabilité de $0,5$ pour les gènes à "1". La sélection est faite sur deux niveaux et nous présentons dans la suite les détails des résultats.

Cas de deux-classes : Les tests préliminaires pour résoudre ce problème de classification en utilisant un classificateur SVM et sans aucune sélection de caractéristiques montrent que le problème n'est pas difficile, et les deux classes sont faciles à distinguer. Un taux de classification de 99,5 est obtenu avec un SVM. Malgré ces bons résultats, nous avons étudié la qualité des caractéristiques utilisées en utilisant notre méthode de sélection de caractéristiques dans le but d'aider les biologistes à mieux comprendre leurs résultats d'un côté et d'autre part pour éviter l'extraction de caractéristiques inutiles.
Le tableau 7.12 montre les résultats sans et avec sélection. Ce tableau montre que même avec un taux de classification élevé de base, nous avons réussi et sur deux niveaux de sélection, à améliorer le taux de reconnaissance et à réduire de plus de 65% le nombre de caractéristiques utilisées pour le premier niveau de sélection et de plus de 86% pour le deuxième niveau de sélection en utilisant notre méthode. Le deuxième niveau de sélection consiste à appliquer la méthode de sélection sur l'ensemble sélectionné. Cette sélection peut aider les biologistes à mieux comprendre l'importance et l'influence de chacune de ces caractéristiques et à réduire la tâche de calcul pendant l'extraction des caractéristiques en évitant d'extraire plus de 65 à 86% des caractéristiques actuelles.

Sans sélection		
Taux de Classification	Nb de Caractéristiques	Erreur (classe1-classe2)
99.5	199	12-2

Avec sélection						
	Niveau1			**Niveau2**		
	Taux	Nb carac	Erreur	Taux	Nb carac	Erreur
Meilleur	99.8	64	4-2	99.83	37	4-1
Moyenne	99.65	70	9-2	99.52	28	12-2
SFS(Fisher)	99.7	66	7-2	99.18	29	17-8
SFS(Corrélation)	99.7	64	7-2	99.06	29	20-9

Table 7.12 – Résultats d'un classificateur SVM avec et sans sélection

Pour la première partie "Sans sélection" on obtient le taux de classification en utilisant un classificateur SVM avec toutes les caractéristiques. La troisième colonne représente le nombre d'exemples erronés dans chacune des classes. La deuxième partie "Avec sélection" est divisée en deux sous parties (Niveau1 et Niveau2). Les trois colonnes, et pour chacun des niveaux, représentent respectivement le taux de classification pour chacune des méthodes, le nombre de caractéristiques utilisées et le nombre d'exemples erronés dans chacune des classes. Les deux premières lignes représentent le meilleur cas ainsi que la moyenne des résultats de dix exécutions de notre méthode. Les deux dernières lignes de la deuxième partie montrent les résultats des autres méthodes de sélection basées sur les scores de *Fisher* et *Corrélation* en utilisant toujours un SVM et la technique *SFS* pour choisir les caractéristiques. Les résultats montrent que notre méthode est plus performante que les

autres dans tous les cas et c'est quel que soit le niveau de sélection. Les meilleurs résultats sont obtenus sur le deuxième niveau en utilisant notre méthode.

Problème de classification 6-classes Les mêmes types de caractéristiques du problème précédent, ont été utilisées pour résoudre ce nouveau problème de six classes. Le tableau 7.13 montre les résultats de la sélection sur cette nouvelle base.

	Sans sélection	***Avec sélection***				
		Niveau1		**Niveau2**		
	Taux de classif	Taux de classif	Nb carac	Taux de classif	Nb carac	% de réduction
Classe 1	83.5	84.00	89	84.55	39	80.40
Classe 2	96.85	96.95	95	97.55	41	79.39
Classe 3	62.1	63.15	101	63.75	40	79.89
Classe 4	78.05	83.3	86	86.3	42	78.89
Classe 5	84.8	84.9	91	85.8	37	81.40
Classe 6	79.4	80.7	93	81.05	38	80.90

Table 7.13 – Résultats d'un classificateur SVM avec et sans sélection pour les six classes

Nous pouvons constater que pour les six classes, nous avons réussi à améliorer les taux de classification tout en diminuant le nombre de caractéristiques de plus de 78% et celas pour les six classes. et nous pouvons tirer la même conclusion que celle du problème précédent.

