ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN GÉNIE DE LA PRODUCTION AUTOMATISÉE
M.Ing.

PAR
LEVASSEUR, Yan

TECHNIQUES DE L'INTELLIGENCE ARTIFICIELLE POUR LA CLASSIFICATION D'OBJETS BIOLOGIQUES DANS DES IMAGES BIDIMENSIONNELLES

MONTRÉAL, LE 8 MAI 2008

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE

M. Jacques-André Landry, directeur de mémoire
Département de génie de la production automatisée à l'École de technologie supérieure

M. Robert Sabourin, président du jury
Département de génie de la production automatisée à l'École de technologie supérieure

M. Éric Granger, membre du jury
Département de génie de la production automatisée à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 23 AVRIL 2008

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

AVANT-PROPOS

Les récents succès des chercheurs dans le domaine de l'Intelligence Artificielle (IA) les poussent à s'attaquer à des défis de plus en plus grands comme l'analyse de systèmes complexes, dans l'espace et dans le temps, faisant intervenir un nombre souvent incalculable de paramètres. Bref, nous désirons amener la machine à modéliser et interpréter ce qu'humainement, nous ne pouvons accomplir.

Pourtant, il nous apparaît pertinent de se questionner sur l'objectif ultime de nos recherches. L'efficience de nos outils ne nous permet-elle pas déjà de mettre fin aux injustices sociales et environnementales, de désamorcer les situations critiques auxquelles l'humanité est confrontée ? La réponse à cette question se trouve, à mon avis, au plus profond de chacun de nous.

De notre côté, nous désirons axer nos recherches sur l'assemblage et l'application de méthodes existantes issues de la vision et de l'IA, avec l'espoir de faire progresser les outils d'analyse qui nous aiderons à apprécier la biodiversité et l'intégrité des écosystèmes, deux choses que nous considérons essentielles à la survie de plusieurs espèces - dont la nôtre - sur cette planète.

REMERCIEMENTS

En premier lieu, nous tenons à remercier *Jacques-André Landry* - directeur de recherche à la maîtrise et véritable ami - de m'avoir permis de continuer à évoluer en recherche universitaire. Bien que la découverte d'un intérêt particulier à la recherche provient principalement de la période passée avec messieurs Jean-Christophe Demers, Jean-Sébastien Lussier et David Rivest-Hénault (été 2003), c'est bien à M. Landry que je dois la chance d'approfondir ma connaissance de l'intelligence artificielle dans un milieu stimulant. De plus, son support aura été unilatéral et d'une convivialité extraordinaire tout au long de mon partenariat avec lui. Ces dernières années, j'ai remarqué à quel point ce genre de relation est peu commun dans le contexte de la recherche. Jacques-André est pour moi une inspiration à ne jamais oublier le côté humain, autant dans le travail que dans la vie quotidienne.

En deuxième lieu, je tiens à remercier Lucie Leduc, ma mère, d'avoir investi énormément de temps et d'avoir partagé ses compétences pour la révision de la langue et du format de ce mémoire.

En dernier lieu, je désire remercier toutes les personnes qui ont travaillé de près ou de loin à la réalisation du programme d'exploration des données Weka. Je félicite également la communauté internationale des logiciels libres pour ses milliers d'exemples de projets collaboratifs.

TECHNIQUES DE L'INTELLIGENCE ARTIFICIELLE POUR LA RECONNAISSANCE D'OBJETS BIOLOGIQUES DANS DES IMAGES BIDIMENSIONNELLES

LEVASSEUR, Yan

RÉSUMÉ

La reconnaissance visuelle d'objets biologiques comme les produits agro-alimentaires et les espèces végétales en milieu naturel a profité de percées majeures lors des 30 dernières années. Aujourd'hui, des algorithmes de reconnaissance performants sont utilisés pour évaluer la qualité de productions agricoles et faire le suivi d'écosystème pour en assurer la protection. Dans la plupart des cas, ce sont des experts en vision et en informatique qui ont développé les solutions personnalisées qui ont permis d'atteindre les résultats désirés.

L'objectif de la recherche présentée est de fournir des recommandations pour le développement d'un algorithme de reconnaissance de formes dans des images qui soit générique et qui nécessite le moins d'intervention humaine possible. Un tel algorithme pourrait être utilisé par les non experts, par exemple les ingénieurs en agro-alimentaire, les botanistes et les biologistes. Pour atteindre notre objectif, nous avons étudié les étapes du processus de reconnaissance à partir d'images.

En pratique, nous avons mis sur pied un système de segmentation, d'extraction de caractéristiques et de classification, en plus d'avoir développé un algorithme de Programmation Génétique (PG). Nous avons intégré ce dernier au logiciel libre d'exploration des données Weka, afin d'encourager la mise en commun des efforts de recherche sur les algorithmes évolutionnaires.

Les classificateurs utilisés pour nos expérimentations sont le classificateur naïf de Bayes, l'arbre de décision C4.5, le K Plus Proche Voisins (KPPV), la PG, le Séparateur à Vastes Marges (SVM) et le Perceptron Multicouche (PM). Dans une seconde série d'expériences, nous avons combiné chacun de ces classificateurs, à l'exception du KPPV, au méta algorithme de boosting.

Nous avons comparé les résultats de classification des six algorithmes choisis pour six bases de données distinctes dont trois ont été créées par nous. Les bases utilisées proviennent d'images de céréales, de grains de pollen, de nœuds de bois, de raisins secs, de feuilles d'arbres et de caractères de l'alphabet romain.

Suite à une segmentation des bases d'images, nous avons extrait, à partir de chaque objet, une quarantaine de caractéristiques. Une base de données alternatives à été créée grâce à la transformation par Analyse en Composante Principale (ACP). Finalement, nous avons compilés les résultats de classification des six classificateurs, puis de leur combinaison par boosting pour les caractéristiques de base et pour les caractéristiques transformées. Chaque

expérience a été réalisée 50 fois avec une séparation aléatoire des bases d'apprentissage et de test.

Nous avons observé de bonnes performances de classification pour les problèmes comportant un grand nombre d'échantillons d'apprentissage. La performance des classificateurs, selon leur taux d'erreur médian, est consistante pour la plupart des bases de données. Le PM et le SVM obtiennent généralement les meilleurs taux de classification. Pour les problèmes comportant un grand nombre d'échantillons, notre approche obtient des résultats encourageants.

Malgré la supériorité apparente de certains classificateurs, nos expérimentations ne nous permettent pas d'émettre une recommandation sur l'utilisation prioritaire d'un classificateur dans tous les cas. Nous suggérons plutôt l'utilisation d'une métaheuristique évolutive pour l'analyse des données d'un problème afin de choisir ou de combiner des classificateurs appropriés. Nous avançons également que la performance de classification de notre approche pourrait être améliorée par l'ajout de nouvelles caractéristiques pertinentes, puis par l'optimisation des paramètres des classificateurs en fonction des données.

ARTIFICIAL INTELLIGENCE TECHNIQUES FOR BIOLOGICAL OBJECT CLASSIFICATION IN BIDIMENSIONAL IMAGES

LEVASSEUR, Yan

ABSTRACT

The visual recognition of biological objects such as food products and plant material in natural environment benefited from major openings during the last 30 years. Today, powerful recognition algorithms are used to evaluate the quality of food productions and to monitor ecosystems to ensure its protection. In the majority of the cases, vision and data processing experts developed customized solutions which allowed reaching the desired results.

The goal of this research is to provide recommendations for the development of a generic object recognition system (from images) which would require as little human intervention as possible. Such an algorithm could be used by non experts such as industrial engineers, botanists and biologists. To this end, we studied the stages of the recognition process starting from images.

In practice, we designed a system for segmentation, feature extraction and classification. In addition, we developed a Genetic Programming (GP) classifier. We integrated the GP algorithm to the free and open source data-mining software Weka to support collaborative research efforts in evolutionary computing.

Six different classifiers were used for our experiments. They are the naïve Bayes, C4.5 decision tree, K Nearest Neighbour (KNN), GP, Support Vector Machine (SVM) and Multilayered Perceptron (MP). In a second round of experiments, we combined each of these classifiers (except for KNN) with the boosting meta-algorithm.

We compared the classification results from the six algorithms for six distinct databases of which we created three. The databases contain information extracted from images of cereals, pollen, wood knots, raisins, tree leaves and computer generated roman characters.

We automatically segmented the majority of the images. We then extracted nearly 40 features from each object. We obtained an alternate database using a transformation by Principal Component Analysis (PCA). Finally, we compiled the classification results of the six classifiers, then of their combination with boosting for the basic feature set and for the transformed set. Each experiment was carried out 50 times, with a random separation of the training and test databases.

We observed good recognition rates for problems comprising a large number of training samples. The performance ranking of the classifiers, according to their median error rate, is consistent for the majority of the problems. The MP and the SVM generally obtain the best classification rates. For problems containing a large number of samples, our approach obtained encouraging results.

In spite of the apparent superiority of some classifiers, the experiments do not enable us to put forth a recommendation on the priority use of a specific classifier in all cases. We rather suggest the use of an evolutionary meta-heuristic for the analysis of a problem's data in order to choose or to combine suitable classifiers. We also put forward that our approach's classification performance could be improved with the addition of new relevant features and the optimization of the classifiers' parameters according to the data.

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ACP	Analyse en Composantes Principales
AM	Apprentissage Machine
C4.5	Algorithme utilisé pour générer un arbre de décision
ED	Exploration des données
FDA	Fonction Définie Automatiquement
IA	Intelligence Artificielle
ID3	« Iterative Dichotomizer 3 »
KDD	Découverte de savoir dans les bases de données, de l'anglais « Knowledge Discovery in Databases »
KPPV	K Plus Proches Voisins
LBP	Forme binaire locale, de l'anglais « Local Binary Pattern »
LIVIA	Laboratoire d'Imagerie, de Vision et d'Intelligence Artificielle
PM	Perceptron Multicouche
PG	Programmation Génétique
PGB	Programmation Génétique avec Boosting intégré
RF	Reconnaissance de Formes
RNA	Réseau de Neurones Artificiel
RVB	Modèle de couleur Rouge Vert Bleu
SVM	Séparateur à Vastes Marges
SVM2	Séparateur à Vastes Marges avec fonction de noyau polynomial du deuxième degré
TFD	Transformée de Fourier Discrète
UML	« Unified Modeling Language »

INTRODUCTION

Le domaine de l'Intelligence Artificielle (IA) démontre les capacités extraordinaires des ordinateurs modernes lorsqu'un problème concret est modélisé avec une précision numérique suffisante. La localisation automatique d'un robot (Groszmann et Poli, 2001), le contrôle de cellules de soudures robotisées (Smartt, Kenney et Tolle, 2002) ou même la reconnaissance de l'écriture manuscrite (Côté et al., 1998) peuvent aujourd'hui être automatisés avec de l'équipement industriel relativement commun.

Ces résultats impressionnants peuvent aussi être observés en agro-alimentaire, en botanique et en biologie, lorsque la vision par ordinateur et des algorithmes de classification sont utilisés pour reconnaître un objet d'origine végétale à partir d'une image numérique. Notre groupe de recherche s'intéresse à ces applications et plus particulièrement à l'identification d'objets biologiques que l'on retrouve en agriculture et en agro-alimentaire, pour l'évaluation de la qualité d'un lot de produits, puis à l'identification d'espèces en milieu naturel pour le suivi et la protection des écosystèmes.

Les ingénieurs en production manufacturière et en agro-alimentaire, les botanistes et les biologistes n'ont généralement pas suivi la formation nécessaire pour utiliser les outils spécialisés récemment développés par les chercheurs du domaine de la reconnaissance de formes. Un outil automatisé générique, obtenant une performance suffisante pour une première analyse dans le cas d'un problème de reconnaissance d'objet dans une image pourrait se révéler très utile : une personne sans connaissance dans le domaine pourrait l'utiliser directement puisque l'outil nécessiterait un minimum d'interventions humaines.

L'objectif de ce projet de maîtrise est d'évaluer les outils à notre disposition pour automatiser le processus de reconnaissance d'objets biologiques. Nous désirons ainsi fournir des recommandations pour le développement d'un outil de reconnaissance automatisé fonctionnant, au possible, indépendamment de l'usager, donc des connaissances d'un expert humain.

Afin de réaliser un algorithme générique de reconnaissance d'objets à partir d'image, nous devons d'abord maîtriser toutes les étapes de la reconnaissance de formes, à partir des images jusqu'à la classification des objets qui la composent. Les étapes majeures sont la segmentation, l'extraction des caractéristiques, la sélection des caractéristiques et la classification. Plusieurs méthodes peuvent être utilisées à chaque étape. Nous devrons donc étudier les méthodes de références, comprendre leur fonctionnement et les utiliser pour la reconnaissance d'objets dans le cadre de plusieurs problèmes différents.

Le développement d'un algorithme générique, produit d'une grande complexité, requiert un travail de longue haleine qui sera réalisé par plusieurs étudiants et chercheurs, pendant plusieurs années. Dans cette optique, les algorithmes implémentés dans le cadre de ce projet seront distribués selon une licence ouverte, de façon à favoriser les contributions à l'atteinte de notre objectif.

Ce mémoire est divisé en cinq chapitres. À la toute fin, nous présentons la conclusion, puis les recommandations et annexes.

Le premier chapitre situe l'étude dans son contexte, soit par rapport aux récentes percées scientifiques quant à la reconnaissance d'objets biologiques dans des images. On y présente ensuite les algorithmes évolutionnaires, suivis de notre point de vue sur la problématique d'outil générique. Finalement, notre objectif global est séparé en objectifs plus spécifiques, puis nos hypothèses de départ sont présentées.

La revue de littérature est présentée au chapitre 2. On y aborde les thèmes d'exploration des données, de reconnaissance de formes, de vision par ordinateur et de prétraitement des données. Suivent quelques méthodes pour l'extraction de caractéristiques et des algorithmes de classification. Nous clarifions aussi le fonctionnement de deux méta algorithmes pour la classification. Finalement, nous présentons la plateforme de développement pour l'exploration des données Weka.

Le chapitre 3 fait état de nos démarches pour la réalisation d'un algorithme de programmation génétique. Nous discutons d'abord de sa conception, puis de son déroulement dans le cas général. Nous présentons ensuite les différents paramètres globaux et génétiques de l'algorithme. L'intégration de l'algorithme dans l'environnement de Weka termine le chapitre.

Dans le chapitre 4, nous dévoilons le protocole expérimental. Nous présentons d'abord les bases de données d'images utilisées, puis les méthodes de segmentation et d'extraction de caractéristiques. Nous abordons ensuite la séparation des bases de données en vue des tests et énumérons les expériences à réaliser avec les classificateurs et méta algorithmes choisis. Finalement, nous précisons la méthode de représentation des résultats.

Le chapitre 5 contient les résultats de nos expérimentations et leur analyse. Les performances de reconnaissance et temps d'apprentissage y sont dévoilés. Une discussion sur l'ensemble des observations complète ce chapitre.

La conclusion présente une synthèse ce qui a été réalisé dans ce projet. Les recommandations finales proposent des avenues pour la poursuite de la recherche selon les objectifs que nous avons définis.

CHAPITRE 1

CONTEXTE DE L'ÉTUDE

Ce chapitre permet de mieux cerner le contexte de notre recherche en détaillant les particularités des objets biologiques et les méthodes utilisées jusqu'à présent pour leur reconnaissance. De plus, nous y détaillons la problématique, spécifions les objectifs et présentons nos hypothèses de départ.

1.1 Les objets biologiques et irréguliers

Le système de perception des vivants leur permet, selon leur nature, d'identifier la présence de nourriture, d'abri, de partenaire, d'ennemi ou de tout autre objet dont la reconnaissance pourra être utile à leur survie. Des informations combinées, issues des organes de perception, permettent aux êtres vivants de classifier et d'identifier l'objet de leurs expériences sensorielles. Cette opération de reconnaissance s'avère possible pour un éventail d'objets différents extrêmement large. Ce processus relève de systèmes complexes, biologiques et évolutifs, qui ont permis à une espèce d'apprendre à reconnaître certaines choses, à les différencier.

Il est extrêmement utile de reproduire cette faculté à l'aide d'ordinateurs ou de machines, afin d'automatiser les multiples tâches reliées au monde humain et biologique de l'identification d'objet. Les actions de reconnaître un visage ou des paroles, de différencier une pomme d'une orange ou un chien d'un chat, correspondent à des comportements d'une grande banalité pour un humain adulte. Cependant, la modélisation numérique de la perception de ces objets ne peut prétendre à la même exactitude que l'action de la reconnaissance humaine, car la définition et la variabilité des objets sont intrinsèques au processus d'apprentissage et varient en fonction de l'objectif. L'automatisation de ces tâches de reconnaissance n'est donc possible qu'avec les outils que présentent des sciences en plein essor : la reconnaissance de formes, l'exploration des données et l'IA.

Les chercheurs actuels de l'IA s'intéressent particulièrement à la reconnaissance audio et vidéo (Camastra et Vinciarelli, 2008), technologies qui émulent, à l'aide de capteurs comme de micros et de cellules photosensibles, puis d'ordinateurs, les sens de l'ouïe et de la vue des êtres vivants (en particulier l'ouïe et la vue des êtres humains).

L'accessibilité grandissante des technologies de pointe comme les caméras numériques (*Voir* figure 1.1) rend les images numériques de plus en plus faciles à acquérir. Le plus souvent, les images contiennent toutes les informations nécessaires à la classification de l'objet.

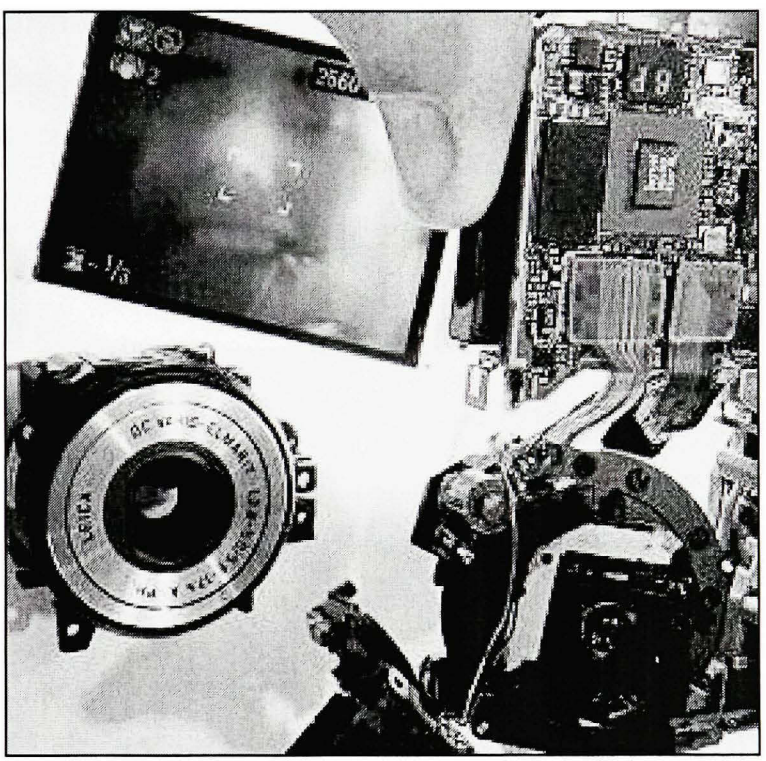


Figure 1.1 ***Caméra numérique partiellement démontée.***
(Tiré de (Wikipédia, 2007c))

Les objets biologiques constituent des exemples riches pour l'analyse numérique en raison des nombreuses caractéristiques (qui présentent naturellement une très grande variabilité) qu'ils fournissent pour l'échantillonnage. La variabilité et la définition changeante des objets biologiques justifient l'utilisation d'algorithmes d'apprentissage pour une reconnaissance adaptée à la réalité visuelle des objets. Le succès d'une analyse automatisée sur plusieurs

exemples issus du monde végétal peut assurer une reconnaissance similaire pour des objets plus réguliers, tels des objets manufacturés.

1.2 Assurance qualité en agro-alimentaire

Historiquement, la reconnaissance automatisée de plantes a été introduite par l'automatisation des tâches reliées à l'assurance qualité dans le domaine de la production agro-alimentaire. Les systèmes par caméras offrent certains avantages par rapport au travail de l'homme dans le travail d'inspection. En effet, l'humain est sujet à la fatigue et à la subjectivité, contrairement à son homologue machine (Chan et Braggins, 1996). Si l'homme réussit à observer des scènes complexes, il est toutefois dépassé par la rapidité et la régularité de l'inspection dans un contexte contrôlé.

Figure 1.2 ***Récolte de canneberges.***
(Tiré de (Wikipédia, 2004))

Des scientifiques utilisent des algorithmes pour évaluer la qualité des produits issus du champs, comme par exemple les canneberges (*Voir* figure 1.2) ou de l'usine de transformation, soit pour une évaluation en lot ou individuellement, puis occasionnellement pour classifier et sélectionner des produits lorsqu'ils se présentent mélangés (Chur chill, Bilsland et Cooper, 1993). Notre recherche nous a fait découvrir plusieurs méthodes particulièrement simples pour procéder à la classification de produits agro-alimentaires. Par

exemple : l'utilisation de l'aire d'un objet dans une image (Berlage, Cooper et Carone, 1984) et autres valeurs moyennes simples (Shatadal et al., 1995). Des chercheurs utilisent des méthodes pour l'évaluation et la classification d'une quantité impressionnante de fruits et légumes différents : les oignons (Van Der Heijden, Vossepoel et Polder, 1996), pistaches (Pearson et Slaughter, 1996), asperges (Rigney, Brusewitz et Kranzler, 1992), carottes (Howarth et Searcy, 1989), céréales (Visen, Jayas et White, 2004) et insectes se retrouvant dans le blé (Zayas et Flinn, 1998), pour ne nommer que ceux-ci.

Finalement, d'autres scientifiques ont développé des techniques procédant au positionnement (Wolfe et Swaminathan, 1987) et à la segmentation (Yang, 1996) des objets d'intérêt dans l'image. Ces méthodes permettent une reconnaissance des objets indépendamment de leur position dans l'image. Les algorithmes récents incluent des descripteurs numériques génériques pour la forme, la couleur (Majumdar et Jayas, 2000) et la texture (Visen, Jayas et White, 2004) pour faciliter la classification des objets photographiés par caméra.

1.3 Identification d'espèces végétales et biodiversité

Avec la progression des techniques propres au domaine de la vision artificielle et de la reconnaissance de formes, il est devenu plus aisé de créer des classificateurs génériques pour des objets irréguliers dans des images numériques. Ces technologies sont donc de plus en plus utilisées pour le développement d'applications diverses et moins industrielles, comme par exemple la reconnaissance d'espèces végétales en biologie et en écologie (Ye et al., 2004). On utilise aussi des systèmes de reconnaissance visuels respectivement pour l'identification de fleurs sauvages (Saitoh et Kaneko, 2000), de plantes (Du, Wang et Huang, 2004; Du, Wang et Zhang, 2007), et d'arbres (Wan et al., 2004).

La télédétection par avion et satellite, qui consiste à réaliser des images prises à grandes distances en utilisant une ou plusieurs bandes spectrales, est de plus en plus utilisée pour l'analyse géographique et pour observer l'évolution des écosystèmes forestiers, arctiques, désertiques, etc. De plus, certaines applications permettent de reconnaître des espèces

végétales en particulier (Dubrovin et al., 2000). Des recherches en télédétection ont permis de découvrir les besoins des plantes en azote (Chion, Da Costa et Landry, 2006), afin de contrôler avec précision la quantité d'intrant nécessaire pour la production agricole.

Ces recherches semblent présenter peu d'opportunités économiques à court terme, leur objectif étant beaucoup plus axé sur la protection de l'environnement que sur son exploitation. Pourtant, l'analyse de systèmes complexes comme les écosystèmes naturels nécessite des outils puissants tels ceux que l'on retrouve dans le domaine de l'exploration des données, par la présence de corrélation à multiples échelles (Brown et al., 2002). Récemment, des chercheurs ont soutenu que l'analyse par images numériques révélait des informations pertinentes relatives à la faune et la flore d'un environnement naturel et son évolution, jusqu'à fournir une indication fiable de la biodiversité de l'écosystème observé (Proulx et Parrott, 2007). Des images obtenues par télédétection (Crosse, 2005) ou simplement avec une caméra portable (Proulx, 2006) ont été utilisées pour démontrer la corrélation entre les données obtenues au moyen de l'image et la complexité structurelle d'un écosystème. Il apparaît aujourd'hui évident que la vision par ordinateur et l'exploration des données s'avèrent des outils d'avenir pour la compréhension et l'analyse de problèmes complexes, comme celui de l'analyse des milieux naturels (Shaanker, 2001).

1.4 Algorithmes évolutionnaires

Les données numériques correspondant aux descripteurs visuels d'un objet obtenus à partir d'une image peuvent être investiguées par un système d'exploration des données, mieux connu sous l'expression anglaise « Data Mining ». L'exploration des données, grâces à des outils visuels (*Voir* figure 1.3) et statistiques, mène à la sélection ou la combinaison des informations les plus significatives.

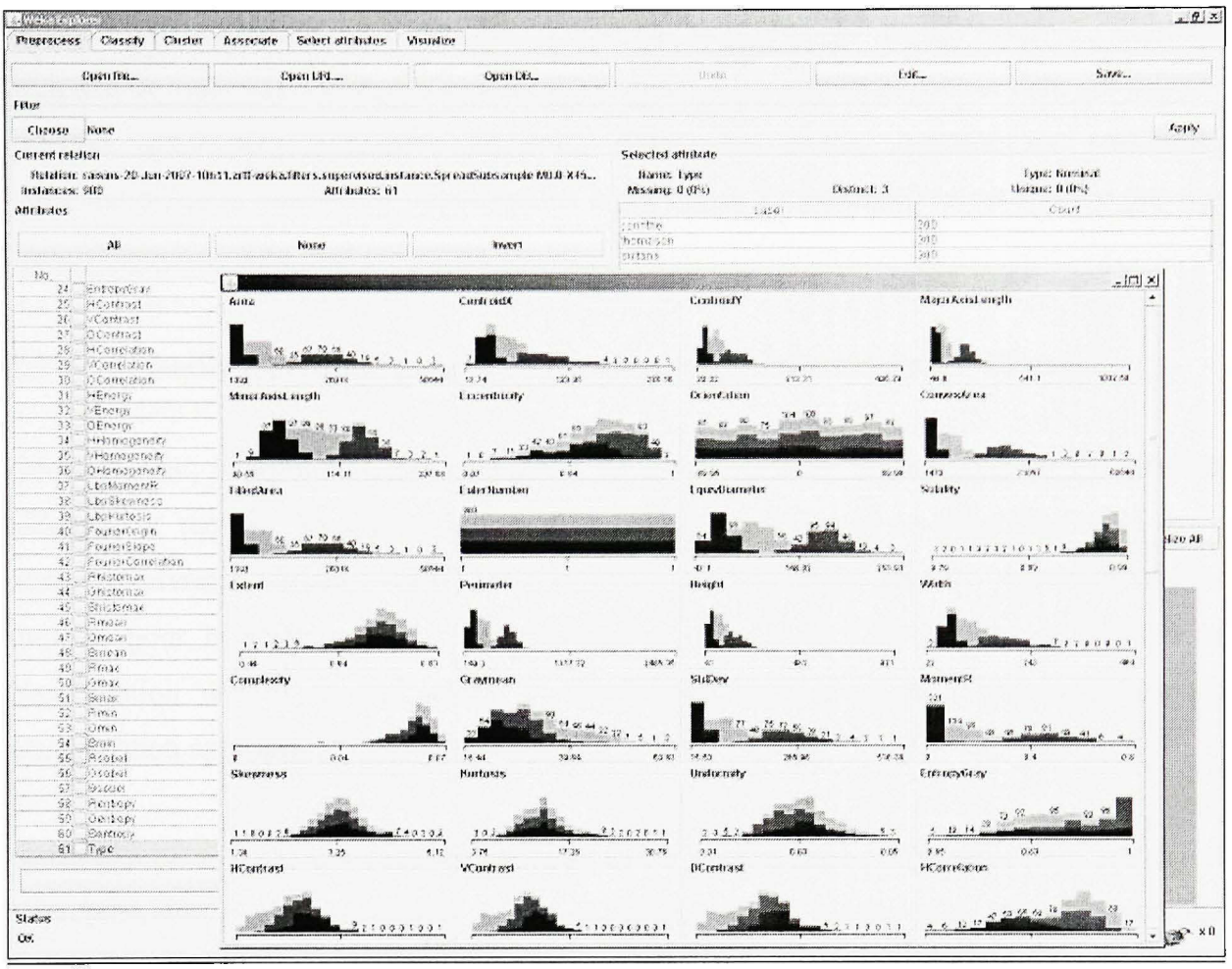


Figure 1.3 ***Exploration graphique des données.***

Source : Capture d'écran du logiciel Weka.