7.2.4 conclusion

Dans cette application le but était de sélectionner les caractéristiques les plus pertinente pour classer des molécules selon des cibles bien précises. Les caractéristiques fournies par les biologistes sont très corrélées de leur point de vue et une sélection dessus devrait conduire probablement à une perte dommageable d'information. Nous avons réussi à démentir cette hypothèse puisqu'en appliquant notre méthode, une réduction de 78% du nombre de caractéristiques a été réalisée et cela avec une amélioration du taux de classification des molécules.

Chapitre 8

Conclusion générale et perspectives

L'analyse de techniques de sélection existantes nous a permis de mettre en évidence un certain nombre de faiblesses parmi lesquelles la complexité très élevée des approches *"wrapper"* ainsi que la dépendance des caractéristiques pertinentes sélectionnées par rapport au classificateur utilisé. De leur coté, les approches *"filter"* présentent plusieurs limitations concernant la redondance ainsi que les interactions entre les caractéristiques. Dans le but de limiter ces inconvénients, nous avons proposé une nouvelle méthode rapide de sélection de caractéristiques. Cette méthode est basée sur la construction et la sélection de classificateurs simples associés à chacune des caractéristiques. Plusieurs types de classificateurs simples ont été étudiés afin de trouver le classificateur le plus performant sur une seule caractéristique. Afin de trouver la meilleure combinaison des classificateurs, une optimisation par un algorithme génétique est proposée et différentes méthodes de combinaison sont considérées et adaptées à notre problème.

Une phase d'expérimentations a été faite en utilisant différentes types de descripteurs calculés sur la base MNIST de chiffres manuscrits. Cette phase nous a amené à valider la performance de la méthode proposée ainsi que sa stabilité. Avec la méthode proposée, nous avons réussi à réduire de 68% le nombre de caractéristiques tout en conservant approximativement les mêmes taux de classification dans le pire des cas et souvent en les améliorant. La comparaison avec d'autres méthodes de la littérature a montré que les résultats obtenus par notre méthode sont meilleurs que ceux des méthodes de filtrage et restent très proches de ceux des approches classiques de type *"wrapper"* mais le temps et la complexité sont grandement réduits (entre 127 et 250 fois plus rapide). Cela rend notre méthode plus avantageuse dans bien des applications.

Une étude sur la diversité entre les classificateurs simples associés aux caractéristiques initiales a été mise en place dans le but d'éliminer la redondance. Des mesures de diversité ont été intégrées dans la méthode en utilisant des techniques d'optimisation multi-objectifs afin de réduire cette redondance. Des expérimentations sur une base artificielle ont été réalisées

et ont montré l'intérêt de ces mesures.
Cette méthode originale de sélection a été appliquée pour résoudre des problèmes dans plusieurs domaines comme l'indexation des lettrines extraites de documents anciens et l'analyse de données biologiques. Pour les lettrines, notre méthode de sélection a aidé à trouver les motifs les plus pertinents pour un problème de classification automatique de lettrines selon des styles définis par les historiens, ainsi et à mieux comprendre et interpréter les résultats. Le but de la seconde application est de classer automatiquement des molécules selon des cibles bien précises. Pour les biologistes, les caractéristiques proposées sont très corrélées et une réduction de la dimensionnalité devrait conduire probablement à une perte d'information. Nous avons réussi à démentir cette hypothèse puisqu'en appliquant notre méthode, une réduction de 78% du nombre de caractéristiques a été réalisée et cela sans détériorer le taux de classification des molécules.