Afin de créer un système de prédiction, un algorithme appelé classificateur utilise les informations contenues dans une base de données d'apprentissage. Différentes approches ont été proposées pour le développement de classificateurs statistiques. Parmi celles-ci, nous nous intéressons particulièrement à l'approche évolutionnaire.

Les algorithmes évolutionnaires, de l'anglais « evolutionary algorithms », utilisent des mécanismes inspirés de la théorie de l'évolution pour résoudre des problèmes de toutes sortes. Ils font évoluer conjointement plusieurs solutions pour le problème qui leur est présenté dans le but de trouver les résultats les plus performants. Ce sont des algorithmes

stochastiques puisqu'ils utilisent des processus aléatoires de façon itérative. L'intérêt principal de ces algorithmes est leur faculté à éviter les maximums locaux dans un espace-solution quasi-infini (Langdon et Poli, 2002). La grande quantité de ressources computationnelles nécessaire pour certains algorithmes évolutionnaires est fournie par des ordinateurs de plus en plus puissants.

1.5 Outils génériques pour la reconnaissance de formes à partir d'images

Généralement, la résolution des problèmes de reconnaissance de formes pour l'inspection automatisée requiert les connaissances d'un expert en vision et en classification et ce, à plusieurs niveaux (Bernier, 1997). Son expertise est essentielle pour la segmentation des images, pour la sélection des caractéristiques à extraire de l'image afin de faciliter la classification des objets segmentés, et finalement, pour le choix d'un algorithme de classification approprié.

Notre groupe de recherche (qui fait partie du LIVIA), composée de Jacques-André Landry et ses étudiants, s'intéresse depuis plusieurs années à l'automatisation du processus de reconnaissance de forme pour des problèmes variés, notamment en agriculture et en écologie. Un des objectifs de notre groupe est de mettre sur pied un système qui pourra classifier l'objet étudié selon des critères qualitatifs afin de le départager selon l'espèce, la variété et le niveau de qualité (contrôle de qualité), à partir d'une image numérique. Cela pourra servir à estimer la valeur d'un lot ou d'une production agricole par exemple. On pourra également ainsi procéder à l'identification de plantes ou d'espèces spécifiques en milieu naturel. Finalement cet outil permettra d'analyser un environnement complexe et d'identifier des objets dont les caractéristiques sont hautement variables.

Les systèmes génériques ont l'avantage d'être utilisable par plus de gens que les systèmes spécialisés, qui demandent souvent une formation particulière. L'avantage d'un système générique de reconnaissance d'objet dans des images est donc de fournir des résultats pour la première analyse d'un problème de reconnaissance, sans avoir recours à un expert en vision

ou en classification. Ceci constitue un gain temporel et une réduction des coûts de développement pour une solution utilisant la vision artificielle et la reconnaissance de formes.

Dans ses recherches, notre groupe privilégie l'utilisation d'algorithmes d'apprentissage évolutionnaires, en particulier la Programmation Génétique (PG), pour leur faculté d'éviter les maximums locaux dans l'espace-solution. Dans un souci d'objectivité, les performances des méthodes utilisées pour nos expérimentations sont le plus souvent comparées à celles d'autres algorithmes couramment utilisés dans la littérature scientifique.

1.6 Objectifs

Le but de notre étude est de fournir des recommandations pour le développement d'une approche de reconnaissance de formes dans des images qui soit générique et qui nécessite le moins d'intervention humaine possible. Pour ce faire, nous nous proposons d'étudier les différentes étapes du processus de reconnaissance, à partir des images jusqu'à la classification des objets qu'elles contiennent. Nous investiguerons peu l'automatisation de la segmentation afin de concentrer nos efforts sur l'extraction de caractéristiques et le choix d'un classificateur générique, adapté à un grand nombre de problèmes.

Spécifiquement, nos objectifs sont les suivants :

1. Identifier un ensemble de caractéristiques génériques issues d'une image qui permettra de solutionner un large éventail de problèmes d'inspection automatisée d'objets biologiques, puis automatiser l'extraction de ces caractéristiques;

2. Évaluer la pertinence de méthodes de sélection de caractéristiques ou de réduction de la dimensionnalité de l'espace de caractéristiques développé en *1* en vue de la classification;

3. Maîtriser un algorithme existant ou développer un nouvel algorithme de PG pour la classification qui pourra être facilement comparé ou réutilisé par d'autres chercheurs;

4. Appliquer des algorithmes de classification couramment utilisés dans la littérature afin de classifier des objets biologiques et comparer les résultats en termes de taux de classification et de temps d'apprentissage pour plusieurs bases de données d'images;

5. Fournir des recommandations pour la conception d'un outil automatisé qui pourrait réaliser les étapes d'extraction des caractéristiques de l'image, de sélection et transformation des caractéristiques pertinentes, puis de classification.

1.7 Hypothèses de travail

Voici les hypothèses principales qui orientent notre étude de la reconnaissance de formes dans des images :

- Les problèmes de reconnaissance bien définis et suffisamment conscrits (présentant une petite variation intra classe et une grande variation inter classe) peuvent être résolus par une stratégie de prise de décision simple utilisant une représentation des formes compactes (Jain, Duin et Mao, 2000);

- Les progrès au niveau de l'algorithme de PG et la rapidité croissante des ordinateurs promettent un futur intéressant à la PG et aux algorithmes évolutionnaires (Banzhaf et al., 1998)

- La PG peut s'adapter à des situations variables. Son taux de bonne classification est généralement comparable à ceux d'algorithmes reconnus par la littérature, mais utilisent plus de ressources (Levasseur, 2005);

CHAPITRE 2

ÉTAT DE L'ART EN RECONNAISSANCE DE FORMES

Ce chapitre présente l'état de l'art en analyse de données numériques, plus particulièrement en reconnaissance de formes dans des images.

D'abord, une approche générale pour l'analyse de données et la découverte de patrons est présentée. Le cas spécifique d'exploration des données qui nous intéresse, la reconnaissance de formes, est ensuite décortiqué. Ensuite, nous discutons de l'organe de perception de notre système : un sous-système de vision par ordinateur. Les sections suivantes élaborent au sujet du traitement et de la transformation des données fournies par le système de vision. Nous présentons ensuite des algorithmes et méta algorithmes de classification. L'étude d'un module intégré de reconnaissance de formes dans des images complémente le chapitre. Nous terminons l'état de l'art avec la description d'une plate-forme d'exploration des données et de développement.

2.1 Exploration des données

Les bases de données ont été traitées et analysées manuellement jusqu'à une époque assez récente. Aujourd'hui, grâce à la capture de données automatisées, la quantité d'information que l'on peut obtenir est beaucoup trop grande pour que le tout soit scruté à la main. La taille des bases de données rend la tâche de reconnaissance de tendances ou de relations inter données très difficile pour un humain. Cette problématique a contribué au développement du domaine de la découverte de savoir dans les bases de données. Cette expression nous vient de la forme anglaise « Knowledge Discovery in Databases (KDD) » (Fayyad, Piatetsky-Shapiro et Smyth, 1996a). L'exploration des données (ED) - une autre traduction de l'anglais provenant de l'expression « Data Mining » - est une branche majeure de la découverte de savoir dans les bases de données (Witten et Frank, 2005). L'ED comprend des techniques et des algorithmes pour analyser de très grandes quantités de données et pour identifier les formes et les patrons qu'on y retrouve.

Habituellement, l'ED est divisée en trois grandes phases. Voici, de façon sommaire, ce que comporte chaque grande phase.

1. **Prétraitement** : cette étape concerne la définition même du problème et est accomplie avant que tout apprentissage ne soit réalisé;

    a. **Étude du domaine** : cette portion du prétraitement concerne l'étude de résultats similaires, de recherches sur le sujet ou de données connues à priori. Pour notre problème, cette étude est présentée à la section 2.3;

    b. **Préparation des données** : la préparation consiste à mettre les données sous une forme adaptée aux algorithmes d'exploration qui suivent dans le processus (*Voir 2* : phase d'exploration). On traite aussi le bruit et les données manquantes (cette sous étape est réalisée dans le cadre de notre recherche et sera traitée à la section 2.4). Les résultats obtenus sont appelés caractéristiques de base;

    c. **Transformation de l'espace des données** : ici la forme des données est transformée afin d'éliminer les caractéristiques non pertinentes, compresser les informations ou les combiner afin de faciliter le travail des algorithmes d'exploration en aval. Ce mécanisme sera traité à la section 2.5. Des procédés comme la sélection de caractéristiques et la construction de caractéristiques (ajout de caractéristiques construites à partir de celles de base (Liu et Motoda, 1998)) font partie de la dernière portion du prétraitement des données.

2. **Phase d'exploration** : dans cette phase, des algorithmes sont utilisés pour la découverte de relations et d'informations significatives à partir des données. Le but de la phase d'exploration peut varier selon la nature du problème. Si l'étude porte sur la compréhension des mécanismes ou de la structure de l'information, on pourra

recourir à une exploration dite descriptive. Si on veut automatiser un processus relié à l'analyse des données, on aura plus souvent recours à l'exploration prédictive;

a. **Exploration descriptive** : l'exploration mène à la visualisation ou la synthèse d'information sur les données. Afin de regrouper les données similaires et offrir des explications générales, on peut utiliser des algorithmes de partitionnement des données, appelés « Data Clustering algorithms » en anglais. Ces algorithmes créent des classes et étiquettent les données selon les similarités des échantillons analysés.

b. **Exploration prédictive** : ce type d'exploration est généralement utilisé pour automatiser la formulation de diagnostiques ou la prise de décision dans des procédés industriels ou informatiques. Deux types d'exploration prédictive sont plus connus : la classification et la régression. On parle de classification lorsque l'information à prédire doit faire partie d'un choix parmi un certain nombre de classes possibles (Weiss et Indurkhya, 1998), et c'est cette partie de l'exploration prédictive qui nous intéresse. (Nous présenterons plusieurs algorithmes de classification dans la section 2.6.) La régression concerne le cas où l'on désire prédire une valeur continue ou décimale. Plusieurs algorithmes de l'exploration prédictive peuvent s'adapter à la classification ou à la régression sans grand changement à leur fonctionnement (Witten et Frank, 2005).

3. **Post-traitements** : cette phase concerne les étapes qui sont entreprises après l'acquisition et l'exploration des données. Elle sera réalisée par des experts du domaine étudié, par exemple des spécialistes de la santé dans le cas d'anomalies détectées par rayons X, des biologistes dans le cas d'identification d'espèces, etc.

   a. **Évaluation des résultats** : cette portion du travail concerne l'évaluation de l'intérêt des patrons reconnus dans le but d'identifier ceux qui représentent du

« savoir ». Il s'agit de départager les relations issues d'un phénomène réel de celles qui sont dues à de simples variations dans les données d'entrées (Fayyad, Piatetsky-Shapiro et Smyth, 1996b). Plusieurs méthodes, comme la séparation de la base de données initiales en plusieurs bases d'apprentissage et de test, puis la validation croisée, de l'anglais « cross-validation » (Weiss et Kulikowski, 1991), sont utilisées pour confirmer la présence d'une forme chez des échantillons qui n'ont pas encore été observés par le système d'ED. Nous présenterons plus en détails une méthodologie pour évaluer les résultats à la section 4.8.

b. **Interprétation du savoir** : cette étape met en lumière les découvertes du processus d'ED et fait le pont entre ces nouvelles informations et le savoir existant concernant le domaine d'étude. Si les découvertes de l'ED ne peuvent pas être expliquées ou reliées au savoir existant du domaine étudié, il est possible que des erreurs se soient glissées dans l'ED. Il faut alors contre vérifier les résultats. Dans notre cas, cette portion du travail est présentée au chapitre 5.

c. **Transfert technologique** : la finalité de tout ce travail est bien entendu l'exploitation des découvertes du processus d'ED. Selon les cas, il est possible de les mettre en application immédiatement ou de procéder à des tests plus précis afin de valider les hypothèses qui découlent des résultats.

L'ED est un processus complet de recherche de savoir en mesure de traiter une grande quantité d'informations. Les phases que nous venons de présenter sont génériques et s'appliquent à des domaines très variés. Dans la perspective plus pratique de la reconnaissance à partir d'images, le domaine de la Reconnaissance de Formes (RF), dont de multiples chercheurs présentent des expérimentations poussées, suggère un parcours plus précis. On peut donc considérer que la RF est un cas particulier d'ED. Le processus de recherche de savoir dans la RF est spécifique au traitement, à la reconnaissance et à la

classification de patrons à partir de données numériques, comme par exemple celles qui peuvent être obtenues d'une image ou d'un enregistrement sonore.

2.2 Reconnaissance de formes

La RF est un domaine de l'apprentissage automatique ou Apprentissage Machine (AM), une traduction de l'anglais « Machine Learning ». La RF est communément définie comme une prise de décision basée sur l'obtention d'une catégorie à partir de données d'entrées brutes ou transformées (Duda, Hart et Stork, 2000).

L'objectif de la RF est donc de prendre la bonne décision, suite à l'observation d'un motif qui a été reconnu grâce à des caractéristiques particulières détectées par le système. Les systèmes de RF utilisent des connaissances à priori et / ou de l'information statistique sur les données pour identifier les patrons fournis en entrée. Pour la RF statistique, un échantillon est représenté par un vecteur de dimension égale au nombre de caractéristiques. Chaque échantillon peut appartenir à une catégorie de motifs, appelée classe.

La recherche en RF est généralement divisée en deux groupes de systèmes de classification : les méthodes supervisées et les méthodes non supervisées.

Les méthodes supervisées fonctionnent selon deux modes : entraînement (apprentissage) et classification (test). En premier lieu, les méthodes utilisent une base de données d'apprentissage qui présente des exemples dont la classe est déjà identifiée (elle aura été identifiée préalablement par un opérateur humain). Dans ce cas, le classificateur doit arriver à générer une fonction, un programme ou un ensemble de règles qui permettra d'obtenir la meilleure performance de classification pour les échantillons fournis en exemple. Éventuellement, le système pourra être éprouvé avec une base de données de test qui contient des patrons qui n'ont pas encore été observés, afin d'évaluer sa performance éventuelle sur des données inconnues. Cette dernière mesure est appelée performance de généralisation.

Les méthodes non supervisées peuvent aussi fonctionner selon un mode d'apprentissage ou de test, mais elles n'utilisent pas d'exemples préalablement identifiés. Les échantillons des bases de données ne font donc pas à priori partie d'une classe. Un système de RF obtenu sans supervision doit lui-même classifier les nouveaux échantillons présentés en fonction des classes qu'il aura élaborées à partir des patrons observés. Dans certains cas, le système de classification non supervisé devra aussi déterminer par lui-même le nombre de classes qui convient pour représenter un problème.

Un système de RF complet comprend le système de perception - soit les organes de prise de mesure -, le système d'extraction et de traitement des caractéristiques à partir des données brutes sous une forme numérique, puis un système de classification.

Il existe deux familles de classificateurs : les classificateurs statistiques et les classificateurs syntaxiques (aussi appelés classificateurs structurels). Les classificateurs statistiques utilisent les caractéristiques des échantillons (souvent représentés comme un vecteur de données numériques) pour identifier leurs classes. Ayant à sa disposition une base de données contenant des échantillons de chaque classe, le classificateur statistique a pour objectif de définir les frontières de décision qui permettent de différencier les classes dans l'espace des caractéristiques. La classification syntaxique est quant à elle basée sur l'analyse des interrelations entre caractéristiques. Grâce à cette qualité, la RF syntaxique peut arriver à classifier des données dont la forme est différente de celle d'un vecteur de caractéristiques. L'approche syntaxique fait usage de l'analogie du langage pour représenter les motifs, structurant les formes plus complexes en agençant les formes plus simples à l'aide d'une grammaire construite depuis l'analyse des formes fournies pour l'entraînement du système.

Les applications de la RF sont nombreuses. Leurs effets, extrêmement positifs, se font sentir, entre autres, dans les domaines de la reconnaissance automatique de la parole (Haton et al., 2006), la classification de texte et de document (Sebastiani, 2002), la reconnaissance de l'écriture manuscrite (Teredesai, Park et Govindaraju, 2001) et le développement

d'intelligence artificielle pour les jeux de stratégies (van der Werf, 2004) comme le complexe jeu de go (*Voir* figure 2.1).

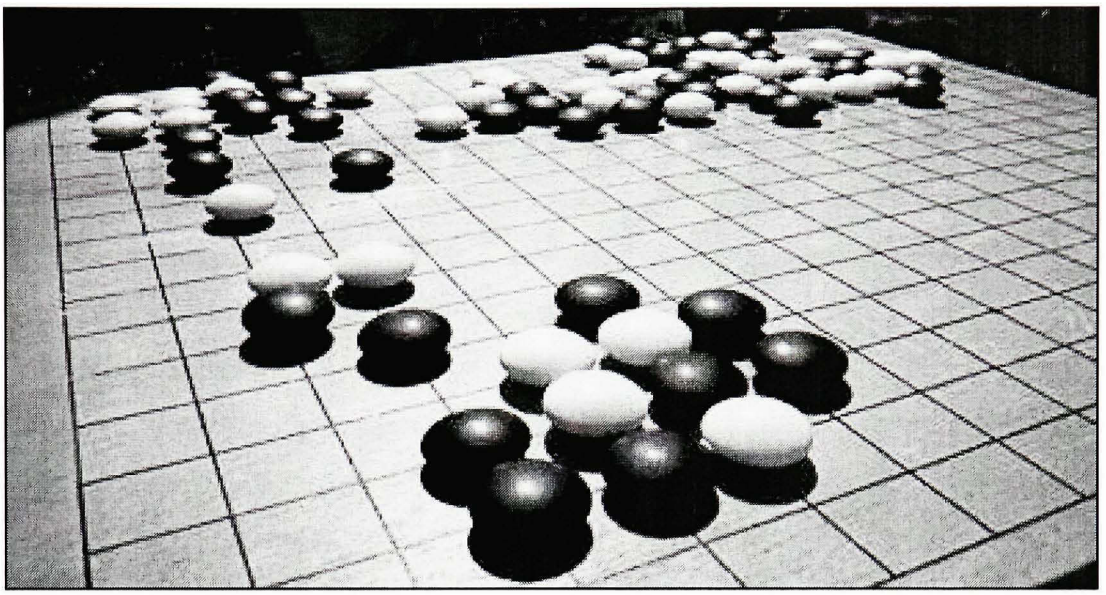


Figure 2.1 ***Jeu de go.***
(Tiré de (Wikipédia, 2005))

2.3 Vision par ordinateur et RF

Dans ce projet, nous voulons utiliser la RF afin d'automatiser la reconnaissance d'objets à partir d'information visuelle. À cette fin, nous devons intégrer un système de vision par ordinateur comme organe perceptif et de prétraitement de l'information comme porte d'entrée à notre système de RF.

L'organisation d'un système de vision par ordinateur peut être simple ou complexe, selon l'application. Généralement, il comprendra, entre autres, les fonctions suivantes (Gonzalez, Woods et Eddins, 2003) :

- **Acquisition de l'image** : une grande variété de capteurs peut produire des images digitales. On retrouve, par exemple, des caméras digitales (sensibles à différentes longueurs d'ondes), des caméras à ultrasons, des capteurs hyper spectraux, des

capteurs à rayons X, des radars, etc. Une image digitale est un bloc de données numériques qui contient l'intensité du signal capté à chaque position sur une « carte » dont la forme correspond à l'ensemble des capteurs. Chaque position sensible du capteur, où peut être stockée une valeur, est appelée pixel.

- **Prétraitement** : selon la qualité des images obtenues par l'organe d'acquisition, il est parfois nécessaire d'utiliser des algorithmes préliminaires de traitement de l'image. Ces algorithmes ont pour but d'améliorer la qualité visuelle de l'image ou de permettre l'extraction de certaines informations particulières qui seraient difficile à retrouver autrement. Par exemple, il arrive que l'on veuille réduire le bruit dans les images d'entrées, rehausser le contraste pour observer un contour d'objet ou transformer les données de l'image vers un autre système de représentation (de couleurs, par exemple).

- **Segmentation** : la segmentation est le travail de séparation des sections intéressantes de celles qui ne le sont pas dans l'image d'entrée. Ce travail peut être fait avant ou après une étape d'extraction de caractéristiques, selon la complexité de la tâche de segmentation. Généralement, des informations de bas niveau (telles les valeurs de pixels ou un histogramme de ces valeurs) sont suffisantes pour départager le fond de l'image des objets qui lui sont superposés. Si, parmi tous les objets détectés, seule une portion est importante, il peut être nécessaire de procéder à l'extraction de caractéristiques telles l'aire ou le périmètre afin de départager les zones d'intérêt. Après segmentation, seules les valeurs des pixels des zones d'intérêt sont conservées.

- **Extraction de caractéristiques développées par des experts en vision** : cette étape consiste à calculer des informations pertinentes pour la classification de l'objet segmenté, à partir des valeurs de pixels qui le composent. Il peut s'agir d'informations de différents niveaux de complexité. On peut par exemple soutirer des informations simples telles la présence de lignes, de courbes ou de contour. Des

informations de plus haut niveau concernant la forme, la texture et la couleur peuvent aussi être extraites.

La tâche de segmentation étant de difficulté variable et pouvant atteindre une très grande complexité, nous n'entrerons pas dans le détail de son automatisation pour un cas général. Pour l'application qui nous intéresse, il nous apparaît important de réduire au minimum toutes les opérations qui sont spécifiques à un problème en particulier. Le choix des caractéristiques à extraire doit être fait judicieusement pour permettre une bonne performance de reconnaissance pour des problèmes divers.

2.4 Prétraitement des données

Pour des cas de classification supervisée, les données sont le plus souvent représentées sous la forme d'un vecteur d'information, c'est à dire une liste de valeurs numériques ou faisant partie d'une énumération; la dernière valeur du vecteur étant le plus souvent la classe de l'échantillon. La classe de l'échantillon, qui fait partie d'un ensemble défini, correspond à un bon exemple de valeur d'énumération. Les valeurs numériques ou d'énumération sont appelées caractéristiques.

La quantité d'information provenant d'une image numérique peut être considérable. Les données brutes obtenues d'une image numérique sont les valeurs de pixels des différents canaux de l'image. Pour une image de 100 par 100 pixels, à trois couleurs, cela représente un vecteur de 30 000 valeurs. Si on veut utiliser un système automatique pour reconnaître des objets représentés par des valeurs de pixels, la quantité de calculs à effectuer, même pour le plus simple des classificateurs, est considérable.

Le nombre de dimensions d'un problème correspond au nombre de caractéristiques associées à chaque objet ou échantillon à classifier. Il est dans notre intérêt de réduire le nombre de dimensions d'un problème sans perdre le pouvoir de classification - l'information pertinente - contenu dans les données. Le traitement des caractéristiques, transformation ou élimination,

peut ainsi réduire les coûts de mesures (quand une caractéristique est éliminée, rien ne sert de la mesurer par la suite). Se débarrasser d'information redondante ou inutile, ou mettre en évidence de l'information utile à la classification peut améliorer les performances d'un classificateur (Fayyad, Piatetsky-Shapiro et Smyth, 1996b).

La sélection de caractéristiques est une méthode de réduction de la dimensionnalité d'un problème. Plusieurs algorithmes ont été développés pour procéder à la sélection automatique. Ces algorithmes tentent d'identifier les caractéristiques les plus utiles et celles qui le sont moins. Les algorithmes de sélection procèdent à une recherche plus ou moins exhaustive des choix de caractéristiques possibles afin d'obtenir un groupe de caractéristiques qui permet de classifier les données avec une performance satisfaisante. La sélection de caractéristiques est couramment utilisée en vision pour réduire la quantité de donnée lorsque les valeurs de pixel sont utilisées comme caractéristiques. La sélection permet alors d'identifier un ensemble de pixels performant pour la reconnaissance : les pixels non choisis sont alors ignorés. Pour des problèmes de classification comportant une taille d'image variable, donc un nombre différent de pixels par image, la sélection sur les pixels n'est pas considérée.

L'extraction de caractéristiques, qui consiste à transformer les données, permet de réduire la quantité de ressources requises pour décrire un ensemble de données avec précision. Si les caractéristiques extraites sont bien choisies, le pouvoir de classification de l'algorithme en amont peut augmenter. L'extraction de caractéristiques permet en outre de simplifier la tâche du classificateur en apportant des informations déterminantes en rapport à la classe de l'objet observé. Une multitude de méthodes d'extraction de caractéristiques sont décrites dans la littérature; nous vous en présentons quelques-unes subséquemment.

1. Une première méthode d'extraction de caractéristiques consiste à utiliser des descripteurs fournis par la littérature et les experts des domaines reliés au problème (Weiss et Kulikowski, 1991) – méthode couramment utilisée en vision par ordinateur. Généralement, d'excellents résultats sont obtenus par cette méthode. En théorie, il est possible de réunir un grand nombre de descripteurs de forme, de couleur et de texture

utilisés en vision, puis de les utiliser tous pour être en mesure d'obtenir de bonnes performances de classification pour des problèmes de vision variés comme la reconnaissance de l'écriture, de visages ou de plantes. Ces caractéristiques sont des interprétations numériques de concepts provenant de la perception visuelle humaine. Une fois les descripteurs les plus utiles et reconnus répertoriés, nous n'aurons plus à choisir les descripteurs spécifiques à un contexte particulier lorsque nous aborderons un nouveau problème.

2. Plusieurs autres méthodes plus générales sont disponibles telles l'Analyse en Composantes Principales (ACP) (Jolliffe, 2002), l'analyse de caractéristiques par frontières de décision (de l'anglais « Decision Boundary Feature Analysis ») (Lee et Landgrebe, 1993), et les réseaux de neurones sans récurrence (de l'anglais Feed-Forward Neural Network) (Setiono et Liu, 1998). Ces méthodes numériques permettent d'extraire des caractéristiques à partir de données brutes sans consulter un expert. Encore une fois, pour des images de taille variable, il est difficile d'appliquer ces méthodes d'extraction sur les données brutes. Par contre, il est possible d'utiliser ces méthodes pour transformer les caractéristiques correspondant aux descripteurs fournis par la littérature (*Voir 1*), afin de réduire encore plus la dimensionnalité du problème. Cette option est discutée à la section 2.5.1.