Nous avons bien conscience que notre travail ne constitue qu'un début des réflexions qui devrons être poursuivies en suivant les nombreuses pistes qui nous sont apparues durant ces années. À court terme pour ce travail, nous pensons qu'il faut :

- Tester la méthode de sélection sur le benchmark de NIPS 2003 (Guyon [2003]). Ce benchmark contient des bases de données spécifiques qui ont été construites dans le but d'évaluer des méthodes de sélection de caractéristiques.
- Appliquer notre méthode sur des données biologiques avec un nombre de descripteurs beaucoup plus large que dans l'étude présentée ici. Cela permettra d'améliorer la qualité du présent système de classification, dont le taux reste limité avec les 199 caractéristiques. Les biologistes disposent des milliers des caractéristiques moléculaires.
- Prendre en compte d'autres descripteurs pour l'indexation de lettrines afin d'améliorer les résultats.
- Tester la méthode d'indexation de lettrines sur d'autres bases graphiques par exemple la base des symboles GREC (Dosch et Valveny [2006]).

Nos travaux offrent également des perspectives à plus long terme. Par exemple, nous envisageons :

- Développer d'autres types de méthodes d'évaluation que la combinaison de classificateurs simples utilisée actuellement.
- Augmenter le nombre de classificateurs dans la base des classificateurs simples construits sur les caractéristiques initiales. Ces classificateurs pourraient être construits sur des triplets de caractéristiques tirées aléatoirement de la base initiale.
- Mieux analyser la nature de la transformation associée au classificateur construit sur la caractéristique dans le but de favoriser la caractéristique initiale ou son transformé.

- Proposer un système de classification automatique des molécules incluant notre méthode de sélection.
- Adapter notre méthode de sélection pour qu'elle soit utilisée dans le cadre de la modélisation par graphes, par la sélection de graphes dans une forêt de graphes.

Bibliographie

Aha, D. W. et Bankert, R. L. (1995). A comparative evaluation of sequential feature selection algorithms. *In 5th International Workshop on Artificial Intelligence and Statistics*, pages 1–7. Ft. Lauderdale, USA.

Alamdari, A. (2006). Variable selection using correlation and single variable classifier methods : Applications. *In* Guyon, I., Nikravesh, M., Gunn, S. et Zadeh, L., éditeurs : *Feature Extraction*, volume 207 de *Studies in Fuzziness and Soft Computing*, pages 343–358. Springer Berlin / Heidelberg.

Almuallim, H. et Dietterich, T. G. (1991). Learning with many irrelevant features. *In In Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 547–552. AAAI Press.

Almuallim, H. et Dietterich, T. G. (1992). Efficient algorithms for identifying relevant features. *In In Proceedings of the Ninth Canadian Conference on Artificial Intelligence*, pages 38–45. Morgan Kaufmann.

Asuncion, A. et Newman, D. J. (2007). Uci machine learning repository. http ://www.ics.uci.edu/mlearn/MLRepository.html.

Blum, A. et Rivest, R. (1993). Training a 3-node neural network is np-complete. *In* Hanson, S., Remmele, W. et Rivest, R., éditeurs : *Machine Learning : From Theory to Applications*, volume 661 de *Lecture Notes in Computer Science*, pages 9–28. Springer Berlin / Heidelberg.

Breiman, L. (1996). Bagging predictors. *Mach. Learn.*, 24:123–140.

Breiman, L. (2001). Random forests. *In Machine Learning*, pages 5–32.

Breiman, L. *et al.* (1984). *Classification and Regression Trees*. Chapman and Hall, New York.

Broadbent, H. A. et Lucas, J. (1988). Neural network model of serial learning. *SIGAPL APL Quote Quad*, 19:54–61.

Brown, R. (1997). *Descriptors for diversity analysis*, volume 7-8. Springer.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2:121–167.

Caron, Y., Charpentier, H., Makris, P. et Vincent, N. (2003). Power law dependencies to detect regions of interest. *In* Nyström, I., Sanniti di Baja, G. et Svensson, S., éditeurs : *Discrete Geometry for Computer Imagery*, volume 2886 de *Lecture Notes in Computer Science*, pages 495–503. Springer Berlin / Heidelberg.