3. Finalement, il existe aussi des méthodes automatiques pour construire des caractéristiques (Bourgouin, 2007; Matheus et Rendell, 1989). Les constructions sont des combinaisons linéaires ou non linéaires des données brutes. La construction peut être réalisée soit avant le processus de classification, soit en parallèle à celui-ci, afin de déterminer itérativement avec le classificateur si les caractéristiques construites aident vraiment à résoudre le problème étudié. Afin de concentrer nos efforts sur l'extraction de caractéristiques prédéfinies, nous avons décidé de ne pas approfondir l'avenue de la construction de caractéristiques.

2.5 Extraction des caractéristiques

On remarque dans la littérature de vision par ordinateur trois catégories importantes d'information à tirer d'une image préalablement segmentée. Il s'agit d'informations morphologiques, d'informations sur la couleur et d'informations sur la texture (Davies, 2004).

- Les informations **morphologiques** concernent la forme de l'objet, sa disposition et ses cavités. Seuls les contours de l'objet sont nécessaires pour l'extraction de ces caractéristiques;

- Les informations sur la **couleur** concernent certaines valeurs statistiques de l'ensemble des valeurs de pixels observées dans l'image segmentée, ou dans une section de celle-ci.

- L'information sur la **texture** concerne les motifs, les répétitions, la densité et la fréquence des valeurs de pixels de l'objet pour une certaine zone ou pour certaines positions relatives. Il s'agit donc d'information combinée, obtenue de la valeur de pixel et de son positionnement à la fois.

Cette section présente les caractéristiques les plus simples, couramment utilisées pour la reconnaissance de forme dans des images.

Morphologie

Les caractéristiques de la forme d'un objet sont basées sur les limites de l'objet observées dans un espace. L'information nécessaire pour le calcul de toutes les caractéristiques morphologiques d'un objet est donc la liste de tous les pixels qui composent le contour de l'objet. Cette information nous permet de déduire quels sont les pixels de l'image qui font partie de l'objet.

Les caractéristiques morphologiques les plus connues sont l'aire, le périmètre, la hauteur et la largeur. Les procédures pour le calcul de caractéristiques un peu plus complexes comme le centroïde, les axes principaux, l'orientation des axes, l'excentricité, le nombre d'Euler, le diamètre équivalent et l'étendue sont expliquées dans la page d'aide de la fonction « regionprops » de Matlab 7.0.1 (Eaton, 2004).

Couleur

Les espaces de représentations des couleurs Rouge Vert Bleu (RVB), Teinte Saturation Lumière (HSL), puis par niveaux de gris (un seul canal) sont couramment utilisés en vision par ordinateur. Bien que ces systèmes diffèrent quant à leur interprétation, le calcul des caractéristiques de couleur est simple dans chaque cas. Dans le cas du modèle RVB, l'extraction de caractéristiques sur la couleur sera répétée pour chaque canal et / ou utilisée sur les valeurs moyennes obtenues de tous les canaux (ce qui correspond aux niveaux de gris).

La valeur moyenne, la maximale et minimale de toutes les valeurs de pixels d'un canal sont des caractéristiques pertinentes. La valeur de pixel ayant le plus d'occurrence (maximum de l'histogramme) est une autre caractéristique de la couleur de l'objet. Finalement, la somme des valeurs, suite à une opération comme l'application d'un filtre sobel (Sobel et Feldman, 1973), ou le calcul de l'entropie (Shannon, 1948) sur un canal, peut être considérée comme une caractéristique de la couleur.

Texture

Il existe plusieurs techniques pour obtenir des informations sur la texture d'un objet. En premier lieu, on utilise l'histogramme des valeurs de pixels de l'objet afin de calculer des caractéristiques statistiques. Un histogramme présente le nombre d'occurrences d'une valeur de pixel parmi tous les pixels qui composent l'objet (par exemple, pour une valeur de 0 jusqu'à 255). Les caractéristiques intéressantes sont la moyenne de l'histogramme, son écart type, puis les moments de degrés supérieurs. Le moment de second degré décrit par Gonzalez, Woods et Eddins (2004, p. 667), qui correspond à la variance, est un indicateur du

contraste de niveaux de gris, utile pour décrire les changements de luminosité brusque sur l'objet. Les moments du troisième et du quatrième degré, nommés asymétrie et coefficient d'aplatissement décrivent chacun un aspect de la forme de l'histogramme des valeurs de pixels. L'uniformité et l'entropie du niveau de gris sont deux autres caractéristiques obtenues de l'histogramme qui décrivent la répartition régulière ou aléatoire des pixels dans l'objet. Ces dernières sont décrites par Gonzalez, Woods et Eddins (2004, p. 669).

Une deuxième technique pour le calcul de caractéristiques sur la texture est la co-occurrence de pixels. Elle consiste à dénombrer l'occurrence simultanée de deux pixels dans un voisinage déterminé qui possèdent des valeurs de pixels spécifiques. Le résultat de l'application d'un opérateur de co-occurrence est une seconde matrice (ou image). Cette matrice précise, à la case j de la ligne i, le nombre d'occurrence d'un pixel contenant la valeur i pendant que son voisin (selon le décalage précisé par l'opérateur) contient la valeur j. À partir de ce résultat, on peut calculer des caractéristiques statistiques comme le contraste, la corrélation, l'énergie et l'homogénéité (*Voir* fonctions « graycomatrix » et « graycoprops » de Matlab).

Enfin, on peut aussi utiliser des analyses plus spécialisés comme la forme binaire locale, tirée de l'anglais « Local Binary Pattern » (Ojala, Pietikäinen et Harwood, 1996) et l'analyse en fréquence, qui utilise la transformée de Fourier discrète.

2.5.1 Transformation des données par analyse en composantes principales

Il est possible d'écourter la phase d'apprentissage des classificateurs en réduisant la dimensionnalité du problème. La combinaison linéaire de caractéristiques par ACP permet de réduire le nombre de caractéristiques utilisées pour décrire les échantillons tout en conservant une proportion choisie de la variance des caractéristiques originales.

Les idées derrière l'ACP ont été introduites indépendamment par Pearson (Pearson, 1901) et, plus tard, par Hotelling (Hotelling, 1933). L'ACP calcule la matrice de covariance à partir

des caractéristiques de la base de données et trouve les valeurs propres et vecteurs propres. Chacun des vecteurs propres devient une transformée linéaire qui agit sur les caractéristiques originales pour créer une nouvelle caractéristique qui est appelée composante principale (Jolliffe, 2002). La valeur propre associée à chaque vecteur propre correspond à la variance des données sur chaque composante principale. Nous assumons ici que la qualité descriptive d'une composante principale est mesurée par la grandeur de sa variance. Les meilleures caractéristiques sont donc celles avec la variance la plus élevée. Ainsi, l'ACP utilise les valeurs propres pour sélectionner les meilleures caractéristiques produites. Le total de la variance conservée (la somme des valeurs propres utilisées) est un indicateur de la somme de l'information contenue dans la base de données originale qui sera conservée. Les vecteurs propres choisis sont utilisés pour transformer les caractéristiques de la base de données originale en de nouvelles caractéristiques, formant une nouvelle base de données.

$$Y = XH \tag{2.1}$$

X est la matrice d'échantillons de taille n par d, Y est la matrice d'échantillons après transformation par l'ACP, de taille n par m, et H est la matrice d par m de transformation linéaire, dont les colonnes sont les vecteurs propres. Puisque l'ACP utilise les vecteurs associés aux plus grandes variances, elle réalise une approximation des données dans un sous-espace linaire en utilisant le critère de l'erreur quadratique moyenne (« mean squared error » en anglais).

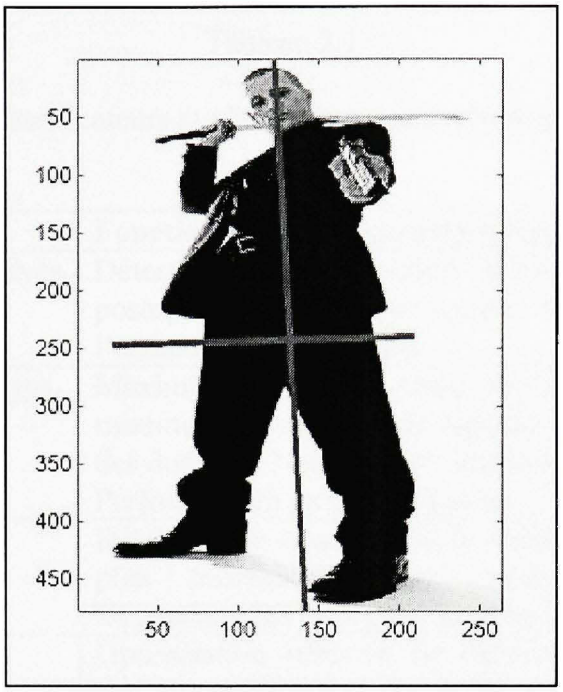


Figure 2.2 ***Les deux axes d'une ACP sur la photo d'un homme debout.***
(Tiré de (Wikipédia, 2006c))

2.6 Classificateurs

Dans cette section, nous présentons six classificateurs statistiques (non pas syntaxiques) de type supervisé. Nous avons sélectionné ces classificateurs parmi la grande variété d'algorithmes décrits dans la littérature puisqu'ils sont couramment utilisés comme classificateurs de référence dans des études comparatives (van der Walt et Barnard, 2006) (Stathakis et Vasilakos, 2006).

Voici un tableau qui présente les classificateurs et commente leurs principales caractéristiques :

Tableau 2.1

Classificateurs étudiés et leurs caractéristiques

Classificateur	**Fonctionnement et caractéristiques**
Classificateur naïf de Bayes	Détermine la classe selon la plus grande probabilité à posteriori. Classificateur simple. Sensible aux erreurs pour l'estimation de la densité.
Séparateur à vastes marges	Maximise la marge entre les classes en utilisant un minimum de vecteurs de support. Dépendant de l'échelle des données. Non linéaire. Insensible au sur-apprentissage. Performe bien en généralisation.
K plus proches voisins	Détermine la classe selon la classe majoritaire chez les K plus proches voisins. Asymptotiquement optimal. Dépendant de l'échelle. Phase de classification lente.
Perceptron multicouche	Optimisation itérative de l'erreur quadratique moyenne par la modification de poids de plusieurs couches de neurones artificiels. Sensible aux paramètres d'entraînement. Entraînement lent. Sensible au sur-apprentissage.
Arbre de décision C4.5	Classifie à l'aide d'un ensemble de valeurs de seuil pour une séquence de caractéristiques. Procédure d'entraînement itératif. Sensible au sur-apprentissage. Phase de classification très rapide.
Programmation génétique	Procède à l'évolution artificielle de programmes pour la classification. Beaucoup de paramètres à ajuster. Phase d'apprentissage longue et très exigeante en ressources computationnelles. Flexibilité de recherche dans l'espace des solutions.

Les principes de fonctionnement de ces classificateurs sont décrits sommairement dans les sections suivantes.

2.6.1 Classificateur naïf de Bayes

Le classificateur naïf de Bayes est un classificateur simple utilisant des probabilités obtenues par le théorème de Bayes (Bayes, 1763; Laplace, 1820) avec de fortes hypothèses d'indépendance des données. Ces hypothèses lui valurent son nom : classificateur naïf.

Ce type de classificateur a fait ses preuves pour des problèmes d'application pratique avec apprentissage supervisé. Malgré ses hypothèses, le classificateur naïf performe généralement bien avec peu de données d'apprentissage. Puisque les données sont considérées indépendantes, seule la variance de chaque caractéristique doit être calculée (et non pas la matrice de covariance en entier).

Le model probabiliste utilisé par ce classificateur provient du théorème de Bayes. Le classificateur qui s'ensuit, étant donné l'indépendance assumée des variables, procède de la façon suivante :

$$classe(f_1,...,f_n) = plusgrand_c\left(p(C=c)\prod_{i=1}^{n} p(F_i = f_i \mid C=c) \right) \tag{2.2}$$

La fonction *plusgrand* ne fait que sélectionner la plus grande valeur parmi les probabilités que l'objet appartienne à chacune des classes. On obtient cette probabilité par la multiplication de la probabilité d'obtenir cette classe (parmi les échantillons d'apprentissage) avec la probabilité d'obtenir chaque caractéristique pour cette classe, selon la valeur moyenne et la variance préalablement calculée. La classe qui obtient la plus grande probabilité en fonction des caractéristiques de l'objet présenté est choisie en tant que classe prédite pour l'objet. Pour plus de précision, il est possible de consulter plusieurs ouvrages et articles concis sur le classificateur naïf de Bayes (Buntine, 1996; John et Langley, 1995).

2.6.2 Séparateur à vaste marge

Une machine à vecteurs de support ou Séparateur à Vaste Marge (SVM) est un classificateur qui fait partie des méthodes d'apprentissage supervisées. Son principe général est de séparer deux ensembles de points par un hyperplan. Le classificateur proposé par Vapnik (Vapnik, 1998) obtient, dans certains cas, des résultats supérieurs à ceux des réseaux de neurones ou des modèles statistiques comme le classificateur naïf de Bayes.

En résumé, le SVM utilise des fonctions de noyau (de l'anglais « kernel functions ») qui, dans un espace augmenté, permettent une séparation optimale des points en différentes

catégories. Les données d'apprentissage sont utilisées pour découvrir l'hyperplan qui séparera au mieux les points.

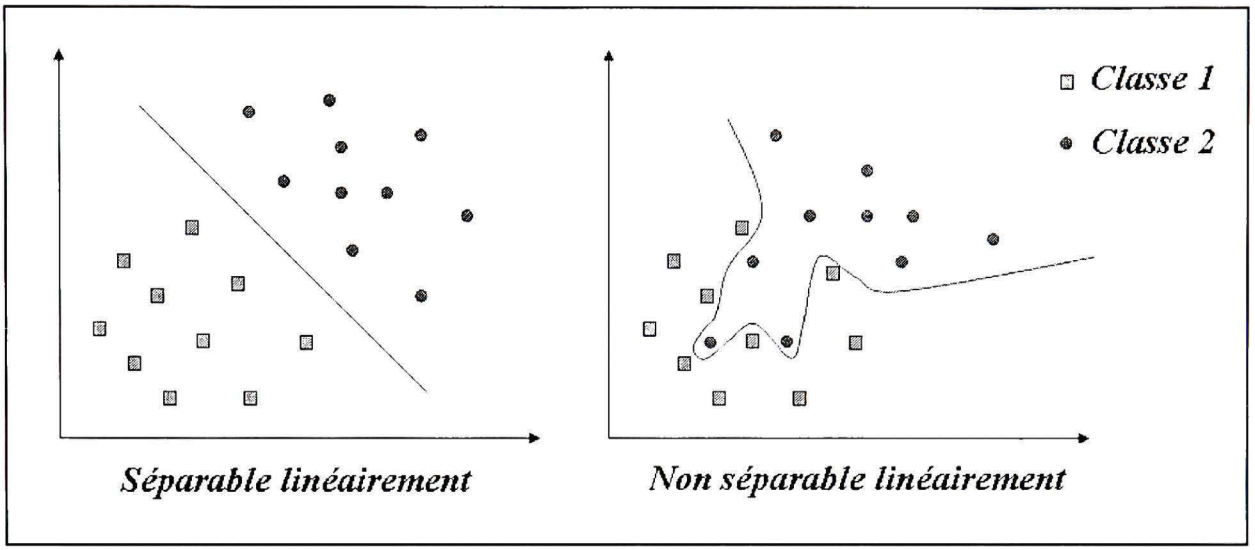


Figure 2.3 ***Exemple de deux classes linéairement séparables et non séparables.***
(Tiré de (Wikipédia, 2006a) [notre traduction])

L'idée du SVM est d'utiliser sa fonction de noyau pour reconsidérer le même problème dans un espace de dimension plus élevée. Ce nouvel espace est caractérisé par la possibilité d'y trouver un séparateur linéaire qui permet de classer les points dans les deux groupes appropriés. Le séparateur linéaire peut ensuite être projeté dans l'espace d'origine (où il devient habituellement non linéaire). La figure 2.3 illustre des distributions séparable et non séparable pour deux classes (carrés et cercles), dans un espace à deux dimensions.

Le critère d'optimisation est la largeur de la marge entre les classes, c'est-à-dire l'espace vide de chaque côté des frontières de décision. La largeur de marge est caractérisée par la distance jusqu'aux échantillons d'entraînement le plus près. Ces échantillons, appelés vecteurs de supports, définissent la fonction discriminante qui permet la classification. Le nombre de vecteurs de support est minimisé en maximisant la marge.

La fonction de décision pour un problème à deux classes obtenue à partir du classificateur à

vaste marge est formalisée comme suit, en utilisant la fonction de noyau $K(x_i,x)$ d'un nouvel échantillon x et d'un échantillon d'entraînement x_i :

$$D(x) = \sum_{\forall x_i \in S} \alpha_i \lambda_i K(x_i, x) + \alpha_0 \tag{2.3}$$

Où S est l'ensemble de vecteurs de support (sous-ensemble obtenu de la base d'entraînement), et $\lambda_i = \pm 1$ (selon la classe de l'objet x_i). Les paramètres $\alpha_i >= 0$ sont optimisés pendant l'apprentissage à l'aide de l'équation :

$$\min_\alpha (\alpha^T \Lambda K \Lambda \alpha + C \sum_j \varepsilon_j) \tag{2.4}$$

Avec les contraintes $\lambda_j D(x_j) \geq 1 - \varepsilon_j, \forall x_j$ dans l'ensemble d'entraînement. Λ est une matrice diagonale contenant les classes λ_j et la matrice K contient les valeurs de la fonction de noyau $K(x_i,x)$ pour toutes les paires d'échantillons d'entraînement. L'ensemble de variables flexibles e_j permet le chevauchement de classes ajusté par la pénalité $C > 0$. Pendant l'optimisation, tous les α_i se rapprochent de 0, sauf les vecteurs de support. Finalement, seuls ces vecteurs sont utilisés.

Le SVM nécessite la sélection d'une fonction de noyau K appropriée. La forme la plus simple de noyau est le produit scalaire entre l'entrée et un échantillon de l'ensemble de support. Le résultat est un classificateur linéaire (*Voir* équation 2.5). Des noyaux non linéaires, polynomiaux de degré p sont obtenus par la forme présentée à l'équation 2.6. D'autres noyaux comme la gaussienne de bases radiales (*Voir* équation 2.7) à écart type σ peuvent aussi être utilisées.

$$k(x_i x) = x_i \cdot x \tag{2.5}$$

$$k(x_i x) = (x_i \cdot x + 1)^p \tag{2.6}$$

$$k(x_i x) = \exp\left(- \frac{\|x_i - x\|^2}{2\sigma^2} \right) \tag{2.7}$$

On peut analyser le SVM selon l'optique de l'appariement de modèles, technique connue sous le nom de « template matching » en anglais. Les vecteurs de supports font office de

modèle avec la différence qu'ils caractérisent les classes selon la frontière de décision – donc l'échantillon limite de la classe – par opposition au modèle habituel de l'appariement qui correspond à l'échantillon moyen ou attendu de la classe. De plus, la frontière de décision est définie selon la combinaison, possiblement non linéaire, des distances avec les vecteurs de supports.

La littérature sur les formes d'implémentation du SVM est abondante (Schölkopf, Burges et Smola, 1998; Shawe-Taylor et Cristianini, 2004).

2.6.3 K plus proches voisins

L'algorithme de classification des K Plus Proches Voisins (KPPV) est parmi les plus simples de tous les algorithmes de l'AM. C'est une méthode d'apprentissage supervisé. Un objet est classifié selon un vote à majorité par ses voisins; l'objet obtient la classe qui est la plus commune chez ses K plus proches voisins dans l'espace des caractéristiques. K doit donc être un entier positif, généralement petit. On choisira souvent un K impair pour éviter l'égalité dans le vote. La distance utilisée pour le calcul de la proximité des voisins est le plus souvent la distance euclidienne.

Les exemples d'apprentissage sont des vecteurs dans un espace multidimensionnel. La phase d'apprentissage consiste simplement à conserver ces données dans un format qui permettra le calcul efficace des distances et la recherche des voisins.

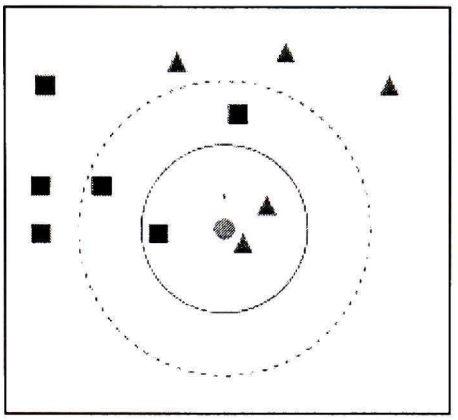


Figure 2.4 ***Exemple de problème dans un espace à deux dimensions.***
(Tiré de (Wikipédia, 2007b))

La figure 2.4 illustre la répartition de deux classes, les carrés et les triangles, dans un espace bidimensionnel. Le cercle représente un nouvel échantillon à classifier. La méthode du KPPV nous permet d'attribuer une classe à cet échantillon, en observant la classe de ses voisins. Par exemple, avec $K = 3$, les trois voisins sont deux triangles et un carré (dans la région circulaire pleine, figure 2.4), ce qui identifie le nouvel échantillon comme un triangle. Par contre, dans le cas de $K = 5$, les voisins sont deux triangles et trois carrés (région circulaire pointillée de la figure 2.4). Dans ce cas, le nouvel échantillon serait classifié comme un carré.

Le paramètre K a donc un effet sur la performance de ce classificateur. Le meilleur choix dépend des données. Généralement, un K plus grand pourra réduire l'effet du bruit, mais les frontières entre les classes seront moins distinctes. Une bonne valeur de K peut être sélectionnée par des techniques heuristiques comme la validation croisée. La performance du KPPV peut être sérieusement diminuée par la présence de données bruitées ou qui ne fournissent pas d'information pertinente. Pour l'utilisation du K plus proches voisins, il est primordial de fournir une échelle appropriée pour les données, puisque des valeurs trop grandes ou trop petites, en comparaison aux autres données, auront un effet soit exagéré ou soit négligeable sur le calcul de distance.

Plusieurs algorithmes d'optimisation ont été présentés afin de diminuer la charge de calcul dédiée à la recherche des voisins les plus proches d'un nouvel échantillon. Nous ne nous

attarderons pas ici aux détails de ces optimisations. Pour plus d'information sur le KPPV et ses variantes, consultez (Shakhnarovich, Darrell et Indyk, 2006) ou encore (Dasarathy, 1991).

2.6.4 Perceptron multicouche

Les Réseaux de Neurones Artificiels (RNA) sont un ensemble d'outils informatiques inspirés du fonctionnement du cerveau animal (Rosenblatt, 1962). Les RNA utilisent les liens entre les neurones pour apprendre à partir d'information observée. Cette méthode d'apprentissage en parallèle est robuste et résistante à l'introduction d'information erronée. Les RNA nécessitent habituellement une bonne quantité de données d'apprentissage, ce qui peut être coûteux en temps machine. Mais, une fois entraînés, ces réseaux fournissent des prédictions suffisamment rapidement pour fonctionner en temps réel sur une ligne de production.

Un RNA standard contient trois types de couches de neurones : une couche d'entrée, une ou plusieurs couches cachées et une couche de sortie. Les RNA les plus employés pour la classification utilisent le principe du flux vers l'avant (appelé feed-forward en anglais). Plusieurs couches de neurones sont inter reliées avec la sortie du neurone sur une couche connectée à l'entrée d'un ou plusieurs neurones sur la couche suivante (le flux de calcul est donc toujours poussé dans le même sens). Le résultat est recueilli sur la couche de sortie du réseau. La figure 2.5 en illustre un exemple.

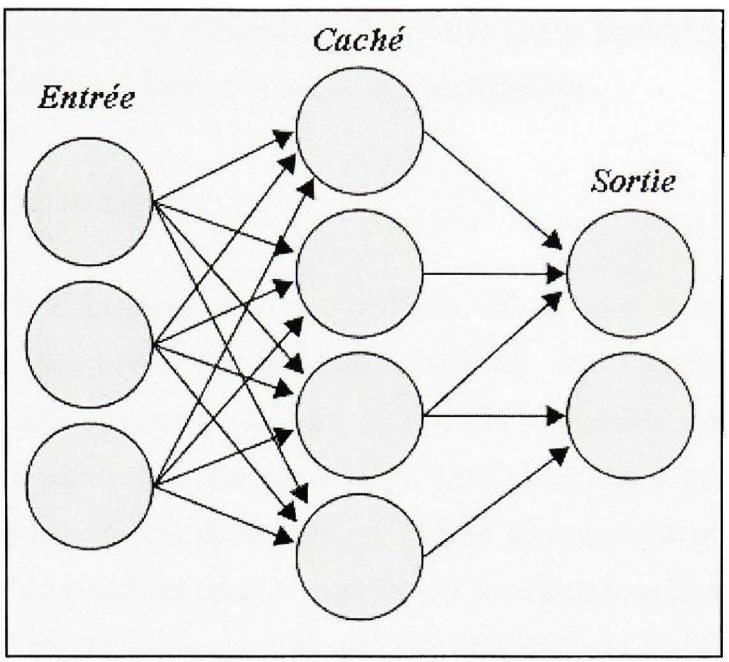


Figure 2.5 ***Structure typique de RNA.***
(Tiré de (Wikipédia, 2006b)[notre traduction])

Le Perceptron Multicouche (PM) est, comme son nom l'indique, un RNA qui peut contenir plusieurs couches cachées (en pratique, souvent une seule suffit). En phase d'apprentissage, l'information numérique est injectée dans la couche d'entrée et transformée à travers le réseau par les poids et fonctions (habituellement des sigmoïdes) attribués à chaque neurone. Le résultat provient des relations entre les neurones d'entrée, qui représentent les caractéristiques de l'échantillon à classifier, et la sortie, qui en représente la classe (Brierley et Batty, 1999). L'entraînement du PM consiste à trouver les poids des neurones qui permettent au mieux de prédire la classe des échantillons d'entraînement à l'aide de leurs caractéristiques. La technique la plus utilisée pour l'ajustement des poids est la rétropropagation du gradient (Jain, Mao et Mohiuddin, 1996). Cette technique consiste à corriger les erreurs selon l'importance des éléments qui ont justement participé à la réalisation de ces erreurs. Une fois l'erreur sur la couche de sortie calculée, les poids des neurones qui lui sont reliées sur la couche qui la précède immédiatement sont corrigés pour diminuer l'erreur, et ainsi de suite jusqu'à la couche d'entrée du réseau. La correction des poids par rétropropagation du gradient est effectuée soit un certain nombre de fois (le nombre

d'itérations est appelé nombre d'époques), ou jusqu'à ce que la performance de classification pour la base de données de validation commence à se détériorer.

2.6.5 Arbre de décision C4.5

La construction d'arbre de décision est une méthode utilisée pour faire de la classification (Quinlan, 1986). Les arbres de décision classifient les échantillons en séparant successivement les informations de la base de données d'apprentissage en sous parties à l'aide de critères de séparation. L'arbre de décision fonctionne donc à la manière d'une clé de détermination (aussi appelée clé dichotomique, utilisée en botanique pour l'identification), présentant une suite de questions selon le modèle « la caractéristique X est de type Y ? ». La réponse indique soit une nouvelle question, soit une prédiction de la classe de l'échantillon. Certains arbres auront un maximum de deux branches par nœud, alors que d'autres pourront en avoir plus de deux.

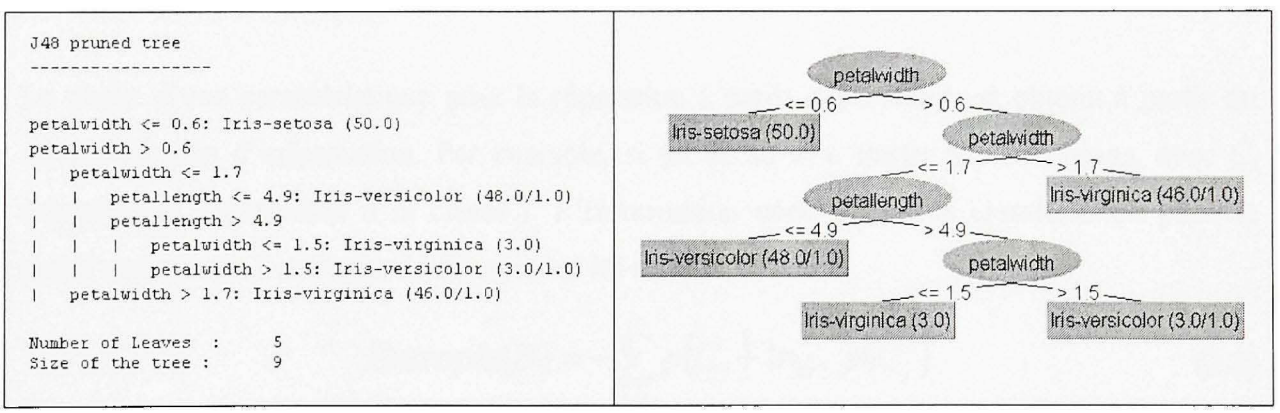


Figure 2.6 ***Texte d'arbre de décision et représentation graphique.***

Source : Image et texte obtenus à partir d'un arbre C4.5 produit par Weka

La phase d'apprentissage des arbres de décision est communément divisée en trois sous étapes. Il s'agit de la croissance de l'arbre (séparation en branches), de l'arrêt et de l'élagage. Ces sous phases varient selon l'algorithme de construction de l'arbre de décision. Le critère

de séparation est la mesure la plus importante pour catégoriser les différents algorithmes de construction d'arbre.

Nous traitons ici plus particulièrement de l'algorithme C4.5, qui est un algorithme performant pour générer un arbre de décision (Quinlan, 1993). Il s'agit d'une amélioration de l'algorithme ID3 (Quinlan, 1986).

ID3 (de l'anglais Iteractive Dichotimizer 3) est utilisé pour la construction d'arbres de décision qui classifient des échantillons à caractéristiques discrètes (non continues). ID3 construit l'arbre de décision avec les données d'apprentissage en utilisant les concepts d'entropie de l'information et de gain d'information. L'entropie de l'information ou entropie de Shannon est une mesure de la quantité d'information délivrée par une source. La définition de l'entropie d'une source, selon Shannon, indique que plus la source est redondante, moins elle contient d'information. Inversement, si la source donne des valeurs toujours différentes, l'entropie sera maximale (Shannon, 1948).

Le choix d'une caractéristique pour la séparation à partir d'un nœud est obtenu à partir du critère de gain d'information. Par exemple, si un nœud doit traiter S échantillons, avec C_j échantillons appartenant à la classe j. L'information nécessaire à la classification pour le nœud courant est l'entropie. L'entropie est ici calculée.

$$Entropie(S) = -\sum_{j=1}^{c} p(C_j) \cdot \log_2 p(C_j) \tag{2.8}$$

Cette valeur correspond à la quantité moyenne d'information nécessaire pour identifier la classe d'un échantillon. Si la caractéristique A est choisie pour séparer les échantillons, un nombre n de sous-ensembles sera obtenu. Si S_i est le sous ensemble i parmi les n, l'entropie de chaque S_i peut être calculée. L'information requise après avoir choisi la caractéristique A pour la séparation est la moyenne pondérée des sous-ensembles d'information :

$$Entropie_A(S) = \sum_{j=1}^{n} p(C_j) \cdot Entropie(S_i) \tag{2.9}$$

Le gain d'information obtenu par la séparation avec A est donc la différence entre l'entropie

de A et l'entropie obtenue de l'ensemble de base.

$$Gain(A) = Entropie(S) - Entropie_A(S) \tag{2.10}$$

Plus l'entropie de A est petite, plus le gain de A sera grand et plus la caractéristique est appropriée pour différentier les classes de S. Lorsque l'attribut est choisi, tous les échantillons sont divisés parmi les différentes valeurs de A et constituent de nouveaux ensembles de départ S. Si tous les échantillons sont de la même classe, le nouveau nœud devient un nœud final étiqueté avec la classe. Ce processus continue jusqu'à ce que tous les échantillons soient identifiés avec une classe.

C4.5 profite du fait que chaque caractéristique permet de prendre une décision qui sépare les données en plus petits groupes. C4.5 utilise le gain d'information normalisé résultant d'un choix d'une caractéristique pour séparer les données. Encore une fois, la caractéristique avec la plus grande valeur de gain d'information normalisée est celle utilisée pour séparer les échantillons au nœud en cours. L'algorithme procède de la même façon pour les sous-ensembles produits.

C4.5 apporte aussi des améliorations intéressantes pour notre application : il peut traiter les données continues en utilisant une valeur de seuil pour la caractéristique choisie (le seuil est déterminé en fonction des valeurs possibles après le tri des données selon l'attribut utilisé pour la séparation) et il élague l'arbre après la création en remplaçant les branches jugées peu utiles par des nœuds finaux.

2.6.6 Programmation Génétique

La Programmation Génétique regroupe une variété d'algorithmes de recherche dont la particularité est de faire évoluer des programmes de taille variable selon une méthode inspirée de la sélection naturelle (Banzhaf et al., 1998). La PG est à distinguer des algorithmes génétiques (AG) qui, bien qu'utilisant aussi le principe de sélection naturelle, représentent une catégorie d'algorithmes distincte.

Selon les paramètres et la taille maximale des programmes allouée par l'usager, des solutions de formes extrêmement diverses peuvent être obtenues par un algorithme de PG. Présentée par Koza au début des années 1990 (Koza, 1990; , 1992), l'idée de faire évoluer des programmes informatiques pour solutionner des problèmes complexes se répand et plusieurs créneaux se développent avec des résultats impressionnants (Koza, 2007).

La PG peut faire évoluer des programmes de prédiction pour la régression symbolique (prédiction en valeur continue) ou la classification. En résumé, elle procède à la création d'une population de programmes aléatoires, puis elle tente de les faire évoluer au moyen de la sélection des plus aptes. Ceux-ci sont reproduits pour obtenir des programmes qui présentent une bonne performance de prédiction. La création et l'évolution des programmes peuvent suivre des règles différentes selon les paramètres choisis par l'utilisateur.

Les programmes générés par un même algorithme de PG présentent une structure uniforme. Traditionnellement, la structure de données choisie est un arbre. D'autres implémentations utilisant des graphes (Kantschik et Banzhaf, 2002), puis du code machine (Nordin, Banzhaf et Francone, 1999) ont été présentées. Pour simplifier la présentation des principes de l'évolution de la PG, nous illustrerons nos propos avec des structures en arbre. L'arbre de la PG peut ressembler à un arbre de décision (comme l'arbre de décision C4.5, par exemple). Les nœuds intermédiaires de l'arbre sont des fonctions ou opérateurs et les nœuds finaux (que nous appellerons feuilles) sont des constantes ou des caractéristiques qui proviennent des échantillons.

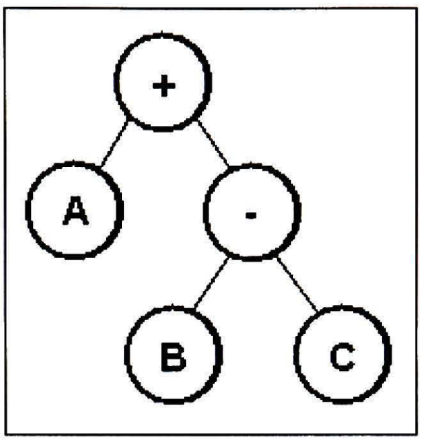


Figure 2.7 ***Programme de PG avec structure en arbre.***
(Tiré de (Levasseur, 2005))

Source : Tiré d'un rapport produit pour le cours Reconnaissance de formes et Inspection, à l'École de technologie supérieure.

La population initiale est constituée de programmes créés aléatoirement. Cette population évolue au rythme d'opérations génétiques (opérations de reproduction) à chaque génération. Les programmes qui réussissent mieux dans l'exécution de la tâche voulue ont normalement plus de chances d'être sélectionnés pour la reproduction. La PG se termine lorsqu'un critère d'arrêt a été atteint (par exemple un nombre de générations, un taux de prédiction ou un temps de calcul).

Il existe plusieurs méthodes pour la gestion des programmes de PG : le calcul de leur aptitude (appelée valeur d'adéquation), la sélection des individus les plus aptes, l'évolution générale de l'algorithme, etc. Ces méthodes sont reprises par les différents chercheurs de la PG. Les procédures les plus uniformes parmi les algorithmes sont celles réalisées par les opérateurs génétiques. On appelle croisement (« crossover » en anglais) l'opérateur génétique le plus communément utilisé. Il consiste à échanger une section de code d'un premier parent avec une section de code d'un deuxième parent. Cette opération imite la reproduction sexuée chez les vivants. Le deuxième opérateur le plus utilisé est la mutation. Il consiste à changer une section de code du programme par du nouveau code généré aléatoirement. Afin de conserver

ou de multiplier les programmes performants, il est possible d'utiliser la simple reproduction (recopier entièrement un programme dans la nouvelle génération). La reproduction de programmes et le scénario évolutif permettent l'apparition de l'élitisme (*Voir* section 3.6.9). L'élitisme permet la conservation des constituants de programme les plus performants au cours de l'évolution.

Afin de procéder à la classification avec la PG, plusieurs implémentations sont possibles. Premièrement, un programme de classification multi classes peut être élaboré. Une autre approche consiste à créer plusieurs programmes, un pour chaque classe par exemple, et de les combiner par un système de vote pour identifier la classe (méthode « un contre tous »). On pourrait aussi utiliser un nombre encore plus grand de programmes, un pour chaque paire de classes, (méthode « un contre un ») et voter. Finalement, on peut aussi combiner plusieurs classificateurs pour un même problème, pour une même classe (Smart et Zhang, 2005), afin d'augmenter les probabilités d'obtenir une solution qui présente une bonne performance de généralisation (Folino, Pizzuti et Spezzano, 2006).

Puisque nous réaliserons nous-mêmes une implémentation de PG dans le cadre de ce travail, les détails des options disponibles seront présentés au chapitre 3.

2.7 Méta algorithmes pour l'apprentissage

Un méta algorithme est un algorithme qui manipule d'autres algorithmes complets. Il peut être utilisé pour ajuster les paramètres de ces sous algorithmes ou pour les combiner afin d'accomplir une tâche complexe; dans ce cas chaque sous algorithme pourra se spécialiser pour une portion du problème.

Les méta algorithmes peuvent être utilisés pour la classification. Ils servent à développer des algorithmes de classification (souvent de façon itérative) que l'on pourra combiner afin d'augmenter la performance globale de classification. Nous présentons ici deux méta algorithmes populaires pour la prédiction statistique : le bagging et le boosting.

2.7.1 Bagging

« Bagging » est un diminutif du nom complet de la méthode, soit en anglais « **B**ootstrap **Agg**regat**ing** ». Le bootstrap est une méthode statistique pour améliorer les propriétés et la fiabilité des estimateurs et des tests statistiques, en particulier pour de petites bases de données.

La méthode du bagging a comme principe de faire la moyenne des prédictions de plusieurs modèles indépendants permettant ainsi de réduire la variance et donc de réduire l'erreur de prédiction (Breiman, 1996). Traditionnellement, le bagging a été utilisé pour combiner des arbres de décisions. Il est pourtant possible d'utiliser la méthode pour n'importe quel type d'algorithme.

À partir d'une base de données d'exemples, un nouvel ensemble d'apprentissage est créé par échantillonnage uniforme avec remplacement (cela permet que certains échantillons soient répétés). Un certain nombre de nouveaux ensembles sont ainsi générés. Un modèle est ensuite obtenu pour chaque ensemble d'apprentissage. Dans le cas d'un problème où une régression est utilisée, on procède à une moyenne de tous les modèles pour avoir un résultat final. Pour des classificateurs, on utilise le vote pour déterminer la classe d'un nouvel échantillon.

2.7.2 Boosting

La méthode du *boosting* est similaire à celle du *bagging* pour la création d'un groupe de modèles qui sont ensuite combinés par une moyenne pondérée des estimations. Elle est différente du bagging au niveau de la construction des modèles : chaque modèle est une version adaptative du précédent qui donne plus de poids, lors de l'estimation suivante, aux échantillons qui ont connu une erreur de prédiction. En théorie, le boosting fonctionne pour des modèles d'apprentissage faibles. Il a été démontré qu'il peut aussi augmenter les performances de certains algorithmes qui ne sont pas considérés faibles comme l'arbre de

décision C4.5 (Quinlan, 1996) ou la PG (Paris, Robilliard et Fonlupt, 2001). Voici le pseudo code d'une technique de boosting bien connue nommée « AdaBoost » :

AdaBoost (Freund et Schapire, 1996)

Ensemble d'entraînement T = toutes les données d'entraînement disponibles
N = nombre total d'échantillons dans T

Initialiser le « poids » W de chaque échantillon i avec $W_i = \frac{1}{N}$

m = nombre de modèles voulus

Faire m fois
    Entraîner un algorithme avec T
    Calculer l'erreur du modèle (E_m) sur T
    Si E_m est plus grand que 0.5, arrêter le processus
    Calculer le facteur $a_m = 0.5 \cdot \ln\left(\frac{1 - E_m}{E_m}\right)$
    Ajuster le poids de chaque échantillon i mal classifié avec $W_i = W_i \cdot \exp(a_m)$
    Normaliser le poids de chaque échantillon avec $W_i = \frac{W_i}{\sum_i W_i}$
    T est recréé en utilisant N échantillons. W_i est la probabilité d'utiliser l'échantillon i pour le nouvel ensemble T.
Fin pour m

On peut classifier en utilisant la réponse la plus forte selon une somme pondérée des modèles : Sortie finale $= \sum_m (a_m \cdot sortie_m)$

2.8 Module intégré de reconnaissance de formes dans des images

Un système intégré de reconnaissance de formes dans des images pourrait grandement contribuer à la propagation de l'utilisation des techniques de l'IA et de l'exploration des données pour automatiser des tâches visuelles. Un outil semblable serait appréciable pour les industriels et chercheurs de domaines variés (automatisation, assurance qualité, dénombrement, évaluation statistiques, identification, etc.).

Idéalement, ce système pourrait assurer la reconnaissance d'objets ou l'évaluation d'une ou plusieurs sorties numériques en prenant simplement une base de données d'images en entrée. L'usager devrait fournir les classes ou les sorties attendues pour chaque échantillon fourni pour l'apprentissage.

À notre connaissance, peu de travail a été accompli pour développer un système complet de reconnaissance de formes dans des images. Le seul outil qui se rapproche des qualités que nous recherchons est PADO : « Parallel Algorithm Discovery and Orchestration », présenté dans la section suivante.

2.8.1 L'architecture PADO de Teller et Veloso

PADO est une architecture d'apprentissage qui implémente des techniques novatrices afin de progresser vers une reconnaissance d'objets naturels de façon générale. Une section du livre Symbolic Visual Learning (Teller et Veloso, 1997) présente des exemples de reconnaissance pour des images et les auteurs discutent aussi de tests effectués sur des objets sonores. L'architecture de PADO est particulièrement intéressante par son originalité.

Premièrement, l'article propose d'utiliser PADO pour la reconnaissance d'objet non segmenté dans des images en tons de gris. Les caractéristiques de l'image restent brutes : les entrées correspondent aux valeurs de tous les pixels de l'image (dans le cas de l'article, il s'agit d'images de 256 pixels de haut par 256 pixels de large). PADO utilise des limites temporelles jumelées à des techniques évolutives afin de cibler rapidement l'information intéressante obtenue des valeurs de l'image.

Ensuite, Teller et Veloso proposent une méthode computationnelle évolutive largement basée sur la Programmation Génétique. Les programmes que l'on doit faire évoluer sont des graphes additionnés de mémoire indexée utilisant des fonctions statistiques comme moyenne simple, maximum, minimum et écart type sur des régions rectangulaires de l'image (donc sur

des matrices de pixels). Les résultats de ces opérations peuvent être transformés par des opérateurs mathématiques comme la multiplication, l'addition, la sélection d'un maximum ou d'un minimum, puis emmagasinés dans la mémoire indexée, et éventuellement utilisés comme arguments pour une autre fonction. Le graphe peut s'exécuter en boucle grâce à des opérateurs logiques puis à un opérateur de fin de programme. On s'assure de la finalité de chaque programme en imposant un maximum de temps d'exécution. Dans le cas de l'atteinte de ce temps maximal, la valeur d'un registre en particulier est retournée comme sortie. Les programmes de PADO ont aussi accès à de plus petits programmes à la façon des Fonctions Définies Automatiquement (*Voir* section 3.6.3). Il s'agit de mini programmes personnels et de programmes stockés dans une librairie (disponibles pour tous les programmes principaux), eux aussi évolutifs, que les programmes principaux peuvent invoquer à leur guise.

Pour un problème de classification, PADO doit produire un certain nombre de programmes par classe. Ces programmes devront reconnaître les objets faisant partie de leur classe. PADO utilise le principe de la PG pour faire évoluer une population de programmes vers son objectif : obtenir le meilleur taux de reconnaissance pour la base de données d'exemples. La sortie de chaque programme est une valeur entre 0 et 255, interprétée comme une valeur de confiance. Cette valeur représente les probabilités d'appartenance de l'échantillon à la classe reconnue par le programme. La transformation des programmes par le principe de la sélection selon leur performance est exécutée par des opérateurs évolutifs appelés SMART. Les opérations génétiques sont effectuées par ces opérateurs, qui sélectionnent des sous graphes et les échangent.

L'évolution de tels programmes a des effets ingénieux. L'utilisation de boucles et de sorties de fonctions comme arguments à d'autres fonctions permettent au programme de localiser l'information pertinente à l'intérieur d'une image assez rapidement. De plus, la contrainte temporelle offre au programme la possibilité de raffiner sa sortie en examinant de façon plus approfondie des sections d'intérêt découvertes si le temps le permet.

Un classificateur final est composé à partir de « l'orchestration » des meilleurs programmes évolués par PADO (S programmes pour chaque classe). La sortie de chaque classificateur O_j est pondérée par une valeur V_j suite à une validation de ses performances pour un certain nombre d'exemples pour chaque classe. Dans une seconde étape, la valeur pour chacune des classes possibles est calculée selon la somme pondérée de la prédiction des meilleurs classificateurs :

$$System_i = \sum_j \left(O_{ij} \cdot \frac{V_{ij}}{V_{iTotal}} \right) \tag{2.11}$$

Le même processus est répété avec une pondération W_i par classe en particulier, la sortie du classificateur étant le maximum pour i des W_i * $System_i$ obtenus.

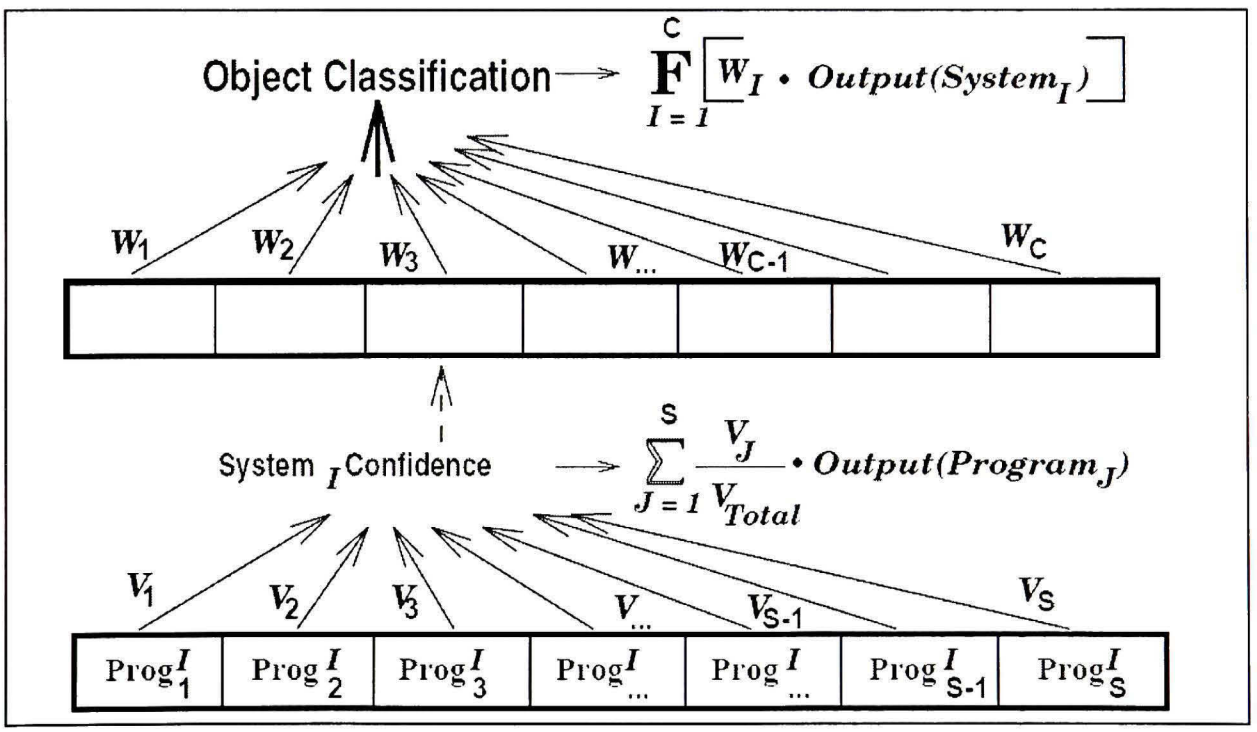


Figure 2.8 ***Orchestration des programmes.***

(Tiré de Teller et Velloso, 1996)

Source : Figure tirée de *PADO: A new learning architecture for object recognition*, de Teller et Veloso, à la page 20.

Après l'évolution des programmes, les pondérations W et V sont ajustées à l'aide de tests de classification sur quelques images.

L'article de Teller et Velloso présente des résultats pour deux petites bases de données d'images à sept classes. Il s'agit d'images d'objets pris sous différents angles avec la présence occasionnelle d'une main ou d'ombres avec un fond plus ou moins uniforme mais de couleur unie. Un ensemble de 14 images par classe est utilisé en phase d'entraînement pour un total de 98 images, puis un ensemble de même taille est utilisé pour tester les classificateurs produits. Finalement une expérience d'apprentissage incrémental est présentée, où sont introduites de nouvelles classes pendant le processus évolutif.

Les résultats de PADO sont très encourageants. Pour deux problèmes de reconnaissance d'objets complexes, sous différents angles et à différentes distances, PADO obtient entre 60% et 70% de reconnaissance. De plus, dans 80% à 90% des cas, la classe réelle comptait parmi les deux meilleures valeurs de sortie du classificateur. Ces résultats sont impressionnants puisqu'ils sont obtenus à partir de primitives extrêmement simples.

2.9 Plates-formes de développement

Bien que plus compétitive par le passé, la communauté des chercheurs de l'AM propose de plus en plus de projets ouverts, par soucis de reproductibilité et pour accélérer la recherche et le développement d'outils performants. Le site internet MLOSS (Sonnenburg, Braun et Ong, 2008) présente les divers logiciels actuellement disponibles dans un format ouvert. Parmi ces logiciels, le logiciel d'ED Weka a retenu notre attention. La section suivante décrit ses fonctionnalités.

2.9.1 Weka

L'acronyme Weka provient de « Waikato Environment for Knowledge Analysis » (Witten et Frank, 2005). Weka est un ensemble d'algorithmes d'AM pour l'ED. Les algorithmes peuvent être utilisés directement sur les bases de données ou appelées par le code de

l'utilisateur. Weka contient des outils pour le prétraitement, la classification, la régression, le « clustering », les règles d'association et la visualisation des données. Il est particulièrement bien adapté à développer de nouvelles techniques d'AM. C'est un logiciel libre qui utilise la licence publique générale GNU (GNU, 2007) ou en anglais « General Public licence » (GPL).

CHAPITRE 3

ALGORITHME DE PROGRAMMATION GÉNÉTIQUE

Afin de tirer profit des caractéristiques proposées par la littérature pour la reconnaissance de forme dans des images, puis pour permettre la comparaison de la PG avec d'autres classificateurs reconnus, nous avons décidé de ne pas poursuivre dans la voie de PADO (qui utilise l'image entière comme données d'entrée au système de classification). En effet, nous préférons utiliser la segmentation puis l'extraction de caractéristiques pour augmenter les performances de classification dans le cas d'images.

Rappelons qu'un de nos objectifs est d'appliquer des algorithmes de classification afin de classifier des objets dans des images et comparer les taux de reconnaissance. Pour ce faire, nous devons choisir les outils logiciels que nous utiliserons pour réaliser nos expériences. Dans un souci de comparaison juste, nous avons décidé d'utiliser une seule plate-forme informatique pour la classification des objets.

Ce chapitre présente la réalisation d'un module de PG s'insérant dans la plate-forme choisie et détaille les options disponibles, inspirées des algorithmes de PG les plus connus dont la plupart sont présentés dans les livres de références (Banzhaf et al., 1998; Koza, 1994; Langdon et Poli, 2002).

Le chapitre est divisé en sept sections. Nous commençons par discuter de certains choix quant à la création de l'algorithme : choix de la plate-forme de développement, objectifs et paramètres de conception. Nous poursuivons avec des caractéristiques de la PG, vues comme des options dans le cadre de notre algorithme : déroulement général, paramètres généraux et paramètres génétiques. Finalement nous clarifions comment notre algorithme a été intégré à la plate-forme de développement.

3.1 Choix de la plate-forme

Nous désirons utiliser une plate-forme informatique qui respecte les critères suivants :

- utilise une licence de code ouvert;
- participe à la recherche sur l'IA et l'exploration des données;
- contribue à former des collaborations dans le réseau des chercheurs;
- offre options et modularité;
- se prête au développement d'un module intégré de sélection de caractéristiques et de classification;
- possède déjà la plupart des algorithmes que nous voulons comparer;
- est facile d'utilisation, avec possibilité d'exécuter des processus en parallèle;
- est codée dans un langage de programmation pratique, portable et actuel;
- est conviviale pour la visualisation des résultats.

La plateforme de développement la plus appropriée dans ces conditions sera selon les préférences Weka, programmée par un groupe de recherche de l'université de Waikato, en Nouvelle-Zélande ou RapidMiner (Mierswa et al., 2006), outil compatible à Weka qui contient différents opérateurs pour l'AM.

RapidMiner a été développé après Weka et permet d'incorporer facilement les outils développés au moyen de Weka. Nous proposons donc de travailler avec Weka. Éventuellement, nous espérons intégrer nos contributions à RapidMiner.