Chen, X.-w. (2003). An improved branch and bound algorithm for feature selection. *Pattern Recognition Letters*, 24(12):1925 – 1933.

Choquet, G. (195). Theory of capacities. *Annales de l'institut Fourier, Grenoble.*, 5:131–295.

Coello, C. et Carlos, A. (1996). *An empirical study of evolutionary techniques for multiobjective optimization in engineering design.* Thèse de doctorat, New Orleans, LA, USA. AAI9639654.

Coello, C. A. et al. (1995). Multiobjective design optimization of counterweight balancing of a robot arm using genetic algorithms. *In Proceedings of the Seventh International Conference on Tools with Artificial Intelligence*, TAI '95, pages 20–, Washington, DC, USA. IEEE Computer Society.

Coustaty, M., Ogier, J.-M., Pareti, R. et Vincent, N. (2009). Drop caps decomposition for indexing a new letter extraction method. *In ICDAR 2009.*

Cover, T. M. et Thomas, J. A. (1991). *Elements of information theory.* Wiley-Interscience, New York, NY, USA.

Dash, M. et Liu, H. (1997). Feature selection for classification. *Intelligent Data Analysis*, 1:131–156.

Davis, L. (1991). *The genetic Algorithm Handbook.* Ed.New-York : Van Nostrand Reinhold,ch.17.

Deb, K., Pratap, A., Agarwal, S. et Meyarivan, T. (2000). A fast elitist multi-objective genetic algorithm : Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6:182–197.

Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees : Bagging, boosting, and randomization. *Machine Learning*, 40:139–157.

Dietz, W. E., Kiech, E. L. et Ali, M. (1989). Classification of data patterns using and autoassociative neural network topology. *In Proceedings of the 2nd international conference on Industrial and engineering applications of artificial intelligence and expert systems - Volume 2*, pages 1028–1036, New York, NY, USA. ACM.

Domingo, C. et Watanabe, O. (2000). Madaboost : A modification of adaboost. *In Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, COLT '00, pages 180–189, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Dos Santos, E., Sabourin, R. et Maupin, P. (2006). An evaluation of over-fit control strategies for multi-objective evolutionary optimization. *In Proceedings of the International Joint Conference on Neural Networks (IJCNN 2006)*, IJCNN 2006, pages 3070 – 3077, Washington, DC, USA. IEEE Computer Society.

Dosch, P. et Valveny, E. (2006). Report on the second symbol recognition contest. *In* Liu, W. et Lladós, J., éditeurs : *Graphics Recognition. Ten Years Review and Future Perspectives*, volume 3926 de *Lecture Notes in Computer Science*, pages 381–397. Springer Berlin / Heidelberg.

Duda, R. O., Hart, P. E. et Stork, D. G. (2000). *Pattern Classification (2nd Edition)*. Wiley-Interscience.

Dujet, C. et Vincent, N. (1998). Feature selection for classification. *International journal of intelligent system*, 13:131–156.

Duval, B. et Hao, J.-K. (2009). Advances in metaheuristics for gene selection and classification of microarray data. *Briefings in Bioinformatics*.

Efron, B. et Tibshirani, R. J. (1993). *An Introduction to the Bootstrap*. Chapman & Hall, New York.

Ferri, F. J., Kadirkamanathan, V. et Kittler, J. (1993). Feature subset search using genetic algorithms. *In Workshop on Natural Algorithms in Signal Processing, IEE*. Press.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals Eugen.*, 7:179–188.

Fleiss, J. L., Levin, B., Paik, M. C. et Fleiss, J. (2003). *Statistical Methods for Rates & Proportions*. Wiley-Interscience, 3rd édition.

Fonseca, C. et Fleming, P. (1993). Genetic algorithms for multiobjective optimization : Formulation discussion and generalization. *In Proceedings of the 5th International Conference on Genetic Algorithms*, pages 416–423, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Fraser, A. M. et Swinney, H. L. (1986). Independent coordinates for strange attractors from mutual information. 33.