La plate-forme de développement que nous avons choisie présente des implémentations de chacun des algorithmes de classification que nous avons sélectionnés, sauf l'implémentation de la PG. Nous avons donc décidé de développer notre propre algorithme de PG et de l'intégrer à Weka. Cet exercice nous permettra de gagner en expérience sur le développement d'un module de logiciel libre d'exploration des données et d'approfondir nos connaissances à propos des variantes de la PG.

3.2 Objectifs

Les qualités recherchées pour l'algorithme de PG sont la convivialité d'utilisation et de comparaison, la rapidité et la modularité. Ce dernier point est particulièrement important puisque l'algorithme de PG constituera un outil de recherche collaboratif. L'intérêt d'un logiciel libre axé sur la recherche est beaucoup plus grand s'il est facile de l'adapter aux particularités de ses expérimentations.

Plusieurs techniques novatrices utilisées pour la PG ont été proposées depuis son avènement. Ces techniques concernent tous les aspects de la PG, en particulier la structure du programme, sa création et son évolution génétique. Afin d'être en mesure de faire des études exhaustives des possibilités de la PG, il est pertinent d'implémenter toutes les techniques éprouvées et celles qui sont prometteuses, ou du moins préparer leur éventuelle intégration à notre algorithme.

3.3 Conception

Le langage de programmation utilisé par Weka est Java, un langage orienté objet rapide et portable. Ses différentes structures, dont la principale est la classe, présentent des attributs (données simples ou objets créés selon le modèle d'une classe) et des méthodes (fonctions représentant une action de l'objet). Le langage orienté objet a la particularité de permettre l'héritage, par lequel une classe partage ses qualités avec les classes héritières. Les classes héritières peuvent offrir des variantes pour des fonctions définies dans la classe ancêtre, de façon à se comporter différemment dans l'appel d'une même méthode. Ce polymorphisme facilite l'implantation d'options puisqu'il permet à l'algorithme d'utiliser toujours la même séquence de fonctions : l'usager ou l'algorithme choisit une classe parmi l'ancêtre et ses héritières et détermine ainsi le code effectif de chaque fonction.

Nous présentons deux diagrammes de classes, aux figures 3.1 et 3.2, inspirés du « Unified Modelling Language » (UML) (Object Management Group, 2007) afin d'illustrer la structure et les relations entre les objets de notre algorithme. Les dossiers représentent des librairies de

classes. Les classes peuvent posséder des attributs (on présente d'abord l'attribut, puis le type de données) et des méthodes (les méthodes sont toujours suivies de parenthèses vides). Les classes en italique représentent des classes abstraites dont seules les classes héritières peuvent permettre la création d'objet. Le signe plus (+) indique une donnée ou méthode publique alors que le signe moins (-) indique une donnée ou méthode privée. Finalement, le losange exprime une relation de possession entre les données (la classe de départ du lien avec losange manipule un objet de la classe d'arrivée du lien). La flèche révèle une relation d'héritage, l'héritier pointant vers son ancêtre. Notez que les attributs et méthodes des classes développées ne sont pas présentés exhaustivement afin d'éviter de surcharger les diagrammes.

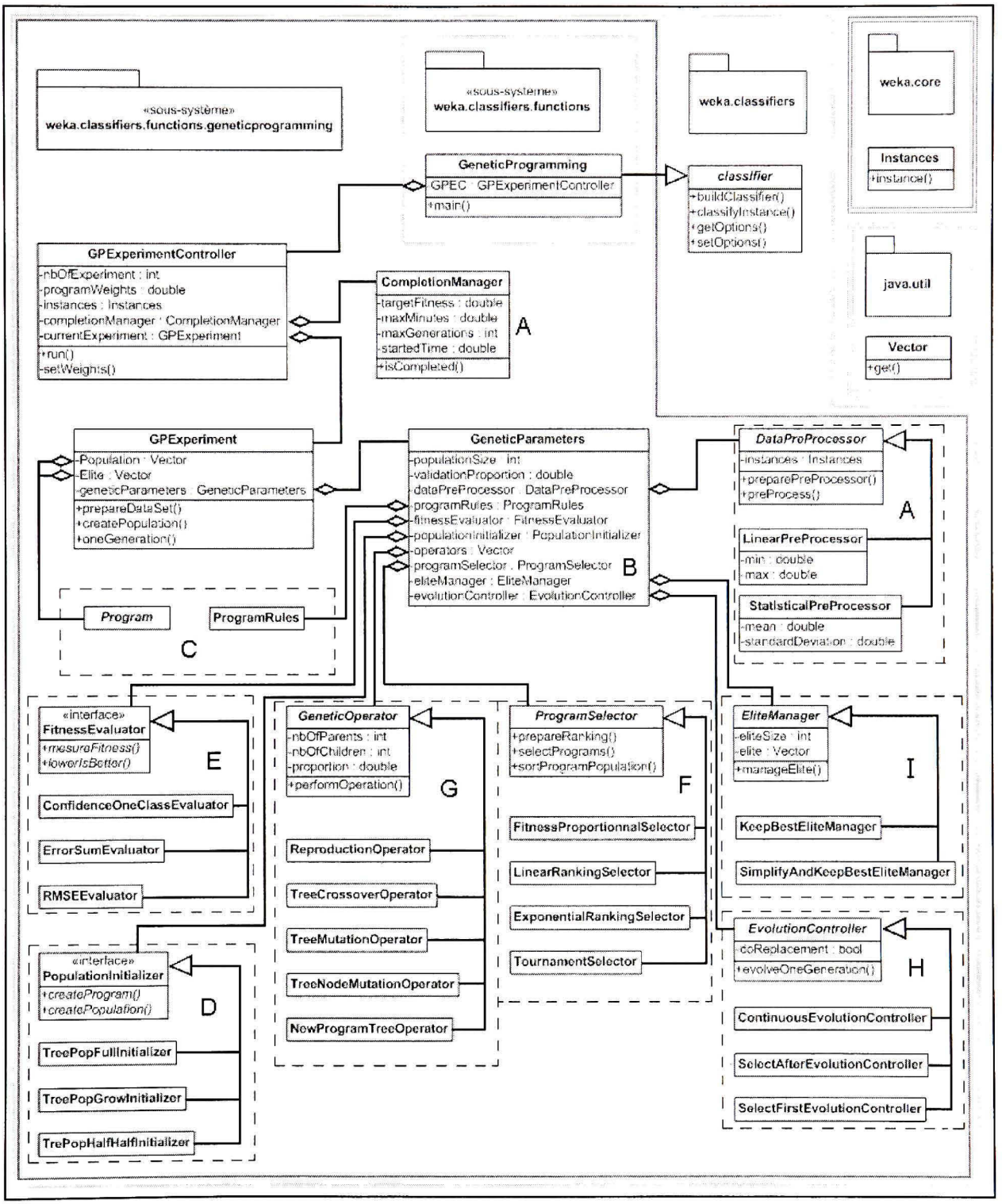


Figure 3.1 *Diagramme de classes pour la librairie geneticprogramming.*

La figure 3.1 présente les classes majeures de notre algorithme. Le tableau suivant présente les sections contenant les détails du fonctionnement de chaque classe importante :

Tableau 3.1

Description et section de référence pour les classes majeures

Ref.	Description et section détaillée
A	*CompletionManager* est la classe responsable de détecter l'atteinte des critères d'arrêt de l'algorithme. *DataPreProcessor* est une classe qui transforme les données d'entrées avant de les faire suivre à l'algorithme de classification. Nous discutons de ces fonctionnalités à la section 3.5.
B	La classe *GeneticParameters* contient certains paramètres généraux (*Voir* section 3.5) et tous les paramètres génétiques associés à la PG. Ces derniers sont présentés un à un à travers la section 3.6.
C	Les classes *Program*, *ProgramRules* et leurs interactions avec d'autres classes mineures sont présentées à l'aide d'un diagramme de classe (*Voir* figure 3.2). Nous expliquons le fonctionnement de la grammaire et présentons l'alphabet disponible pour les programmes aux sections 3.6.1 et 3.6.2.
D	*PopulationInitializer* est une interface utilisée pour créer les programmes initiaux, au tout début de la PG. Les différentes options pour la création de programmes sont présentées à la section 3.6.4.
E	L'interface *FitnessEvaluator* calcule l'adéquation, ou la pertinence d'un programme. Nous présentons différentes fonctions d'évaluation de l'adéquation à la section 3.6.5.
F	*ProgramSelector* est responsable de la sélection des programmes pour la reproduction. Nous décrivons à la section 3.6.6 les quatre méthodes de sélection que nous avons implantées.
G	Les classes héritières de *GeneticOperator* sont les opérations génétiques qui permettent de transformer les programmes. Nous présentons les cinq opérateurs disponibles pour notre algorithme à la section 3.6.7.
H	La classe *EvolutionController* est responsable de la succession des générations de programmes. Nous clarifions les différences entre les scénarios évolutifs disponibles à par les classes héritières à la section 3.6.8.
I	L'*EliteManager* permet de conserver en mémoire les programmes ayant obtenus la meilleure adéquation, même si ces programmes ne font plus parti de la population. Cette fonctionnalité est décrite à la section 3.6.9.

La colonne *Ref.* du tableau 3.1 fait référence aux lettres et aux encadrés en ligne pointillée de la figure 3.1. Les classes *Program* et *ProgramRules* (notées *A* sur la figure 3.1) puis les classes qui leurs sont reliées sont schématisées à la figure 3.2.

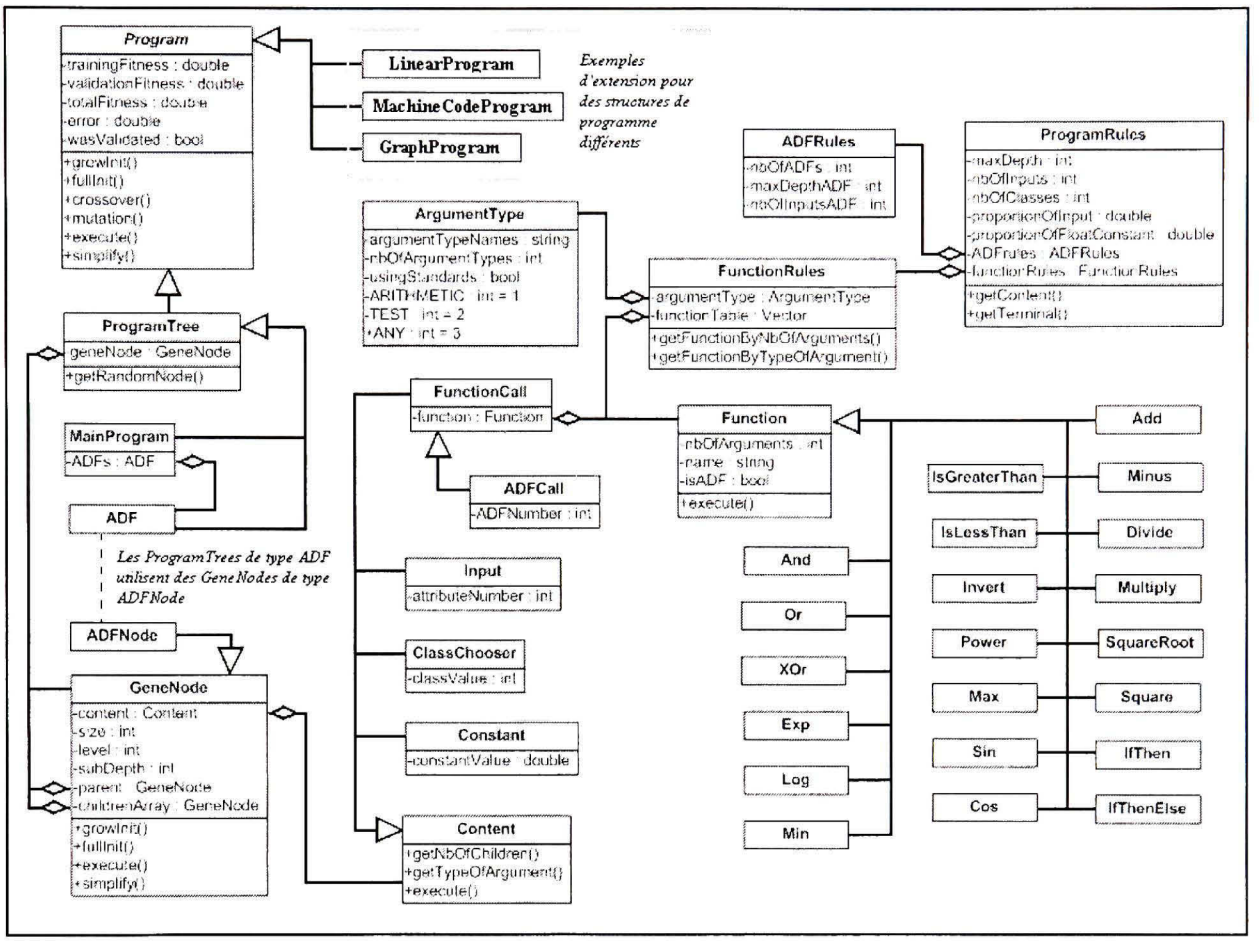


Figure 3.2 ***Diagramme de classes situant Program et ProgramRules.***

Dans les diagrammes des figures 3.1 et 3.2, les classes héritières représentent pour la plupart des options possibles pour la PG. Le plus souvent, c'est l'utilisateur qui choisira l'option désirée.

3.4 Déroulement général de la PG

Le module de PG qui sera développé pour Weka contiendra tous les mécanismes nécessaires à la création de programmes performants pour la prédiction. Voici la séquence d'opérations générale de notre algorithme, qui s'adapte aux options choisies par l'usager :

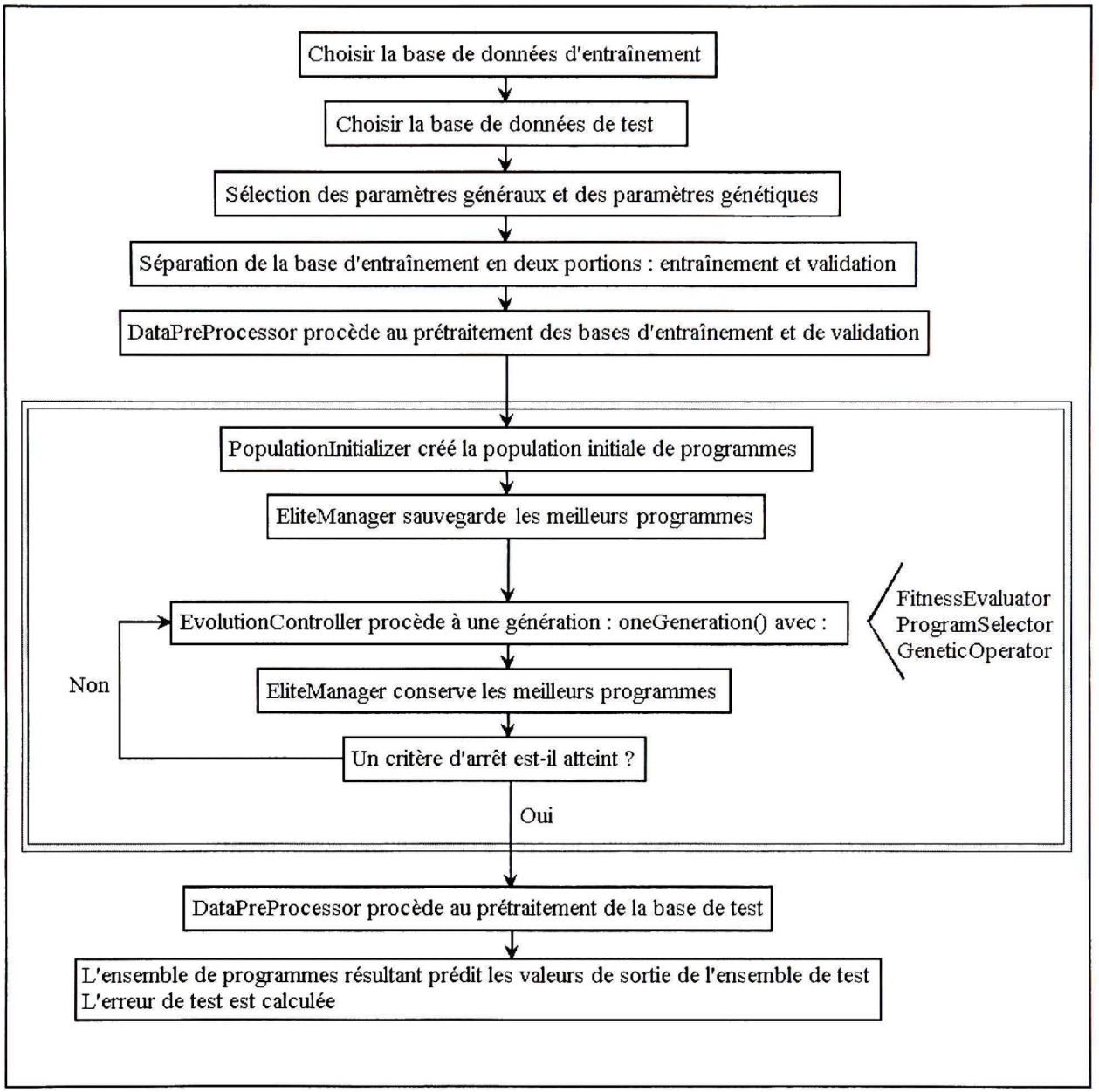


Figure 3.3 ***Déroulement de l'algorithme de PG.***

DataPreProcessor, *PopulationInitializer*, *EliteManager*, etc. sont des noms de classes ancêtres dont le comportement varie selon l'option choisie (*Voir* figure 3.1 et tableau 3.1). La portion encadrée de la figure 3.3 peut se répéter plus d'une fois dans le cas d'un problème de classification (*Voir* section 3.5.1).

Afin de choisir les paramètres voulus pour la PG, l'usager peut procéder de deux façons différentes. Il peut démarrer une évolution de PG au moyen d'une ligne de commande en spécifiant les paramètres selon des codes (fournis dans la documentation de programmation). L'autre choix, plus simple, est d'utiliser l'interface disponible dans Weka pour gérer les options des classificateurs. En voici une illustration :

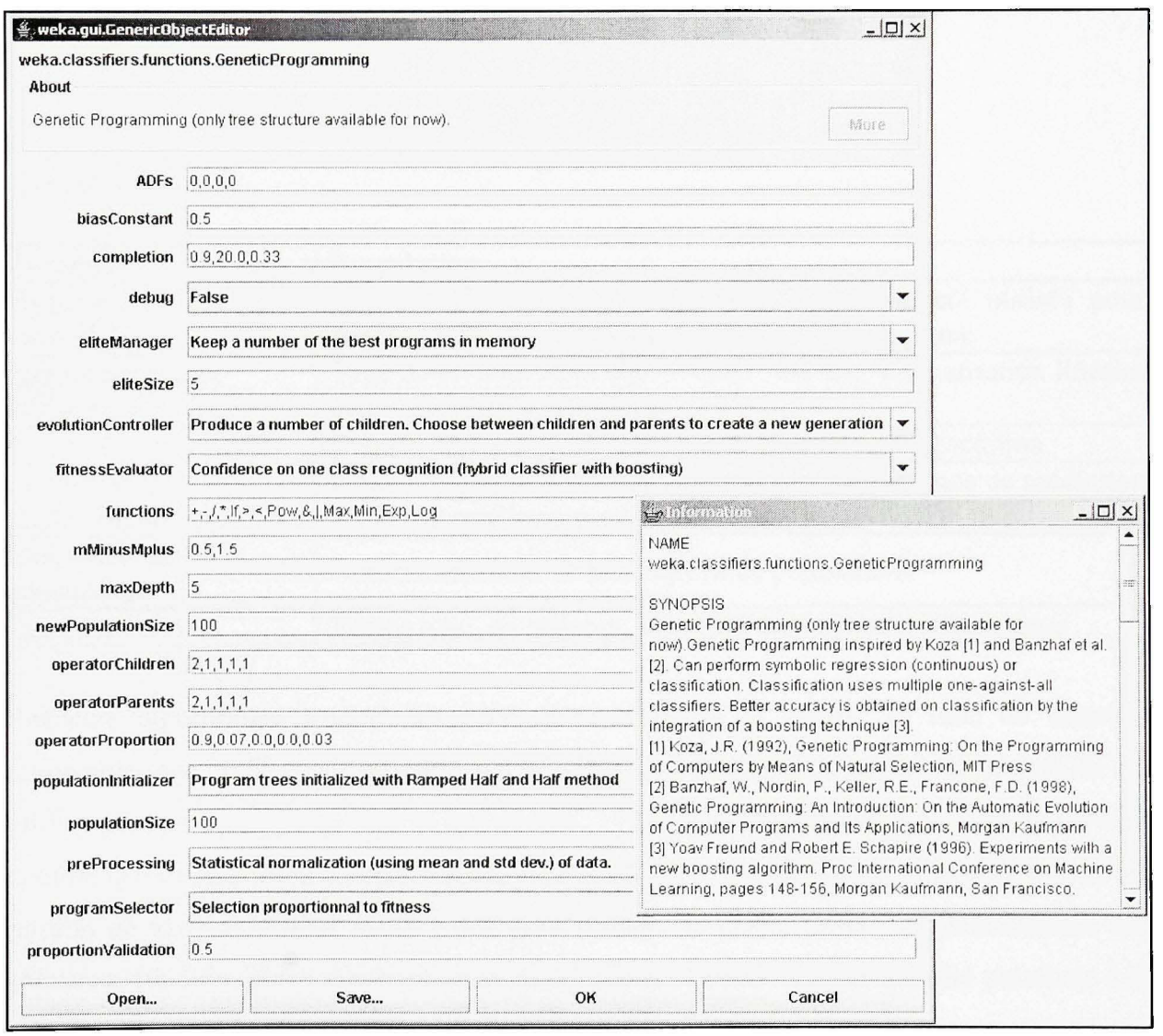


Figure 3.4 ***Interface à l'usager graphique pour l'algorithme de PG.***

3.5 Paramètres généraux

Certains paramètres ont une influence globale sur l'algorithme de PG. Ces paramètres déterminent collectivement la rapidité et la complexité de l'algorithme. Ils concernent le prétraitement et le nombre d'itérations, ou d'itérations possibles, pour la ou les évolutions de programmes (aussi appelés individus). Chaque paramètre joue un rôle très important.

Tableau 3.2

Paramètres généraux de la PG

Paramètre	**Description**
Proportion de validation	Proportion de la base de données d'entraînement utilisée pour estimer le pouvoir de généralisation d'un programme
Prétraitement des données	Type de prétraitement des données : aucune, normalisation linéaire ou normalisation statistique
Taille de la population	Nombre de programmes conservés pour chaque génération
Performance de prédiction désirée	Critère d'arrêt de l'évolution relatif à la performance de prédiction du ou des meilleurs programmes obtenus
Maximum de générations	Nombre maximal de successions de populations
Temps maximal	Temps total, en minutes, alloué à l'évolution de programmes

Plusieurs algorithmes comme les RNA tirent avantage à séparer la base de données d'apprentissage en deux : la première partie sert pour l'entraînement alors que la seconde est utilisée en validation (Jain, Mao et Mohiuddin, 1996). Comme pour les RNA, la PG ne tient compte que de la portion d'entraînement pour modifier le classificateur qu'il fait évoluer. La portion de validation n'est utilisée que pour calculer la performance de généralisation des solutions. En effet, puisque cette portion de la base de données n'a pas déjà été présentée à la solution développée, la performance de validation informe sur la performance espérée du système pour d'éventuelles données nouvelles. En ce qui concerne notre algorithme de PG, le gestionnaire d'élite, qui conserve les meilleurs programmes en mémoire (*Voir* section 3.6.9), considère à la fois la performance d'entraînement et de validation des programmes.

Avant même de débuter avec la création des programmes, il peut être bénéfique de transformer la base de données d'entraînement. En effet, si l'échelle et la plage de valeur d'une caractéristique diffèrent beaucoup de celles des autres caractéristiques, il est normalement plus difficile pour l'algorithme de prédiction de tenir compte de cette caractéristique. Le SVM (Burges, 1998), le K plus proches voisins (Shakhnarovich, Darrell et Indyk, 2006) et la PG (Banzhaf et al., 1998) performent généralement mieux si les données d'apprentissage sont normalisées. D'ailleurs, Weka offre des options de prétraitement pour ses implantations de SVM et de K plus proches voisins. Notre algorithme propose au choix une normalisation linéaire (toutes les données seront alors comprises entre 0 et 1) ou une normalisation statistique (la moyenne des données sera 0 et leur écart type sera 1).

La PG se décrit en termes de population, d'individu (synonyme de programme), de génération, de sélection et de reproduction. Ces termes sont empruntés au vocabulaire de la génétique biologique. La génération sert à décrire les successions de populations. La taille de la population, en nombre de programmes, spécifie le nombre d'individus conservés par l'algorithme de PG pour chaque génération. Le nombre de programmes dans la population peut varier entre les générations, lorsque la sélection et la reproduction sont en cours.

Chacune des évolutions de programmes doit avoir un ou des critères de fin d'évolution. Dans le monde de la recherche scientifique, le critère sélectionné est souvent un nombre maximal de générations. Pour une application plus pratique, il peut être avantageux de choisir un critère relié à la performance et à la disponibilité des machines de calcul. Ainsi, nous considérons utile un critère de temps maximal alloué au développement d'une solution par PG. Aussi, rien ne sert de continuer la recherche si l'algorithme a atteint les objectifs de classification : on peut donc arrêter la recherche lorsqu'une performance de prédiction désirée a été obtenue. Finalement, si une solution offre une performance parfaite (erreur de 0 ou taux de classification de 100%), l'algorithme ne pourra plus guider l'évolution des programmes afin d'améliorer la reconnaissance. Dans ce cas, il vaut mieux arrêter la recherche.

3.5.1 Combinaison de programmes

Une fois qu'un programme de prédiction a été obtenu à l'aide d'une recherche de PG, on peut l'utiliser directement pour la classification, ou continuer de créer des programmes afin de les combiner en une solution plus performante par la suite.

Il existe plusieurs méthodes pour combiner des systèmes de prédictions statistiques. Par exemple, on peut utiliser une moyenne des résultats de plusieurs systèmes de régressions dans le cas de prédiction en valeur continue, ou utiliser un système de vote pour les résultats de plusieurs classificateurs (Duda, Hart et Stork, 2000). Plusieurs de ces méta algorithmes sont déjà disponibles dans Weka, notre plateforme de développement.

Ici, nous nous intéressons aux mécanismes qui peuvent être intégrés au processus de la PG et qui nous permettent de faire une combinaison avantageuse des programmes obtenus à la suite de recherches de solutions. Nous avons étudié en particulier la combinaison de classificateurs issus de la PG.

Dans le cas d'un problème de classification à n classes, il existe plusieurs approches pour la résolution. Voici les trois approches les plus répandues :

1. Développer un classificateur unique qui donne, en sortie, la classe du nouvel échantillon fourni en entrée;

2. Développer un classificateur par classe d'échantillons (il y aura donc n classificateurs). Chaque classificateur est responsable de reconnaître une classe en particulier;

3. Développer un classificateur par paire de classes (le nombre de classificateurs est donné par la somme itérative des i, de $i = 1$ jusqu'à $i = n - 1$). Chaque classificateur est responsable de départager deux classes en particulier.

Deux chercheurs ont réalisé des expériences pour tester les trois méthodes dans le contexte de reconnaissance de formes par PG (Teredesai et Govindaraju, 2004). Ils concluent que la méthode *2*, qui utilise un classificateur par classe d'échantillons, constitue la meilleure approche. C'est donc cette variante qui sera utilisée par notre algorithme pour les problèmes de classification.

Avec cette méthode, les classificateurs obtenus par la PG doivent avoir un certain type de valeur de sortie. Deux approches sont encore une fois proposées :

1. Classificateur binaire : son résultat est 0, s'il prédit que l'échantillon ne fait pas partie de la classe, ou 1 s'il prédit que l'échantillon fait partie de la classe;

2. Classificateur à sortie continue : le résultat est une valeur décimale (par exemple entre 0.0 et 1.0) qui représente la confiance avec laquelle le classificateur associe l'échantillon avec la classe désignée;

Lorsqu'un nouvel échantillon est présenté, chaque classificateur doit prédire si l'échantillon fait partie de la classe pour laquelle il a été entraîné. C'est le classificateur jumelé a la valeur de sortie la plus grande qui détermine la classe du nouvel échantillon. En cas d'égalité, c'est le classificateur qui offre la plus grande performance de reconnaissance avec l'ensemble d'entraînement qui attribuera la classe. En pratique, nous avons intégré à notre algorithme uniquement le classificateur à sortie continue.

L'étude du système de PG de Teller et Velloso, PADO (section 2.8.1), présente aussi un autre type de combinaison que nous appellerons orchestration. L'orchestration utilise plusieurs classificateurs à sortie continue par classe d'échantillons. Nous introduirons dans notre module, une implémentation de l'orchestration de programmes telle que décrite dans (Teller et Veloso, 1997), puis, nous étudierons ses performances.

Finalement, nous intégrerons le méta algorithme de boosting au processus évolutif de la PG. Plusieurs études concernant l'implantation d'une méthode de boosting pour la PG font état de gains considérables au niveau de la performance de reconnaissance et du temps de calcul de l'algorithme (Folino, Pizzuti et Spezzano, 2004) (Paris, Robilliard et Fonlupt, 2001). Un méta algorithme de boosting comme celui disponible dans Weka peut combiner plusieurs classificateurs obtenus par PG. Par contre, l'intégration des principes du boosting à l'intérieur même du processus de PG permet une plus grande économie de ressources. Voici son déroulement général, sous forme de pseudo code :

PG avec boosting intégré

C = nombre de classes du problème
P = nombre de programmes voulus pour le boosting
Ensemble d'entraînement T = toutes les données d'entraînement disponibles
N = nombre total d'échantillons dans T

À faire C fois (j = *1 jusqu'à* C)

Vider la population de programmes *POP*

Initialiser le « poids » W de chaque échantillon i avec $W_i = \frac{1}{N}$

À faire P fois (k = *1 jusqu'à* P)

Si *POP* est vide, remplir *POP* avec un nouvel ensemble de programmes

Faire évoluer un programme qui reconnaît la classe j par la PG (dans cette version de PG, le calcul de l'adéquation considère le poids W_i de chaque échantillon à classifier : *Voir* section 3.6.5), en utilisant T et *POP*.

Calculer l'erreur du meilleur programme, E_{jk}, sur l'ensemble d'apprentissage, son facteur α_{jk} et ensuite le poids W_i de chaque échantillon d'apprentissage selon la méthode AdaBoost (*Voir* pseudo code de la section 2.7.2). T n'est par contre pas transformé.

Fin pour P
Fin pour C

On peut classifier en utilisant la réponse la plus forte selon une somme pondérée des sorties des programmes par classe (*Voir* équation 3.1).

À la fin de la routine, on obtient $P \bullet C$ programmes, soit P programmes pour chaque classe C. Le score de classification pour la classe j est obtenu par la somme pondérée par α_{jk} de la sortie de chaque programme jk. Le j qui obtient le score le plus grand indique la classe de l'échantillon présenté :

$$\max_{j}\left(\sum_{k=1}^{P}\left(a_{jk}\cdot sortie_{jk}\right)\right) \tag{3.1}$$

Un gain en ressources computationnelles du boosting intégré par rapport au boosting de PG standard est obtenu puisque la PG n'a pas besoin de recommencer l'évolution à zéro lorsqu'elle recherche un programme performant sur l'ensemble d'apprentissage marqué de nouveaux poids. Il est probable qu'un programme performant pour l'ensemble nouvellement pondéré existe déjà dans la population développée lors de la recherche précédente.

Le chapitre 5 présente la comparaison des performances de l'algorithme de PG avec orchestration et celui avec boosting intégré.

3.6 Paramètres génétiques

Les paramètres génétiques concernent la structure, la création ou la transformation des programmes. Le choix de ces paramètres influence l'évolution des programmes, donc leur capacité à converger vers une solution performante pour le problème présenté. Les paramètres génétiques comprennent habituellement le type de programme, la grammaire, la règle de création, la fonction d'adéquation, la sélection, les opérateurs génétiques, le scénario évolutif et les paramètres liés à l'élitisme.

Un premier paramètre influence énormément la forme que peut prendre les autres paramètres de la PG. Il s'agit du type de programme. Tel que mentionné à la section 2.6.6, des structures linéaires, en graphe ou en code machine ont été proposées pour la PG. Notre algorithme adoptera quant à lui la structure en arbre, construction qui a servi de base pour le développement de la PG (Koza, 1990). Les fonctions de notre module de PG sont préparées

pour rester utilisables dans l'éventualité d'implémentation d'un autre type de structure pour les programmes.

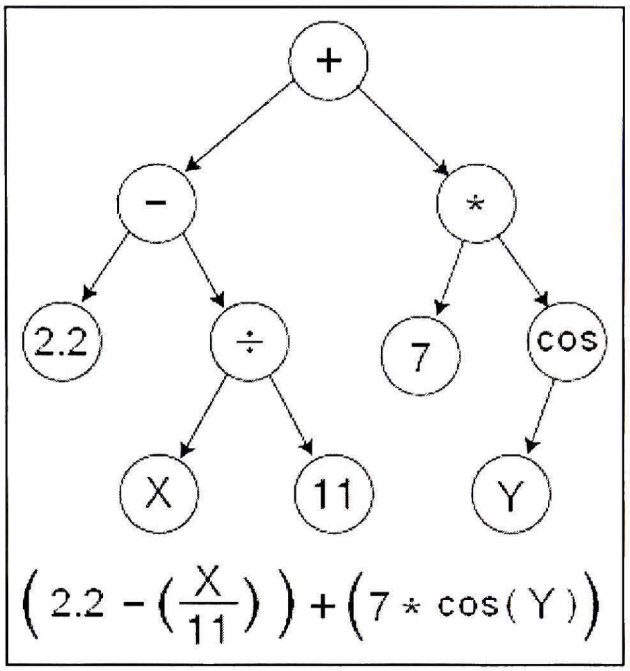


Figure 3.5 ***Programme structuré en arbre et son interprétation.***
(Tiré de (Wikipédia, 2007a))

Un arbre est composé de nœuds connectés entre eux par des liens. La structure en arbre se distingue de la structure en graphe par le fait qu'elle ne boucle jamais sur elle-même. Chaque nœud est lié à un ou plusieurs nœuds dits « enfants ». Dans la construction d'un programme à partir de l'arbre, les nœuds enfants représentent les arguments de la fonction contenue dans le nœud parent. Les nœuds qui n'ont pas d'enfants sont alors appelés « feuilles » et contiennent un élément terminal comme une constante ou une caractéristique issue de l'échantillon. Un seul nœud est dépourvu de parent. Ce nœud appelé « racine » est le dernier nœud dans l'évaluation du programme d'un arbre. Ce que nous appellerons, dorénavant, profondeur d'un arbre est constitué du nombre de niveaux qu'il contient. Cette profondeur est définie récursivement comme le maximum des profondeurs de ses sous arbres, augmenté de 1. Par exemple, le programme en arbre illustré à la figure 3.5 a une profondeur de 4.

3.6.1 Grammaire

La grammaire de la PG est divisée en deux parties. Une partie est appelée « alphabet », puisqu'elle décrit les éléments de bases qui peuvent se retrouver dans une structure. L'autre partie consiste en une série de règles qui permettent de créer ou de vérifier la conformité d'une structure à la grammaire. Commençons par décrire les règles appliquées à notre structure en arbre.

Afin d'éviter les erreurs d'exécution lors du déroulement de la PG, nous devons nous assurer que les programmes développés par notre algorithme seront toujours syntaxiquement corrects (selon le langage de programmation utilisé). Pour ce faire, il faut que chaque nœud qui n'est pas une feuille contienne un élément de l'alphabet qui possède des liens vers des enfants, et que chaque nœud feuille contienne un élément de l'alphabet qui n'a pas de lien vers un nœud enfant. Il est également nécessaire que l'exécution de chaque nœud soit exempte de possibilité d'erreur (par exemple division par zéro). Finalement, le résultat de l'exécution de chaque nœud doit respecter le format utilisé pour les valeurs d'entrée de toutes les fonctions de l'alphabet. Pour notre algorithme les valeurs $-\infty$ à $+\infty$ sont acceptables (la valeur *NaN* « Not a number » de Java n'est pas acceptable).

Ensuite, il apparaît judicieux de mettre en place des mécanismes pour augmenter les chances de produire des programmes performants. Par exemple, il peut être avantageux d'augmenter les chances qu'un nœud contenant une fonction de type conditionnel (par exemple l'énoncé « if ») ait comme enfant un nœud contenant une fonction de test (par exemple l'opérateur « plus grand que »). Ainsi, nous avons mis au point une classe qui gère les types d'arguments des fonctions. Les types par défaut sont « arithmétique », « test » ou « indifférent ». Nous y reviendrons à la section 3.6.4 qui porte sur la règle de création et de transformation des programmes.

3.6.2 Alphabet

En général, l'alphabet de la PG est composé de terminaux et de fonctions. Pour une structure en arbre, les feuilles contiennent des terminaux, alors que tous les nœuds qui ne sont pas des feuilles contiennent des fonctions.

Les terminaux n'ont jamais d'argument ou d'enfant. Ils ne font que rapporter au nœud parent la valeur qu'ils contiennent. Dans notre algorithme, il n'existe que deux sortes de terminaux : les valeurs constantes et les caractéristiques d'échantillons. Un nœud pourra donc contenir n'importe quelle valeur constante, ou n'importe quelle caractéristique qui définit les échantillons de la base de données utilisée. Le tableau 3.3 illustre les fonctions qui sont disponibles pour notre algorithme :

Tableau 3.3

Fonctions disponibles pour l'algorithme de PG

Nom	**Signe**	**Type**	**Nb args**	**Type arg**	**Commentaire**
Addition	+	A	2	A, A	
Soustraction	-	A	2	A, A	
Division	/	A	2	A, A	Division protégée : dans le cas d'une division par 0, le résultat est 1
Multiplication	*	A	2	A, A	
Carré	Sq	A	1	A	L'argument est élevé au carré
Racine carrée	Sqrt	A	1	A	Si l'argument est négatif, le résultat est *-Sqrt(-argument)*
Puissance	Pow	A	2	A, A	Le résultat est le premier argument élevé à la puissance du deuxième argument
Sinus	Sin	A	1	A	Sinus de l'argument (en radians)
Cosinus	Cos	A	1	A	Cosinus de l'argument (en radians)
Maximum	Max	A	2	A, A	Le résultat est la plus grande valeur parmi les arguments
Minimum	Min	A	2	A, A	Le résultat est la plus petite valeur parmi les arguments
Exponentielle	Exp	A	1	A	Le résultat est le nombre d'Euler élevé à la puissance de l'argument

Nom	Signe	Type	Nb args	Type arg	Commentaire
Logarithme	Log	A	1	A	Le résultat est le logarithme naturel de l'argument. Si l'argument est inférieur à 0 le résultat est 0.
Si, alors	If	T	2	T, I	Si le premier argument est différent de 0, alors le résultat est le deuxième argument, sinon le résultat est 0.
Si, alors, sinon	If3	T	3	T, I, I	Si le premier argument est différent de 0, alors le résultat est le deuxième argument, sinon le résultat est le troisième argument.
Plus grand que	>	T	2	A, A	Si le premier argument est plus grand que le deuxième, le résultat est 1, sinon, le résultat est 0.
Plus petit que	<	T	2	A, A	Si le premier argument est plus petit que le deuxième, le résultat est 1, sinon, le résultat est 0.
Non	!	T	1	T	Si l'argument est 0, le résultat est 1, sinon, le résultat est 0.
Et logique	&	T	2	T, T	Si les deux arguments sont différents de 0, le résultat est 1, sinon, le résultat est 0.
Ou logique	|	T	2	T, T	Si les deux arguments sont 0, le résultat est 0, sinon, le résultat est 1.
Non-ou logique	Xor	T	2	T, T	Si un des deux arguments est 0 et l'autre est différent de 0, le résultat est 1, sinon, le résultat est 0.
Fonction définie automatiquement	ADF	I	0-?	I	Le nombre d'argument est défini par l'usager (*Voir* section 3.6.3).

La colonne *Type* indique le type de la fonction, *Nb arg* indique le nombre d'arguments, alors que la colonne *Type arg* dénote le type des arguments de la fonction. Les codes suivants sont utilisés pour les types : *A* pour Arithmétique, *T* pour Test et *I* pour Indifférent

Les fonctions présentées ici se conforment bien aux règles énoncées pour la production de programmes avec structure en arbre. Elles pourraient facilement être utilisées pour un autre type de structure comme les programmes linéaires. Les fonctions pour la PG peuvent prendre une toute autre forme lorsque le type de structure est différent (dans le cas de graphe avec mémoire indexée par exemple (Teller et Veloso, 1997)) ou si le problème l'exige (cas du

problème de la fourmi artificielle (Koza, 1992)). Les classes de notre algorithme s'adaptent facilement à ces situations.

3.6.3 Fonctions définies automatiquement

Koza a proposé les « Automatically Defined Functions » (Koza, 1994), ou Fonctions Définies Automatiquement (FDA). Il s'agit de programmes qui se retrouvent jumelés avec un programme principal ou accumulés en librairie et qui peuvent être invoqués par les programmes de la population comme n'importe quelle autre fonction. Les FDA ont la particularité de voir leur nœud feuille soumis à une règle différente que les programmes normaux. Les terminaux des FDA peuvent être des constantes, mais pas des caractéristiques. Les caractéristiques sont remplacées par un argument (du nœud d'appel de la FDA), selon le nombre d'arguments de la FDA. La figure 3.6 illustre l'utilisation d'une FDA par un programme principal :

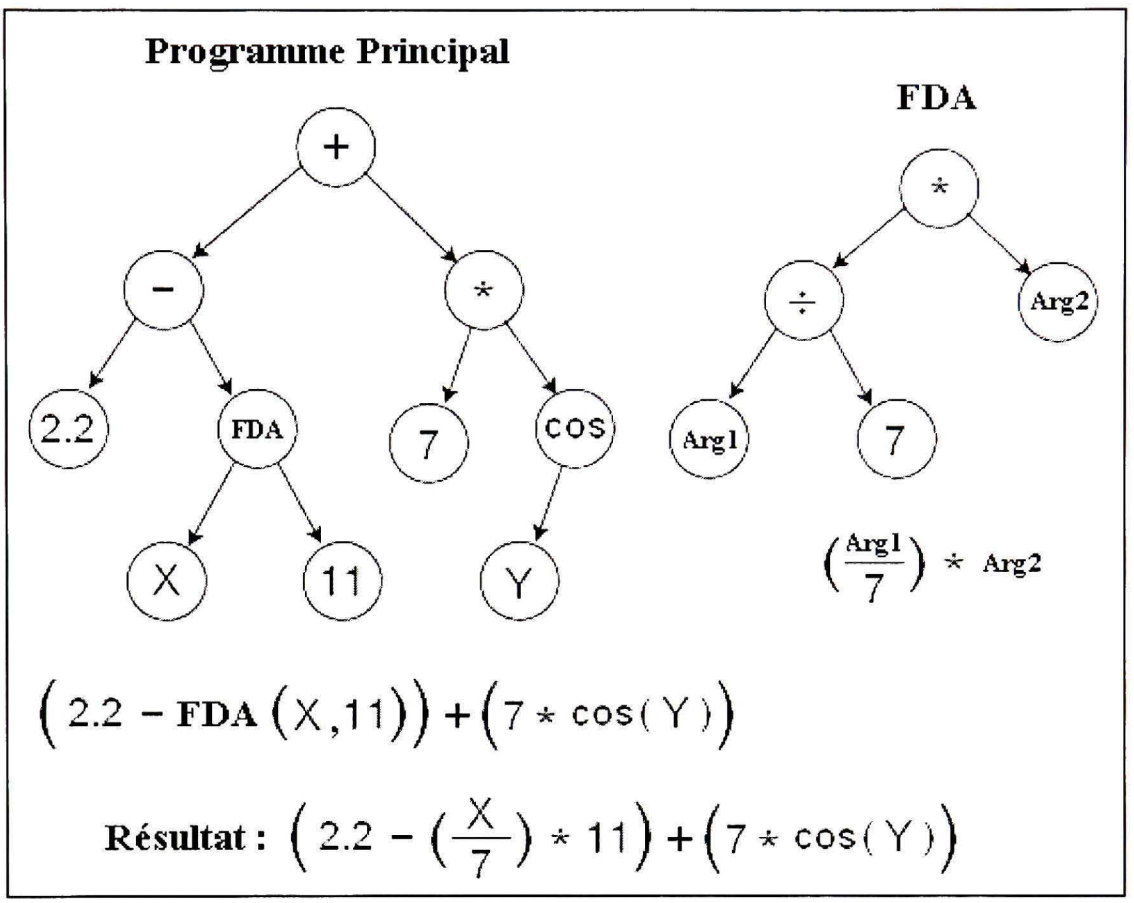


Figure 3.6 ***Programme principal avec FDA et interprétation.***

La FDA de la figure 3.6 a comme arguments X et 11. L'équation mathématique illustrant les opérations du programme « résultant » est présentée au bas de la figure.

L'intérêt de la FDA est de découvrir des parties de programme réutilisables. L'évolution de programmes par PG laisse observer une tendance à « décortiquer » les problèmes complexes en sous problèmes. Dans les cas où la résolution de plusieurs sous problèmes nécessite des opérations analogues, il sera beaucoup plus facile au processus de PG de faire appel à une même FDA à plusieurs reprises que d'utiliser les opérations génétiques normales pour reproduire les sous programmes équivalents aux FDA.

Dans notre implémentation, les FDA sont disponibles. Si elles sont utilisées, chaque programme est « propriétaire » d'un certain nombre de FDA. Les FDA ont des contraintes

qui leur sont propres (pour la création et la transformation), mais calquées sur celles des programmes principaux.

3.6.4 Règles de création et de transformation

La première étape d'une recherche de programme pour le PG est de créer une population de programmes aléatoires. Ces programmes ne sont pourtant pas entièrement aléatoires car ils doivent respecter la grammaire et certaines règles de création. Cette section présente les règles que nous utiliserons. Ces repères permettent de conserver les programmes dans un état fonctionnel et de limiter l'espace mémoire alloué à la population de programmes.

En premier lieu, le processus récursif de création de nœuds enfants à la suite de la sélection aléatoire d'une fonction doit avoir une fin. Il existe plusieurs façons de résoudre ce problème. D'abord, on peut recourir à la solution la mieux connue et la plus utilisée qui consiste à imposer une valeur maximale pour la profondeur des arbres (Koza, 1992). Ainsi, tous les nœuds créés à la profondeur maximale choisie devront contenir un terminal (et non pas une fonction, qui ne pourrait avoir d'arguments). Il est aussi possible de limiter le nombre total de nœuds utilisés par la population entière (Silva et Costa, 2005). Bien que cette deuxième solution s'avère intéressante, elle ne s'est pas révélée plus performante que la solution traditionnelle. Par conséquent, notre algorithme opérera avec l'imposition d'une profondeur maximale.

En deuxième lieu, l'usager peut définir deux paramètres ou laisser leur valeur par défaut. Le premier correspond à la proportion de nœuds feuilles qui contiendra une caractéristique, par opposition à une constante. Cette valeur peut être choisie entre 0 et 1 (la valeur par défaut est 0.5 : la moitié des nœuds feuilles contiendra des caractéristiques). Le second se rapporte à la proportion des constantes qui seront des valeurs décimales, en opposition à celles qui seront des entiers (la valeur par défaut est 1.0 : toutes les constantes sont décimales). Ces paramètres ont une influence sur le temps de convergence de la PG, en fonction des plages de valeurs des bases de données utilisées. Nous tenons aussi à préciser que les valeurs par défaut

ont été choisies pour des bases de données d'apprentissage et de test normalisées (cette normalisation est facilement réalisable en pratique : V*oir* section 3.5).

Ensuite, il faut souligner le fait que le type d'argument influence le choix d'une fonction par rapport à une autre. Le tableau 3.3 présente les types d'arguments pour chaque fonction de base de notre alphabet. Ces informations nous permettent de choisir une fonction du type approprié pour le nœud enfant d'une autre fonction. Évidemment, la règle de la profondeur maximale, qui impose des terminaux aux nœuds feuilles, a préséance sur la sélection d'une fonction.

Tous les contenus qui se qualifient, lors des choix concernant les terminaux ou les fonctions, puis pour sélectionner une fonction selon le type d'arguments, ont autant de chance d'être sélectionnés.

La littérature relate trois méthodes de création de programmes (Banzhaf et al., 1998). Notre algorithme offre la possibilité de choisir une de celles-ci :

1. Méthode complète ou « full » : elle consiste à créer des arbres qui auront toujours la profondeur maximale. Le contenu choisi pour un nœud est toujours une fonction, sauf lorsque la profondeur maximale est atteinte;

2. Méthode de croissance ou « grow » : les arbres ainsi produits n'auront pas toujours la profondeur maximale. Les nœuds qui n'ont pas encore atteint la profondeur maximale peuvent contenir des terminaux (constante ou caractéristique). La probabilité pour ces nœuds de contenir un terminal est le rapport entre le nombre de caractéristiques et la somme du nombre de caractéristiques et du nombre de fonctions;

3. Méthode de rampe moitié-moitié : elle permet à une population initiale de contenir une plus grande diversité de programmes. La population est d'abord divisée en groupes de programmes de taille égale. Chaque groupe a une profondeur maximale

différente, soit la suite des entiers de 2 jusqu'à la valeur maximale choisie (par exemple pour une valeur maximale choisie de 6, il y aura un groupe pour chacune des profondeurs maximales de 2, 3, 4, 5 et 6). Chaque groupe produit la moitié de ses programmes selon la méthode complète et l'autre moitié avec la méthode de croissance.

La figure 3.7 illustre deux programmes créés respectivement avec la méthode de croissance et la méthode complète, avec une profondeur maximale de 3 :

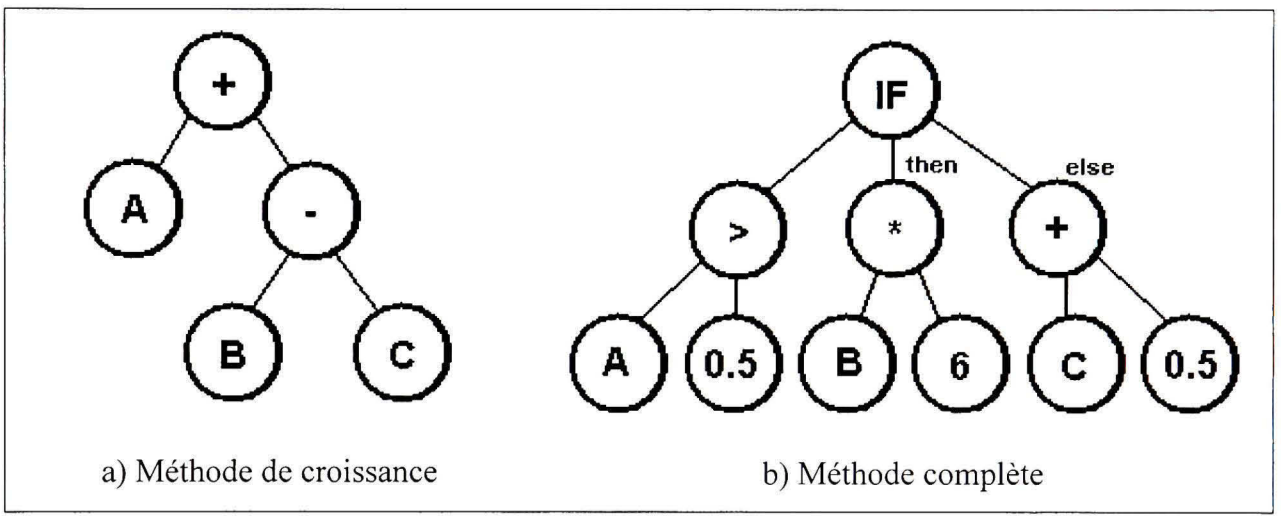


Figure 3.7 ***Programmes en arbres créés selon deux méthodes.***
(Tiré de Levasseur, 2005)

Afin d'illustrer un peu mieux le processus récursif de création de programme, nous en présentons ici le pseudo code :

Algorithme récursif de création d'un programme en arbre

N = nœud en cours
TF = type de fonction voulue

N = nœud racine, *TF* = Indifférent

(emplacement *Retour*)
Si profondeur de *N* = profondeur maximale

N contient un terminal
aller à *Fin*
Sinon, *N* contient une fonction selon *TF*
Pour *i* = *1* jusqu'au nombre d'arguments de la fonction contenue dans *N*
Si (méthode de croissance) *et* (tirage au sort obtient un terminal)
N contient un terminal
aller à *Terminer*
Sinon
Appel récursif vers *Retour* avec :
TF = type d'argument de l'argument *i* du contenu de *N*
N = l'enfant *i* de *N* (la profondeur est ajustée)
Fin pour i
(emplacement *Terminer*)
Fin de la création

Si les FDA sont permises, il convient de les créer au même moment que les programmes principaux. La FDA est créée avec la même méthode que le programme principal auquel elle est associée. Quelques règles additionnelles, dont les paramètres sont choisis par l'usager, contraignent la création des FDA. Ces paramètres sont :

1. Le nombre de FDA par programme principal;
2. La profondeur maximale des FDA. Elle peut différer de celle des programmes principaux;
3. Le nombre minimal d'arguments pour chaque FDA;
4. Le nombre maximal d'arguments pour chaque FDA.

Par défaut, notre algorithme n'utilise pas les FDA. Si l'usager choisit de les utiliser, il devra préciser la valeur des quatre paramètres. Pour chaque FDA, le nombre d'arguments est choisi au hasard entre les nombres minimal et maximal choisis par l'usager. Tel que mentionné en 3.6.3, les FDA ont un ensemble de terminaux composés d'arguments (et non de caractéristiques) et de constantes.

3.6.5 Fonctions d'adéquation

L'adéquation (connue dans la littérature anglaise comme « fitness ») est la mesure de la performance d'un programme dans la prédiction des valeurs de sorties à partir des valeurs d'entrées. Il s'agit donc d'un indice de pertinence du programme pour la résolution du problème représenté par la base de données d'apprentissage. L'adéquation est une valeur numérique, ce qui nous permet de comparer les performances des programmes. C'est d'ailleurs ce critère que nous utiliserons pour sélectionner les programmes parmi la population (*Voir* section 3.6.6) afin de les transformer (*Voir* section 3.6.7).