Freund, Y. (1999). An adaptive version of the boost by majority algorithm. *In Proceedings of the twelfth annual conference on Computational learning theory*, COLT '99, pages 102–113, New York, NY, USA. ACM.

Freund, Y. et Schapire, R. E. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. *In Proceedings of the Second European Conference on Computational Learning Theory*, pages 23–37, London, UK. Springer-Verlag.

Friedman, J., Hastie, T. et Tibshirani, R. (1998). Additive logistic regression : a statistical view of boosting. *Annals of Statistics*, 28:2000.

Furey, T. S., Cristianini, N., Duffy, N., Bednarski, D. W., Schummer, M. et Haussler, D. (2000). Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16(10):906–914.

Gadat, S., Laurent, Y. et Guyon, I. (2007). A stochastic algorithm for feature selection in pattern recognition. *Journal of Machine Learning Research*, 8:8.

Goldberg et David, E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st édition.

Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A. et Bloomfield, C. D. (1999). Molecular classification of cancer : class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537.

Grandvalet, Y. (2004). Bagging equalizes influence. *Machine Learning*, 55:251–270.

Guyon, I. (2003). Design of experiments of the nips 2003 variable selection benchmark. technical report, 2003. http ://www.nipsfsc.ecs.soton.ac.uk/papers/datasets.pdf.

Guyon, I. et Elisseeff, A. (2003). An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182.

Guyon, I., Weston, J., Barnhill, S. et Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Mach. Learn.*, 46:389–422.

Hastie, T., Tibshirani, R. et Friedman, J. (2001). *The Elements of Statistical Learning. Springer series in statistics*. Springer, New York.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20:832–844.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, Michigan.

Huang, C.-J., Yang, D.-X. et Chuang, Y.-T. (2008). Application of wrapper approach and composite classifier to the stock trend prediction. *Expert Syst. Appl.*, 34:2870–2878.

Huang, J., Cai, Y. et Xu, X. (2007). A hybrid genetic algorithm for feature selection wrapper based on mutual information. *Pattern Recogn. Lett.*, 28:1825–1844.

Hugo, D. N., David, C., Thomas, D. et David, C. (199). Boosting naive-bayes classifiers to predict outcomes for hip prostheses. *International Joint Conference on Neural Networks*, pages 1 – 5.

Indyk, P. et Motwani, R. (1998). Approximate nearest neighbors : Towards removing the curse of dimensionality. pages 604–613.

Ishibuchi, H. et Nakashima, T. (2000). Multi-objective pattern and feature selection by a genetic algorithm. *In Proc. of Genetic and Evolutionary Computation Conference (GECCO'2000*, pages 1069–1076. Morgan Kaufmann.

Jain, A. et Zongker, D. (1997). Feature selection : Evaluation, application, and small sample performance. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 19:153–158.

Jensen, F. V. (2001). *Bayesian Networks and Decision Graphs*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

John, G. H. (1997). *Enhancements to the data mining process*. Thèse de doctorat, Stanford, CA, USA. UMI Order No. GAX97-23376.

John, G. H., Kohavi, R. et Pfleger, K. (1994). Irrelevant features and the subset selection problem. *In MACHINE LEARNING : PROCEEDINGS OF THE ELEVENTH INTERNATIONAL*, pages 121–129. Morgan Kaufmann.

Jolliffe, I. T. (1986). *Principal Component Analysis*. Springer-Verlag.

Kachouri, R., Djemal, K. et Maaref, H. (2010). Adaptive feature selection for heterogeneous image databases. *In* Djemal, K. et Deriche, M., éditeurs : *Second IEEE International Conference on Image Processing Theory, Tools 38 ; Applications, 10*, Paris, France.

Kalousis, A., Prados, J. et Hilario, M. (2007). Stability of feature selection algorithms : a study on high-dimensional spaces. *Knowledge and Information Systems*, 12:95–116. 10.1007/s10115-006-0040-8.

Kim, H., Kim, J., Sim, D. et Oh, D. (2000). A modified zernike moment shape descriptor invariant to translation rotation and scale for similarity-based image retrieval. *In ICME00*, page MP5.

Kira, K. et Rendell, L. A. (1992). The feature selection problem : Traditional methods and a new algorithm. *In AAAI*, pages 129–134, Cambridge, MA, USA. AAAI Press and MIT Press.

Kitoogo, F. E. et Baryamureeba, V. (2007). A methodology for feature selection in named entity recognition. *International Journal of Computing and ICT*, pages 18–26.

KITTLER, J. (1978). Feature set search algorithms. *Pattern Recognition and Signal Processing*, pages 41–60.

KOHAVI, R. et JOHN, G. H. (1997). Wrappers for feature subset selection. *Artif. Intell.*, 97:273–324.

KOLLER, D. et SAHAMI, M. (1996). Toward optimal feature selection. pages 284–292. Morgan Kaufmann.

KRIZEK, P., KITTLER, J. et HLAVÁČ, V. (2007). Improving stability of feature selection methods. *In* KROPATSCH, W., KAMPEL, M. et HANBURY, A., éditeurs : *Computer Analysis of Images and Patterns*, volume 4673 de *Lecture Notes in Computer Science*, pages 929–936. Springer Berlin / Heidelberg.

KRZANOWSKI, W. et PARTRIDGE, D. (1996). Software diversity : Practical statistics for its measurement and exploitation. *Information and Software Technology*, 39:39–707.

KUDO, M. et SKLANSKY, J. (2000). Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition*, 33(1):25 – 41.

KUNCHEVA, L. et WHITAKER, C. (2001). Ten measures of diversity in classifier ensembles : limits for two classifiers. *IEE Seminar Digests*, 2001(50):10–10.

KUNCHEVA, L. I. (2004). *Combining Pattern Classifiers : Methods and Algorithms*. Wiley-Interscience.

KUNCHEVA, L. I. et JAIN, L. C. (1999). Nearest neighbor classifier : Simultaneous editing and feature selection.

LEARDI, R. (1994). Application of a genetic algorithm to feature selection under full validation conditions and to outlier detection. *Journal of Chemometrics*, 8(1):65 – 79.

LEBART, L. et MORINEAU, A. (1995). *Statistique exploratoire multidimensionnelle*. Dunod-Paris.

LECUN, Y., BOTTOU, L., BENGIO, Y. et HAFFNER, P. (1998). Gradient-based learning applied to document recognition. *In Proceedings of the IEEE*, pages 2278–2324.

LI, Y. et GUO, L. (2008). Tcm-knn scheme for network anomaly detection using feature-based optimizations. *In Proceedings of the 2008 ACM symposium on Applied computing*, SAC '08, pages 2103–2109, New York, NY, USA. ACM.

LIU, H., MOTODA, H. et YU, L. (2002). Feature selection with selective sampling. *In Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, pages 395–402, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Liu, H. et Setiono, R. (1996). Feature selection and classification - a probabilistic wrapper approach. *In in Proceedings of the 9th International Conference on Industrial and Engineering Applications of AI and ES*, pages 419–424.

Liu, H. et Setiono, R. (1998). Incremental feature selection. *Applied Intelligence*, 9:217–230.

Liu, H. et Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, 17:491–502.

Maldonado, A., Doucet, J., Petitjean, M. et Fan, B.-T. (2006). Molecular similarity and diversity in chemoinformatics : From theory to applications. *Molecular Diversity*, 10:39–79.

Marichal, J.-L. (2000). *On Choquet and Sugeno Integrals as Aggregation Functions*. Physica Verlag, Heidelberg.

Marichal, J.-L. (2002). *Aggregation of interacting criteria by means of the discrete Choquet integral*, pages 224–244. Physica-Verlag GmbH, Heidelberg, Germany, Germany.

Marill, T. et Green, D. M. (1963). On the effectiveness of receptors in recognition systems. *IEEE transactions on Information Theory*, 9:11–17.

Melville, P. et Mooney, R. J. (2004). Creating diversity in ensembles using artificial data. *Journal of Information Fusion : Special Issue on Diversity in Multi Classifier Systems*, 6(1):99–111.