La fonction d'adéquation est la suite de calculs, à partir de la base de données d'apprentissage, qui permet d'obtenir l'adéquation d'un programme. Cette fonction compare la valeur de prédiction désirée, fournie dans la base de données d'apprentissage, et la prédiction du programme.

Dans le cas de problèmes dont les valeurs de sorties sont continues, deux fonctions d'adéquation sont proposées :

1. Somme de l'erreur : l'adéquation du programme correspond à la somme des erreurs de prédiction. Une erreur de prédiction est la valeur absolue de la différence entre la valeur prédite et la valeur désirée;

2. Erreur quadratique: l'adéquation du programme correspond à la somme des erreurs de prédiction élevées au carré. Cette fonction d'adéquation pénalise grandement un programme qui fait quelques prédictions très inexactes, par rapport à un autre qui fait un grand nombre d'erreurs de prédictions plus justes.

Dans ces deux cas, l'adéquation représente une mesure de l'erreur. On préférera donc un programme qui a une valeur d'adéquation plus petite à un autre qui aurait une valeur

d'adéquation plus grande. Une performance parfaite obtiendra la note de 0 (puisqu'elle correspond à une erreur nulle).

Dans le cas d'un problème de classification (pour lequel les sorties sont une valeur d'énumération), la fonction d'adéquation dépend de l'approche utilisée au niveau de la combinaison des classificateurs (*Voir* section 3.5.1).

1. Pour une approche à classificateur unique, l'adéquation correspond simplement au nombre de bonnes prédictions du programme. Cette valeur peut être normalisée (entre 0.0 et 1.0) en divisant l'adéquation par le nombre d'échantillons présents dans la base de données d'entraînement.

2. Dans le cas d'une approche à un classificateur par classe d'échantillons, le classificateur fait la reconnaissance d'une classe en particulier. Le calcul de l'adéquation dépend du type de classificateur choisi :

    a. Classificateur binaire : comme pour l'approche à classificateur unique;

    b. Classificateur à sortie continue : la sortie du programme P est une valeur de confiance limitée entre -1.0 et 1.0. L'adéquation est calculée à partir de la somme S des valeurs de confiance de P pour chaque échantillon i, en fonction de la classe C fournie par la base de données d'apprentissage.

    $$S = \sum_i P(i) \cdot C(i) \tag{3.2}$$

    $C(i)$ est 1.0 si l'échantillon i fait partie de la classe reconnue et -1.0 dans le cas contraire. Finalement, l'adéquation correspond à la somme S des valeurs, normalisée entre 0.0 et 1.0.

    c. Classificateur à sortie continue pour algorithme de boosting intégré : la technique est sensiblement la même qu'en b, mais le poids W des échantillons d'entraînement est pris en compte :

$$S = \sum_{i} P(i) \cdot W(i) \cdot C(i) \tag{3.3}$$

Comme le poids des échantillons est lui aussi normalisé (total des poids est 1.0), la somme S pourra être normalisée de la même façon qu'en b.

Donc, pour un problème de classification, plus l'adéquation est grande, plus le programme est performant. Un taux de prédiction parfait est obtenu lorsque l'adéquation est 1.0.

Notre algorithme offre les fonctions d'adéquation de somme des erreurs et somme des erreurs quadratiques pour un problème continu, et les fonctions à sortie continue pour la PG avec orchestration et à sortie continue pour algorithme de boosting intégré.

Les fonctions d'adéquation que nous proposons sont générales, pour les types de problèmes les plus souvent rencontrés. Pour des problèmes plus spécifiques, comme par exemple celui de la fourmi artificielle (Koza, 1992), une fonction d'adéquation dédiée peut être programmée. Notre module permet à un programmeur d'intégrer facilement une nouvelle fonction d'adéquation.

Notez que, si la validation est utilisée, une valeur d'adéquation de validation est calculée à partir de la base de données prévue à cet effet. Cette valeur est prise en considération pour évaluer le critère d'arrêt, mais pas pour la sélection des programmes (seule l'adéquation d'entraînement compte dans ce cas).

3.6.6 Méthodes de sélection

La PG s'inspire de l'évolution des être vivants, en particulier du principe de la sélection naturelle. Elle utilise un mécanisme qui reproduit artificiellement ce principe en sélectionnant, selon le critère d'adéquation, les individus qui se reproduiront dans la prochaine génération. Dans cette perspective, les programmes dont l'adéquation est supérieure ont plus de chance d'être sélectionnés pour la reproduction et c'est ainsi qu'ils

deviennent de plus en plus performants pour la tâche de prédiction. Nous présentons quatre méthodes de sélection (Banzhaf et al., 1998), toutes disponibles pour notre algorithme.

1. **Roulette** : cette méthode sélectionne les reproducteurs selon l'adéquation d'un programme par rapport à la somme de l'adéquation de tous les programmes de la population. Elle est désignée par l'appellation anglaise Fitness-Proportionnal Selection.

$$P(i) = \frac{A(i)}{\sum_{i} A(i)} \tag{3.4}$$

   A(i) est l'adéquation du programme *i* alors que *P(i)* représente la probabilité de le sélectionner pour la reproduction.

2. **Sélection par Rang** : elle est basée sur l'ordre d'adéquation des *N* programmes de la population. Une fois les programmes triés en ordre, une fonction permet de calculer la probabilité de choisir chacun d'eux.

a. **Rang linéaire** : pour ce type de sélection, l'usager doit choisir les valeurs des constantes P_- et P_+

$$\text{probabilité minimale} = \frac{P_-}{N} \tag{3.5}$$

$$\text{probabilité maximale} = \frac{P_+}{N} \tag{3.6}$$

en respectant la contrainte suivante :

$$P_- + P_+ = 2 \tag{3.7}$$

Finalement. La probabilité de sélectionner le programme de rang i est obtenue par la fonction suivante :

$$P(i) = \frac{1}{N}\left(P_- + (P_+ - P_-)\cdot\left(\frac{i-1}{N-1}\right)\right) \tag{3.8}$$

b. **Rang exponentiel** : dans ce cas, l'usager doit choisir la valeur de la constante c selon la contrainte $0.0 < c < 1.0$. Plus c est élevée, plus favorisée sera la sélection des meilleurs programmes. Voici la fonction de calcul de la probabilité de sélection :

$$P(i) = e^{-c\cdot i} \tag{3.9}$$

3. **Tournoi** : Il est particulièrement adapté à un scénario évolutif continuel (section 3.6.8). Tous les individus ont la même chance d'être sélectionnés pour le tournoi. En premier lieu, un nombre (défini par l'usager) de programmes sont sélectionnés dans la population. Ensuite, les programmes choisis sont triés par ordre d'adéquation. Les meilleurs sont appelés gagnants et les plus faibles perdants. Finalement, l'opération génétique est appliquée sur les gagnants du tournoi, et, si le scénario évolutif est continuel (*Voir* section 3.6.8), les programmes produits par l'opération remplacent les perdants du tournoi. Le nombre de programmes retenus et le nombre de perdants du tournoi sont respectivement déterminés par le nombre de parents et d'enfants attribués aux opérateurs génétiques (*Voir* le tableau 3.4 de la section 3.6.7).

3.6.7 Opérations génétiques

Les programmes créés initialement ont généralement une adéquation très faible. Les opérateurs génétiques procèdent à des transformations sur la population initiale et réalisent ainsi une forme d'évolution. Ces mécanismes servent donc à la recherche d'une solution plus performante. Trois opérateurs sont couramment utilisés en PG : le croisement, la mutation et la reproduction. Nous les utiliserons pour notre recherche, en y ajoutant une variante de mutation et l'insertion d'un tout nouveau programme. Il est important de noter que, lors de la transformation d'un programme par un opérateur, la règle de profondeur maximale (*Voir* section 3.6.4) doit être respectée.

1. **Croisement** : Le croisement combine le matériel génétique de deux programmes parents en échangeant une portion du code d'un parent avec une portion du code d'un autre parent. Pour un programme en arbre, il s'agit d'échanger les sous arbres sélectionnés aléatoirement à partir de deux programmes. Les résultats du croisement sont deux programmes enfants. La figure 3.8 illustre le croisement de deux programmes en arbre.

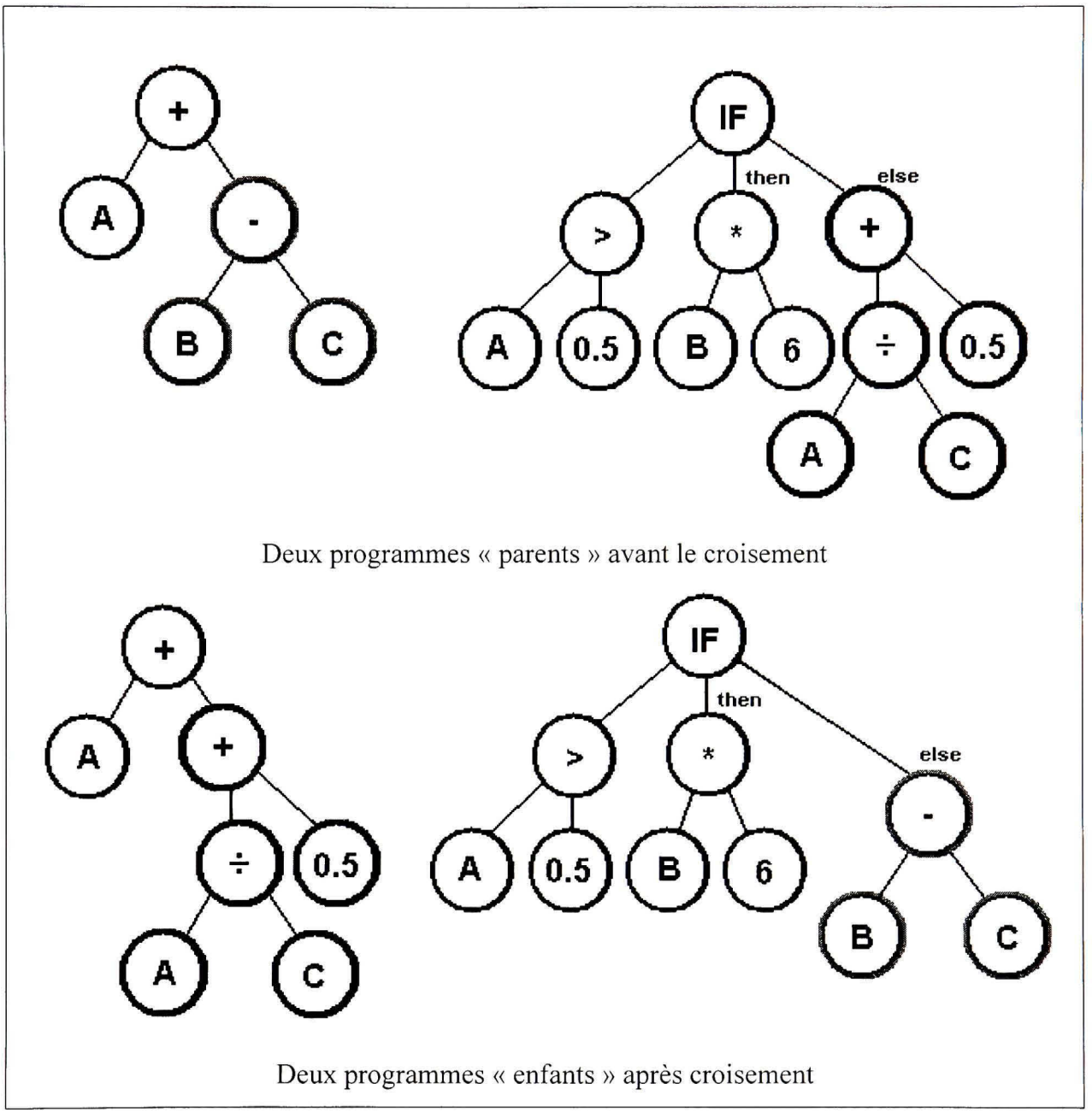
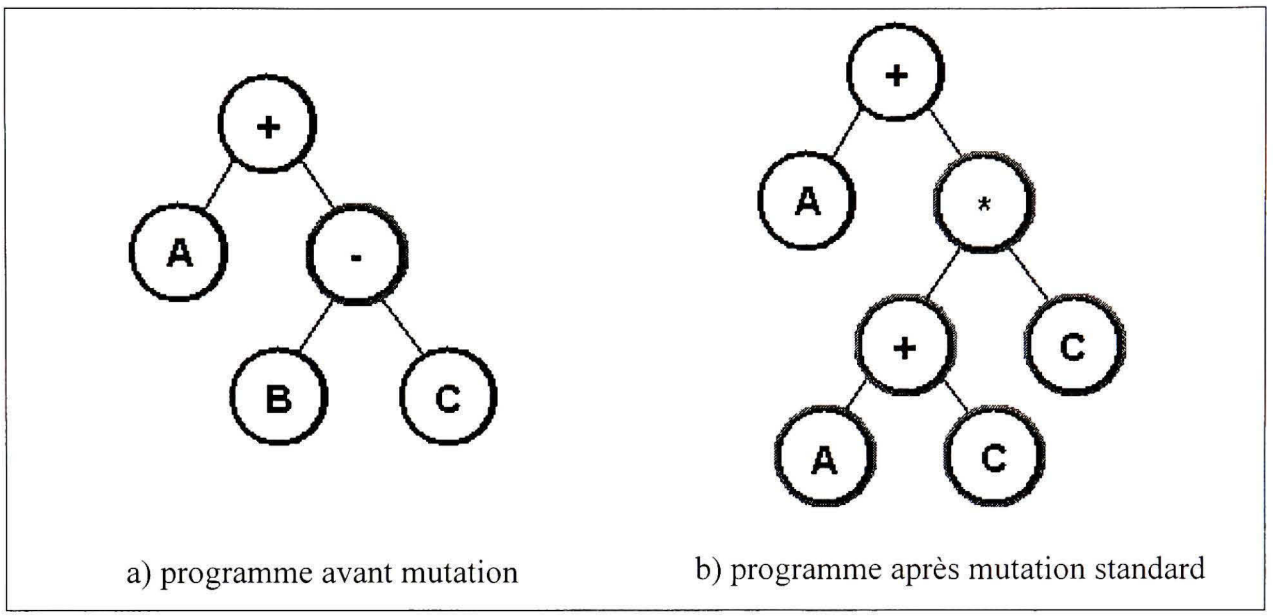


Figure 3.8 ***Croisement génétique entre deux programmes en arbre.***

(Tiré de Levasseur, 2005)

2. **Mutation (standard)** : La mutation remplace une section de programme par une nouvelle section. Pour un programme en arbre, un nœud est choisi aléatoirement, puis

remplacé par un sous arbre généré par la méthode de création choisie. Nous utiliserons par défaut la méthode complète (section 3.6.4).

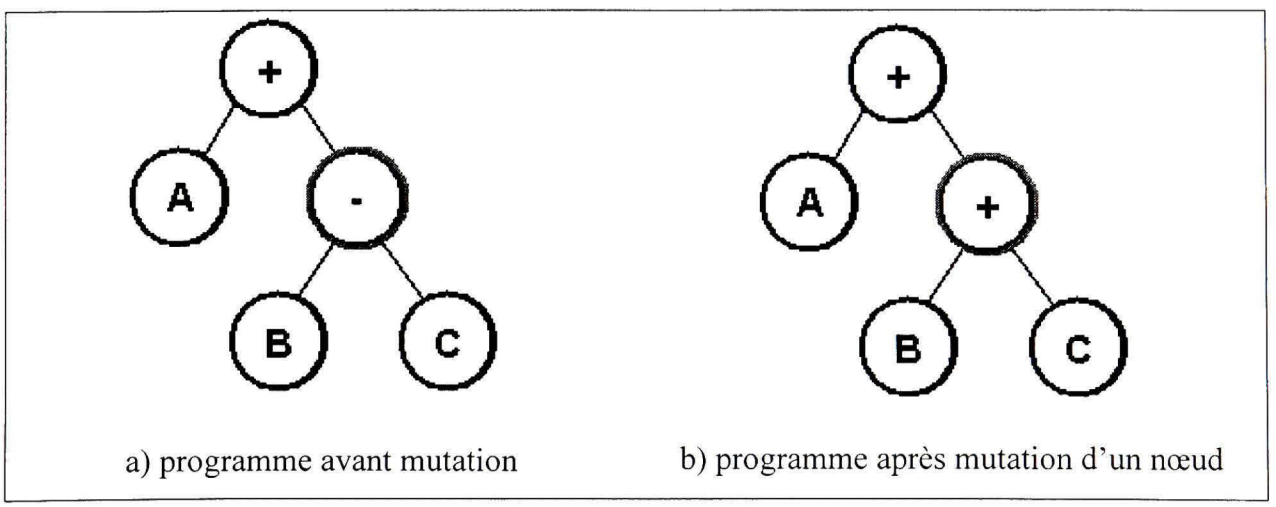


a) programme avant mutation b) programme après mutation standard

Figure 3.9 ***Mutation génétique standard d'un programme en arbre.***

(Tiré de Levasseur, 2005)

3. **Mutation d'un nœud** : Cette opération consiste à transformer le contenu d'un nœud choisi aléatoirement dans un programme en arbre. Pour que le programme reste conforme à la grammaire, le nouveau contenu doit avoir le même nombre d'arguments que l'ancien contenu.



a) programme avant mutation b) programme après mutation d'un nœud

Figure 3.10 ***Mutation génétique d'un nœud d'un programme en arbre.***

(Tiré de Levasseur, 2005)

4. **Reproduction** : la reproduction consiste à faire une copie d'un programme. Cette opération, jumelée avec une méthode de sélection qui favorise les programmes qui présentent une adéquation plus forte, est une façon de conserver les portions de code pertinent, donc de réaliser l'élitisme.

5. **Nouveau programme** : cette opération ne fait qu'insérer un nouveau programme, construit selon la méthode de création choisie. Par défaut, nous utiliserons la méthode de croissance. L'intérêt de cet opérateur est de maintenir la diversité génétique d'une population au cours de l'évolution.

Il existe de nombreuses méthodes, comme le croisement sensible au contexte (D'haeseleer, 1994) et le croisement « intelligent » (Teller et Veloso, 1997), pour tenter de contrer les effets pervers des opérations génétiques. En effet, puisque les opérateurs procèdent par choix aléatoires, il y a de fortes chances que les programmes produits par ces opérateurs soient peu performants (Nordin, Francone et Banzhaf, 1996). Afin de permettre une grande latitude quant à la sélection d'enfants performants, notre algorithme laisse le loisir à l'usager de choisir le nombre de parents et d'enfants pour chaque opérateur. En choisissant un plus grand nombre d'enfants pour une opération, il est possible de créer l'effet de « brood recombination » (Tackett et Carmi, 1994) et de sélectionner les meilleurs enfants. Le tableau suivant explique l'influence du nombre de parents et d'enfants sur nos opérateurs :

Tableau 3.4

Influence du nombre de parents et d'enfants sur les opérateurs

Opérateur génétique	**P**	**E**	**Influence du nombre de parents et d'enfants**
Croisement	2	2	Les deux premiers parents sélectionnés sont croisés autant de fois que nécessaire pour produire le nombre d'enfants voulu.
Mutation et mutation d'un nœud	1	1	Le premier parent subit la mutation autant de fois que le nombre d'enfants attendu.
Reproduction	1	1	Le premier parent est reproduit autant de fois que le nombre d'enfants désiré.
Nouveau programme	-	1	Le nombre d'enfants indique le nombre de nouveaux programmes créés

Les valeurs sous les colonnes *P* et *E* du tableau *343* indiquent respectivement le nombre de parents et d'enfants par défaut pour les opérateurs génétiques.

Dans le cas d'une sélection par tournoi, le nombre de parents détermine le nombre de programmes sélectionnés pour le tournoi (dans ce cas, il doit y avoir au moins autant de parents qu'il y a d'enfants).

Finalement, lorsqu'une opération génétique transforme un programme, la même opération génétique est utilisée pour transformer ses FDA, si elle en possède. Les mécanismes utilisés par les opérateurs sont exactement les mêmes pour les programmes principaux que pour les FDA.

3.6.8 Scénarios évolutifs

Un fois la qualité d'un programme évaluée à l'aide de l'adéquation, il convient de décider si l'individu sera reproduit, conservé ou éliminé. Il existe plusieurs scénarios qui réalisent les étapes nécessaires à l'évolution des programmes. Chacun d'eux procèdent aux tâches suivantes : création des programmes, sélection, reproduction et évaluation de l'adéquation dans des cycles différents. Nous présentons trois scénarios couramment utilisés en PG.

1. Le premier s'inspire du déroulement de l'évolution des algorithmes génétiques. Nous l'appelons « sélection avant reproduction » et l'illustrons dans le schéma qui suit :

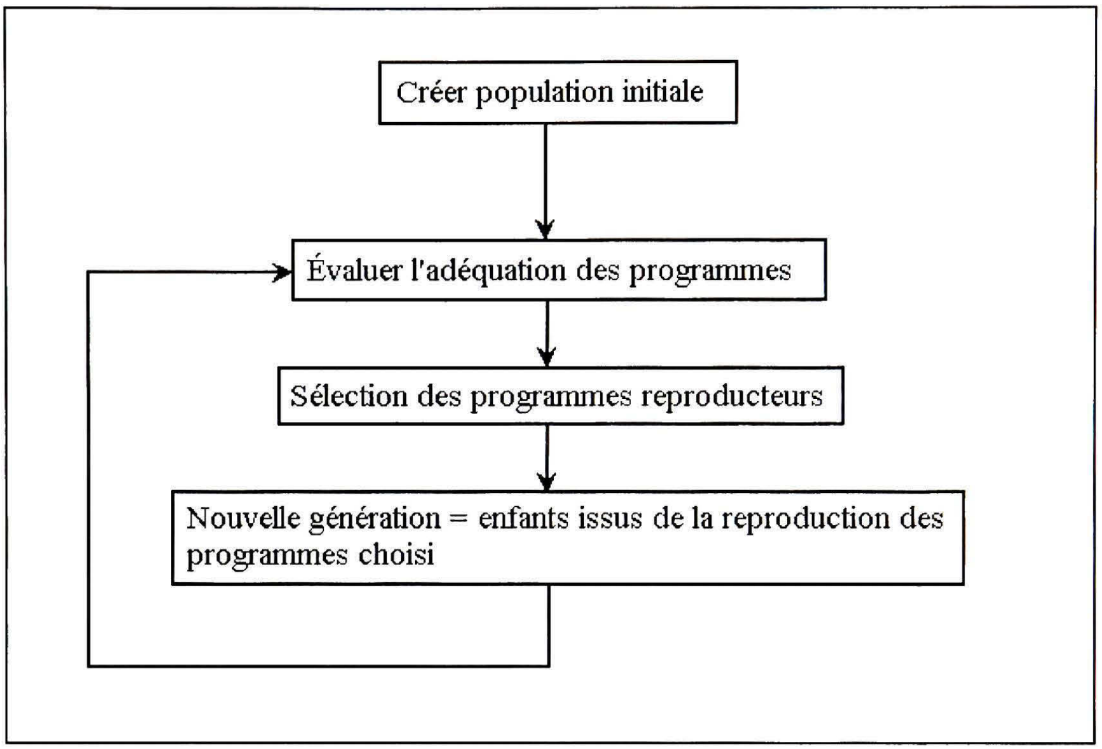


Figure 3.11 ***Scénario évolutif sélection avant reproduction***

Les programmes enfants, obtenus des opérations génétiques, remplaceront complètement, dans la génération suivante, les programmes parents à partir desquels ils ont été produits. Il est donc possible que la nouvelle génération contienne des programmes dont l'adéquation est plus faible que la génération précédente. Avec ce scénario, il est suggéré d'utiliser des mécanismes d'élitisme (*Voir* section 3.6.9) pour prévenir la dégénérescence des programmes au fil de l'évolution.

2. Le deuxième s'inspire du domaine des stratégies évolutives. Le schéma suivant montre les étapes de fonctionnement de ce scénario appelé « sélection après reproduction » :

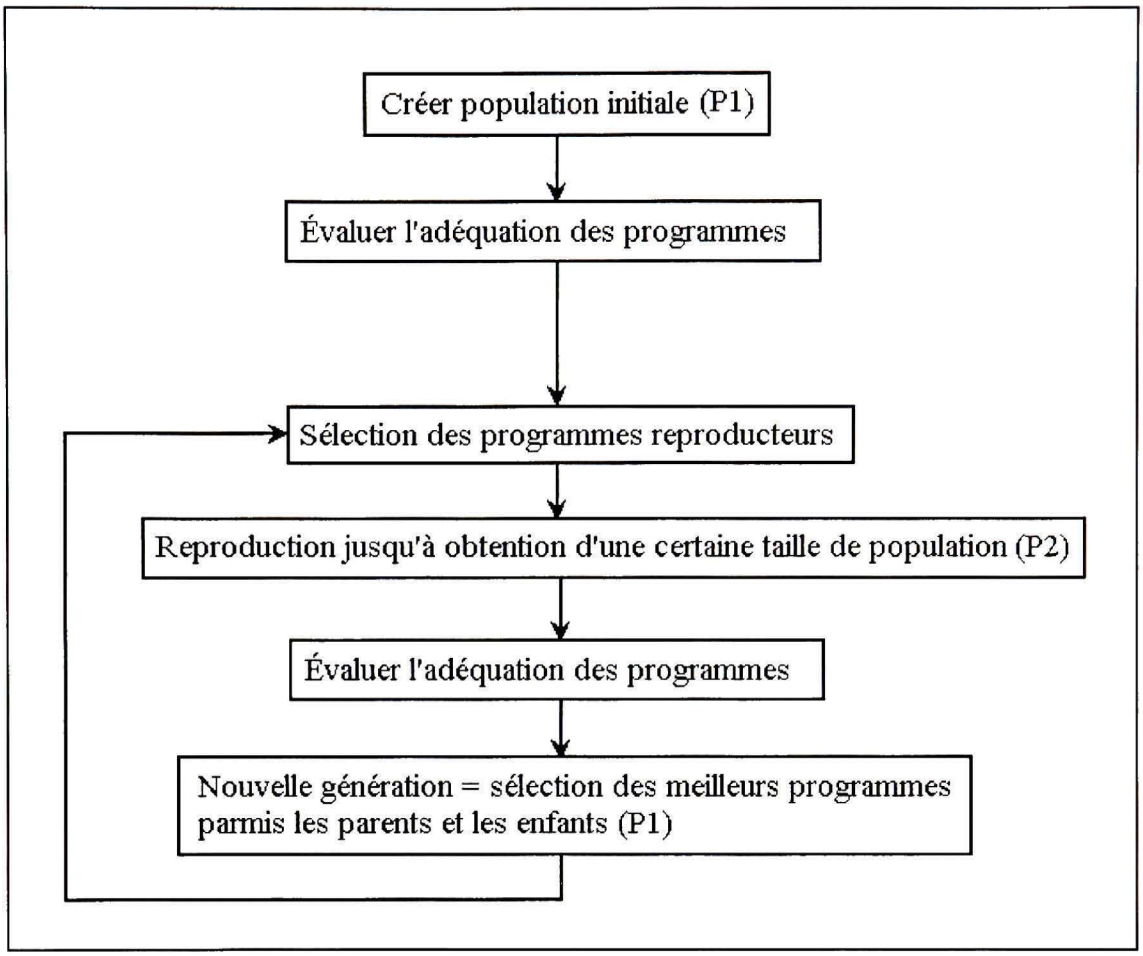


Figure 3.12 ***Scénario évolutif sélection après reproduction***

Deux distinctions caractérisent la sélection après reproduction. Premièrement, il est possible de choisir un nombre de programmes à produire *P2*, lors de la phase de reproduction (ce nombre peut être différent du nombre de programmes à créer initialement, *P1*). Deuxièmement, ce processus boucle après la sélection des *P1* programmes ayant les meilleures adéquations à partir des populations de parents et d'enfants réunies. Ce mécanisme évite la dégénérescence des programmes puisqu'il n'élimine que les programmes les plus faibles.