Messick, S. et Abelson, R. (1956). The additive constant problem in multidimensional scaling. *Psychometrika*, 21:1–15.

Michalewicz, Z. (1992). *Genetic algorithms + data structures = evolution programs*. Springer.

Miller, B. L. et Goldberg, D. E. (1995). Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems*, 9:193–212.

Murphey, Y., Zhihang, C. et Hong, G. (2001). Neural learning using adaboost. *International Joint Conference on Neural Networks*, 2:1037 – 1042.

Narendra, P. M. et Fukunaga, K. (1977). A branch and bound algorithm for feature subset selection. *IEEE Trans. Comput.*, 26:917–922.

NAVIDOMASS (2007-2010). Navigation into documents masses,projet anr-06-mdca-012 de sauvegarde et d'indexation du patrimoine historique français, http ://l3iexp.univ-lr.fr/navidomass/.

Neapolitan, R. E. (2003). *Learning Bayesian Networks*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Oliveira, L. S., Sabourin, R., Bortolozzi, F. et Suen, C. Y. (2002). Feature selection using multi-objective genetic algorithms for handwritten digit recognition. *In Proceedings of the 16 th International Conference on Pattern Recognition (ICPR'02) Volume 1 - Volume 1*, ICPR '02, Washington, DC, USA. IEEE Computer Society.

Pareti, R. et Vincent, N. (2006). Ancient initial letters indexing. *In ICPR (2)*, pages 756–759.

Peng, H., Long, F. et Ding, C. (2005). Feature selection based on mutual information : criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:1226–1238.

Pudil, P., Novovičová, J. et Kittler, J. (1994). Floating search methods in feature selection. *Pattern Recogn. Lett.*, 15:1119–1125.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1:81–106.

Quinlan, J. R. (1993). *C4.5 : programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Radtke, P. V. W., Wong, T. et Sabourin, R. (2006). An evaluation of over-fit control strategies for multi-objective evolutionary optimization. *In Proceedings of the International Joint Conference on Neural Networks (IJCNN 2006) Volume 1 - Volume 01*, IJCNN 2006, pages 6359–6366, Washington, DC, USA. IEEE Computer Society.

Randić, M. (1996). Molecular bonding profiles. *Journal of Mathematical Chemistry*, 19:375–392. 10.1007/BF01166727.

Raymer, M. L., Punch, W. F., Goodman, E. D., Kuhn, L. A. et Jain, A. K. (2000). Dimensionality reduction using genetic algorithms.

Raymer, M. L., Punch, W. F., Goodman, E. D., Sanschagrin, P. C. et Kuhn, L. A. (1997). Simultaneous feature extraction and selection using a masking genetic algorithm. *In In Proceedings of the 7 th International Conference on Genetic Algorithms*, pages 561–567. Morgan Kaufmann.

Ritthoff, O., Klinkenberg, R., Fischer, S. et Mierswa, I. (2002). A hybrid approach to feature selection and generation using an evolutionary algorithm. *In In Proc. 2002 U.K. Workshop on Computational Intelligence (UKCI-02*, pages 147–154. University of.

Roweis, S. T. et Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *SCIENCE*, 290:2323–2326.

Salton, G. et McGill, M. J. (1986). *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA.

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, 5:197–227.

Sebban, M. et Suchier, H.-M. (2003). Étude sur l'amélioration du boosting : réduction de l'erreur et accélération de la convergence. *Journal électronique d'intelligence artificielle*.

Siedlecki, W. et Sklansky, J. (1993). Handbook of pattern recognition & computer vision. chapitre A note on genetic algorithms for large-scale feature selection, pages 88–107. World Scientific Publishing Co., Inc., River Edge, NJ, USA.

Sikora, R. et Piramuthu, S. (2007,). Framework for efficient feature selection in genetic algorithm based data mining. *European Journal of Operational Research*, 180(2):723–737.

Skurichina, M. et Duin, R. P. W. (1998). Bagging for linear classifiers. *Pattern Recognition*, 31:909–930.

Somol, P., Pudil, P. et Kittler, J. (2004). Fast branch & bound algorithms for optimal feature selection. *IEEE Pattern Analysis and Machine Intelligence*, 26:900–912.

Srinivas, N. et Deb, K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2:221–248.

Tabbone, S. et Wendling, L. (2003). Binary shape normalization using the Radon transform. *In 11th International Conference on Discrete Geometry for Computer Imagery - DGCI'2003*, volume 2886 de *Lecture Notes in Computer Science*, Naples, Italy. Springer. Colloque avec actes et comité de lecture. Internationale.

Tenenbaum, J. B., Silva, V. d. et Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323.

Thomas, J., Jouve, P.-E. et Prudhomme, E. (2008). Échantillonnage adaptatif de jeux de données déséquilibrés pour les forêts aléatoires. *In EGC*, pages 213–214.

Todeschini, R. et Consonni, V. (2000). *Handbook of Molecular Descriptors*, volume 11. Wiley-VCH.

Tremblay, G., Sabourin, R. et Maupin, P. (2004). Optimizing nearest neighbour in random subspaces using a multi-objective genetic algorithm. *In Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 1 - Volume 01*, ICPR '04, pages 208–, Washington, DC, USA. IEEE Computer Society.

Tusher, V. G., Tibshirani, R. et Chu, G. (2001). Significance analysis of microarrays applied to the ionizing radiation response. 98.

Vapnik, V. (1998). *Statistical Learning Theory*. John Wiley & Sons.

Vezhnevets, A., Vezhnevets, V. et Vezhnevets, V. (2005). Modest adaboost : Teaching adaboost to generalize better. *Graphicon-2005, Novosibirsk Akademgorodok, Russia*.

WHITNEY, A. W. (1971). A direct method of nonparametric measurement selection. *IEEE Trans. Comput.*, 20:1100–1103.

YANG, J. et HONAVAR, V. (1998). Feature subset selection using a genetic algorithm. *IEEE Intelligent Systems and their Applications*, 13(2):44–49.

YULE, G. U. (1900). On the association of attributes in statistics. volume 194 de *A*, page 257–319. Philosophical Transactions of the Royal Society of London.

ZERNIKE, F. (1934). Diffraction theory of the cut procedure and its improved form the phase contrast method. *Physica*, 1:689–704.

ZHANG, D. et LU, G. (2002). Shape based image retrieval using generic fourier descriptors. *In Signal Processing : Image Communication 17*, pages 825–848.

ZIPF, G. K. (1949). *Human Behavior and the Principle of Least Effort.* Addison-Wesley (Reading MA).

Publications

Chapitre de livre

Hassan Chouaib, Florence Cloppet and Nicole Vincent. *Graphical Drop Caps Indexing.* Graphics Recognition. Achievements, Challenges, and Evolution. Lecture Notes in Computer Science, 2010

Conférences et workshops internationaux

Hassan Chouaib, Florence Cloppet, Salvatore Tabbone and Nicole Vincent. *Generic feature selection and document processing.* International Conference on Document Analysis and Recognition (ICDAR'2009) Barcelona-Spain

Mickaël Coustaty , Nicolas Sidère , Jean-Marc Ogier, Pierre Héroux , Jean-Yves Ramel, Hassan Chouaib , Nicole Vincent , Salim Jouili and Salvatore Tabbone. *Content-Based Old Documents Indexing.* International Workshop on Graphics Recognition - GREC 2009 , Larochelle-France

Hassan Chouaib, Salvatore Tabbone, Oriol Ramos, Florence Cloppet et Nicole Vincent. *Feature selection combining genetic algorithm and Adaboost classifiers.* International Conference on pattern recognation (ICPR) Floride 2008

Conférences et workshops nationaux

Hassan Chouaib, Salvatore Tabbone, Oriol Ramos, Florence Cloppet et Nicole Vincent. *Sélection de caractéristiques à partir d'un algorithme génétique et d'une combinaison de classifieurs Adaboost.* Colloque international Francophone sur l'Écrit et le Document (CIFED) Rouen 2008