3. Le dernier scénario est dit « continuel ». Il n'offrira un bon fonctionnement que s'il est combiné avec la méthode de sélection par tournoi :

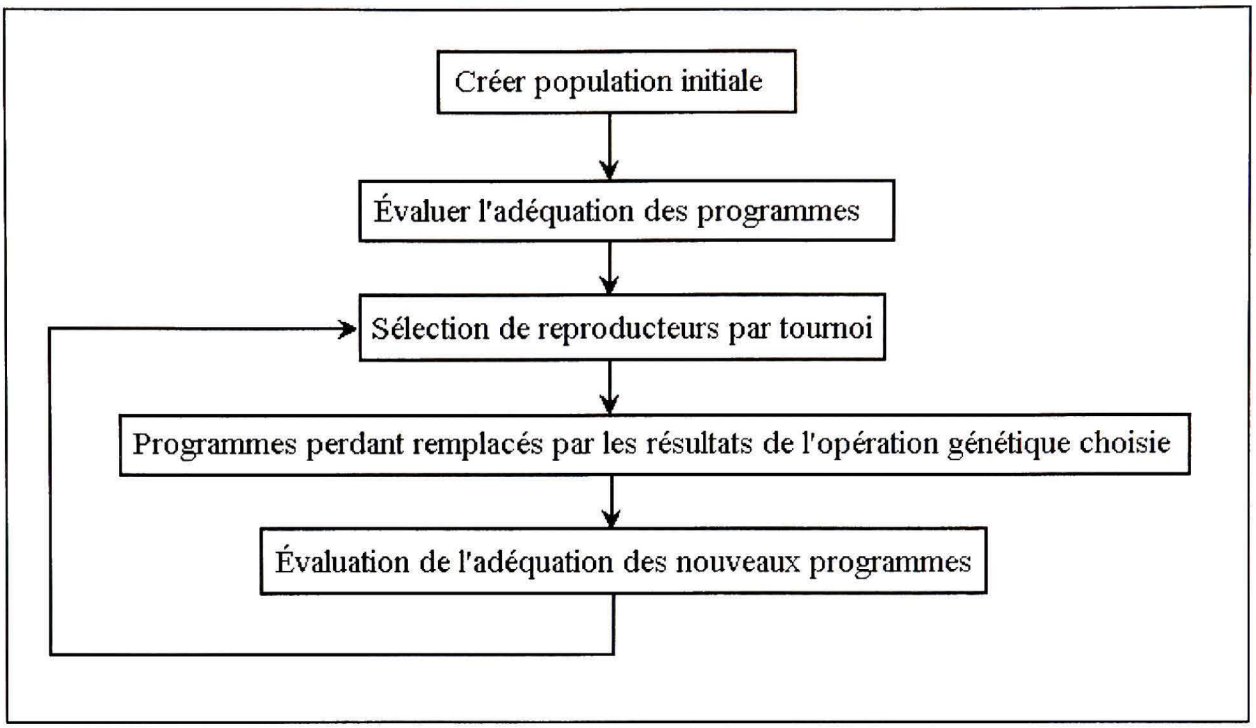


Figure 3.13 *Scénario évolutif continuel*

Le scénario continuel n'est pas basé sur le principe de générations successives. Il continue de produire des programmes enfants à partir de parents sélectionnés par tournoi, tant qu'il n'a pas atteint les critères d'arrêt. La population reste constante car les enfants produits remplacent les perdants du tournoi (voir section 3.6.6). Ce type de sélection permet d'éviter la dégénérescence des meilleures solutions puisqu'elles ne seront jamais perdantes de tournoi, donc jamais remplacées. Le scénario continuel a l'avantage de permettre l'implémentation de processus en parallèle et de fournir des résultats comparables à ceux obtenus par d'autres scénarios. Pour une évolution continuelle, nous définissons une génération comme le moment où le nombre initial de programmes a été produit (le décompte recommençant à zéro à chaque nouvelle génération).

3.6.9 Élitisme

L'élitisme est généralement défini comme le degré avec lequel les constituants des meilleurs programmes sont reproduits dans la population. Il peut être nécessaire pour conserver les portions de code qui permettent l'évolution. Par contre, l'élitisme peut se révéler néfaste à l'évolution lorsqu'il est trop fort : il reproduit trop souvent les mêmes constituants de programmes et ralentit l'évolution des programmes puisque ceux-ci ne peuvent découvrir de nouvelles combinaisons.

Dans la PG, il est possible de contrôler le niveau d'élitisme au moyen de l'opérateur génétique de reproduction, des méthodes de sélection et des scénarios « sélection après reproduction » et « continuel ».

Dans tous les cas, il est bénéfique de conserver les meilleurs programmes obtenus au cours de l'évolution, même s'ils sont éliminés par le processus évolutif. Nous appelons cette pratique l'élitisme artificiel.

Pour notre module, un gestionnaire d'élites conserve les meilleurs programmes au long de l'évolution, réalisant l'élitisme artificiel. L'usager peut choisir le nombre d'élites. Dans le cas d'un problème de régression en valeur continue, le nombre d'élites correspond au nombre de programmes conservés en mémoire, mais seul le meilleur programme sera utilisé pour la prédiction. Dans le cas de la classification, le nombre d'élites correspondra au nombre de programmes par classe qui seront conservés et utilisés.

Finalement, il est parfois pratique de « simplifier » les programmes pour éliminer les parties inutiles (toujours égales à une constante) et permettre de meilleures visualisation et compréhension. Ceci peut être réalisé automatiquement en vérifiant, pour chaque sous arbre, si les valeurs obtenues des fonctions varient lorsque le programme est testé sur la base de données d'apprentissage. Le gestionnaire d'élite peut simplifier les meilleurs programmes si

ce comportement est désiré. Notre module procède systématiquement à la simplification des programmes obtenus à la toute fin d'une évolution.

3.7 Intégration à Weka

L'appel de l'algorithme de PG se fait à l'aide de la classe suivante à partir de Weka :

- *weka.classifiers.functions.GeneticProgramming*

Le tableau 3.4 illustre les valeurs proposées par défaut pour la PG :

Tableau 3.5

Paramètres par défaut de l'algorithme de programmation génétique

Paramètre	**Valeur par défaut**
Proportion de la base d'entraînement réservé pour validation	0.5 (50%)
Prétraitement des données	Normalisation statistique
Taille de la population	100 programmes
Profondeur maximale	5 niveaux
Critères d'arrêt	Adéquation = 0.9, Générations = 20, Temps = 0.33 minute
Fonctions	+, -, /, *, If, >, <, Pow, &, |, Max, Min, Exp, Log (*Voir* tableau 3.2)
Fonctions définies automatiquement	Non utilisées
Création de la population	Méthode de rampe moitié-moitié
Sélection des programmes	Proportionnel à l'adéquation (roulette)
Proportions des opérateurs génétiques	Croisement = 0.9, Mutation =0.07, Nouveau programme = 0.03
Nombre de parents et d'enfants	Par défaut selon l'opérateur (tableau 3.3)
Scénario évolutif	Sélection après reproduction, taille de la nouvelle population = 100
Gestionnaire d'élite	Conserver les meilleurs programmes en mémoire
Taille de l'élite	5 programmes

Ces mêmes paramètres sont présentés dans la boîte de réglage des options illustrée à la figure 3.4 de la section 3.4.

Notre projet, comme Weka, utilise la licence publique générale GNU. Il est possible de télécharger le code source et consulter la page de documentation par internet (Levasseur, 2008d).

CHAPITRE 4

PROTOCOLE EXPÉRIMENTAL

Notre objectif est d'évaluer si la classification d'objets dans des images peut être réalisée automatiquement, sans l'aide d'un expert, et offrir une performance de prédiction suffisante pour des applications réelles. De plus, les expériences serviront à mesurer et comparer la performance de différents algorithmes de classification, suite à l'extraction de caractéristiques générales à partir d'images.

Ce chapitre présente les outils de développement et les méthodes utilisées pour l'acquisition des données, les étapes de traitement préalables à la classification, l'apprentissage et la classification, puis la comparaison des résultats.

4.1 Outils de développement

Les principaux critères pour la sélection de nos outils de développement sont : performance de l'outil pour la tâche désignée, facilité d'utilisation et reproductibilité. Ce dernier critère nous encourage à préférer les logiciels les plus utilisés (par exemple le logiciel de développement Matlab, couramment utilisé en recherche et développement) ou les plus accessibles, comme les logiciels libres. Ces derniers ont une licence dite libre qui donne à chacun le droit de les utiliser, de les étudier ou de les modifier. Ils peuvent être téléchargés gratuitement sur internet. Le tableau 4.1 illustre le choix du logiciel pour chaque section de notre travail :

Tableau 4.1

Choix des logiciels

Usage	**Logiciel**	**Qualités recherchées**
Segmentation et extraction de caractéristiques	Matlab (Eaton, 2004)	Rapidité de développement, simplicité, boîte à outil pour la vision par ordinateur
Segmentation par un opérateur humain	Gimp (Natterer et Neumann, 2007)	Logiciel libre, simplicité d'utilisation
Réduction de la dimensionnalité et classification	Weka	Logiciel libre, visualisation conviviale, modularité, beaucoup d'algorithmes disponibles
Comparaison des résultats	Matlab	Simplicité, bons outils de visualisation
Développement du module de PG en Java	Eclipse (Eclipse Foundation, 2006)	Logiciel libre, modularité, rapidité

Essentiellement, nos travaux pourront être reproduits par d'autres avec Matlab et Weka. Le code utilisé pour les expérimentations que nous présentons peut être téléchargé sur internet en visitant notre site personnel (Levasseur, 2008c).

4.2 Données de l'étude

Les bases de données pour nos expériences ont été choisies ou produites selon deux critères. Premièrement, nous voulons des images d'objets variables, qui sont considérablement difficiles à classifier pour un humain. Deuxièmement, nous voulons que la plupart des bases de données soient reliées au domaine agro-alimentaire, puis qu'une partie provienne d'un autre domaine. Finalement, l'accessibilité des bases de données nous a limité dans nos choix.

Nous utiliserons pour nos expérimentations huit bases de données, dont quatre ont été entièrement préparées par nous. Pour six de ces bases, la segmentation sera réalisée manuellement à l'aide d'une procédure adaptée à l'environnement particulier de la prise d'image (section 4.3), qui comprend un fond facilement identifiable. Finalement, pour les huit bases de données, des caractéristiques (présentées à la section 4.4) seront extraites des

images segmentées ou non. Voici un tableau qui résume les informations concernant nos données :

Tableau 4.2

Base de données utilisées

No	Base de données	C	Images	Total	Seg.	Propriété particulière
1a	Céréales (grande)	6	950	5700	Manuelle	Quatre des six classes peuvent être facilement confondues
1b	Céréales (petite)	6	500	3000	Manuelle	Même que *1a* mais réduite pour des questions de temps de calcul. Les résultats restent significatifs
2	Raisins secs	3	450	1350	Manuelle	Classes faciles à différencier
3	Grains de pollen	7	196	1372	Manuelle	Cinq classes sur sept peuvent être confondues, images en niveau de gris
4	Nœuds de bois	6	27	162	Manuelle	Problème difficile du fait qu'il y a peu d'échantillons par classe
5a	Feuilles	3	60	180	Aucune	L'objet à classifier représente environ 15% de l'image
5b	Feuilles (rognées)	3	60	180	Partielle	L'objet à classifier représente environ 40% de l'image
6	Chiffres par ordinateur	10	33	330	Manuelle	Seule base de données d'images contenant des objets non biologiques

Dans le tableau 4.1, *C* correspond au nombre de classes, *Images* au nombre d'images par classe, *Total* au nombre total d'images pour la base de données et *Seg.* au type de segmentation utilisé.

Voici une description détaillée de chacune des bases de données présentées au tableau 4.1 :

1. La base de données d'images de céréales a été élaborée par Yan Levasseur et Brice Bourgouin, et a déjà servi aux expérimentations de ce dernier dans le cadre d'un mémoire (Bourgouin, 2007). Elle contient des images de six classes de céréales. Toutes les images ont été prises sur fond noir uniforme. Les céréales peuvent avoir

n'importe quelle orientation dans l'image. Un grand nombre d'images par classe sont disponibles. Deux bases ont été créées à partir de l'ensemble des images.

a. Une première contient 950 images par classe. Elle est appelée grande base de céréales;

b. Une deuxième, jugée utile pour réduire le temps de calcul alloué à la phase d'apprentissage de certains classificateurs (en particulier la PG), contient 500 images par classe.

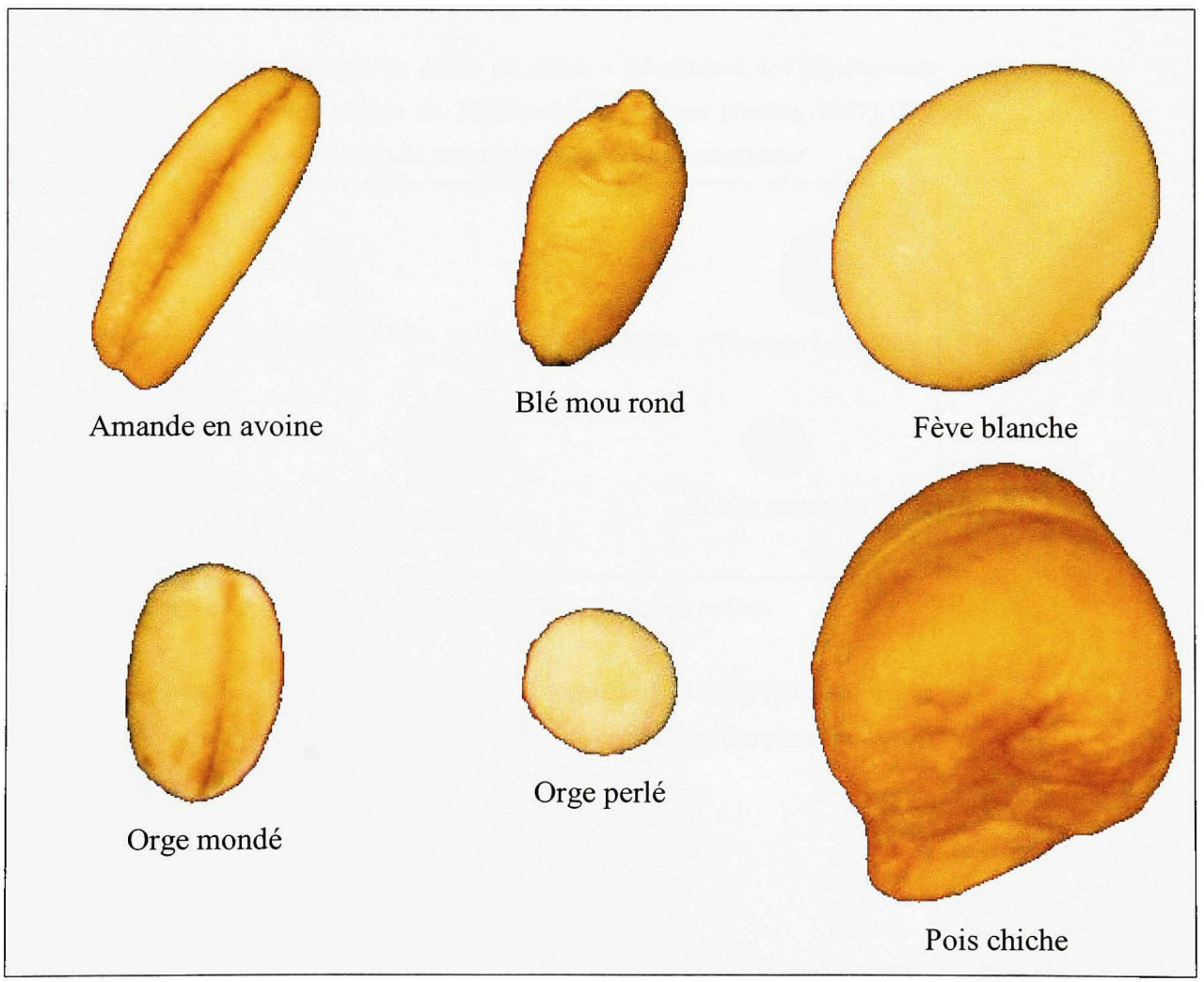


Figure 4.1 ***Images segmentées des six classes de céréales.***

2. Joël Bibeau et Yan Levasseur ont produit la base de données d'images de raisins secs (Bibeau, 2006). La taille des raisins et de légères différences de couleur permettent une bonne classification manuelle des trois classes représentées. Les images ont été prises sur un fond bleu uniforme.

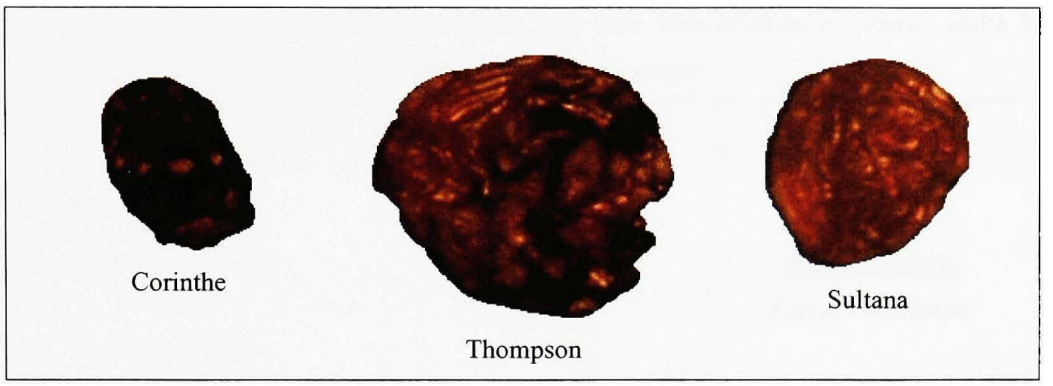


Figure 4.2 ***Images segmentées des trois classes de raisins secs.***

3. La base de données d'images de grains de pollen a été obtenue des départements d'informatique et d'électronique de l'Université de Bangor (France, 2007). Les images sont petites et en niveaux de gris, mais restent faciles à segmenter.

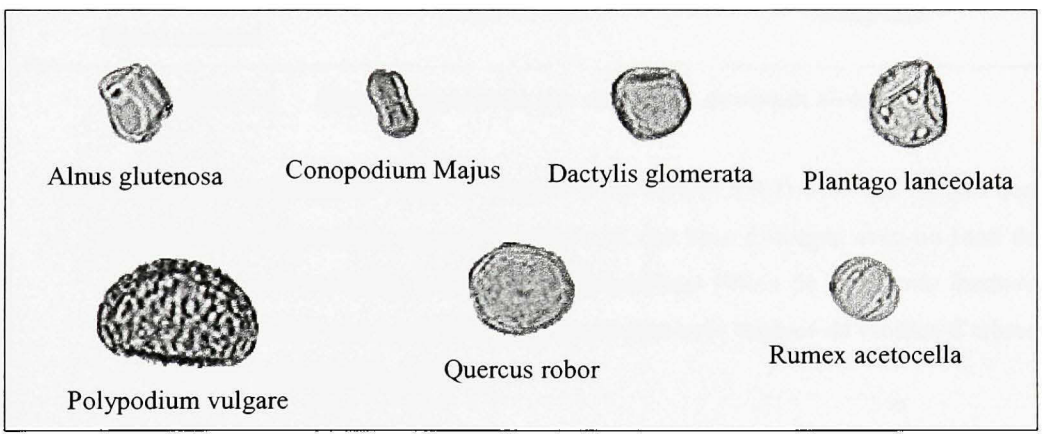


Figure 4.3 ***Images segmentées des sept classes de pollen.***

4. Le Laboratory of Wood Technology de l'Helsinki University of Technology (Finlande) partage une base de données d'image de nœuds dans des planches de bois

(Silven, Niskanen et Kauppinen, 2000). Cette base ne fournit que peu d'échantillons pour chaque classe de nœuds. Cette particularité permettra d'observer le pouvoir de généralisation des classificateurs avec une base d'apprentissage restreinte. Nous avons nous-mêmes segmenté les nœuds du fond lisse qui les entouraient. L'inexactitude de notre segmentation pour cette base de données pourra rendre la prédiction encore plus difficile pour nos classificateurs.

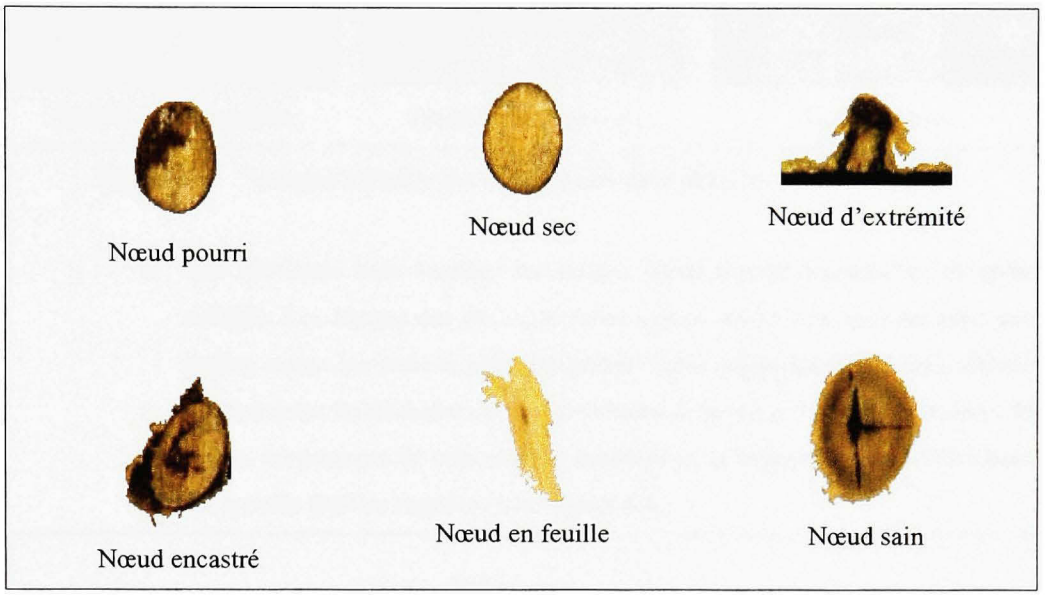


Figure 4.4 ***Images segmentées des six classes de nœuds de bois.***

5. Inspirés par le travail de Teller et Veloso (*Voir* section 2.8.1) avec des images non préalablement segmentées, nous avons cherché une base d'images avec un fond de couleur et texture variables. Le groupe *Computational Vision* du *California Institute of Technology* offre une base d'images comprenant trois espèces de feuilles d'arbres sur des fonds variés (Weber, 2007).

a. Une première base contient les images entières, toutes de même taille et non segmentées (*Voir* figure 4.5);

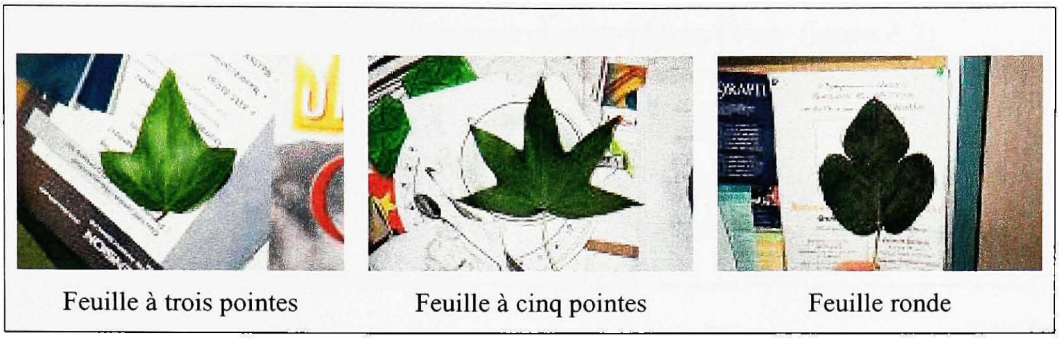


Figure 4.5 ***Images réduites des trois classes de feuilles avec fond variable.***

b. Une deuxième base contient les images partiellement segmentées, de taille variable. Les images ont été segmentées grossièrement une par une avec une fenêtre carrée qui cible la région d'intérêt. Cette segmentation partielle simule une précision facile à obtenir sur une chaîne de production sans algorithme de vision, simplement en changeant le montage pour la photographie. Cette base est appelée feuilles rognées (*Voir* figure 4.6).

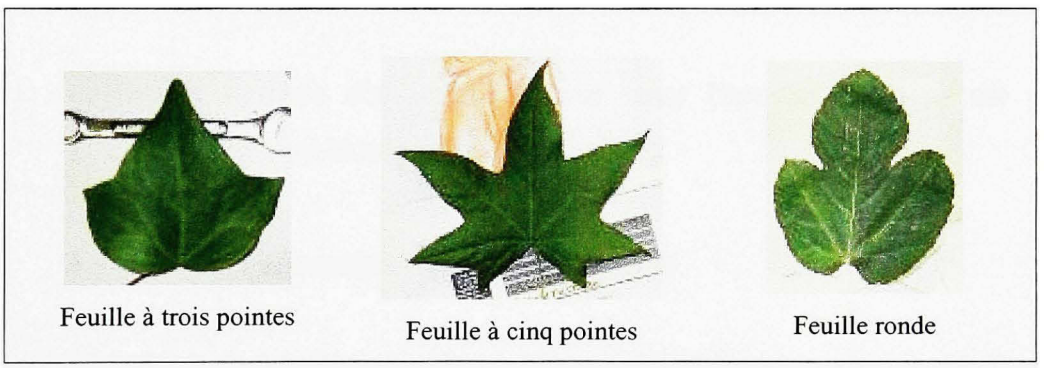


Figure 4.6 ***Images rognées réduites des trois classes de feuilles.***

6. La dernière base de données a été produite par Yan Levasseur. Il s'agit d'une base d'images de chiffres romains selon la police standard « times new roman ». 11 grandeurs de polices (de 12 à 32 points, par saut de 2 points) sont utilisées, ainsi que

les versions normale, italique et grasse de la police. La base de données contient donc 33 échantillons pour chaque classe (*Voir* figure 4.7).

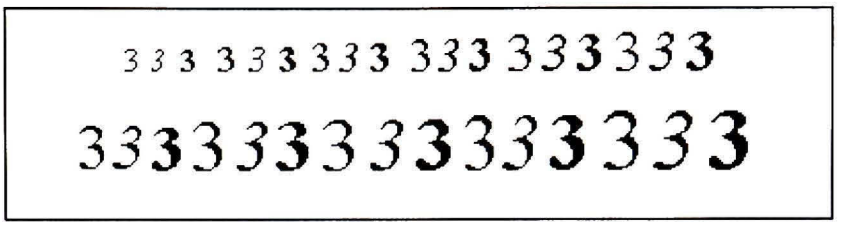


Figure 4.7 ***Image segmentée des 33 échantillons pour la classe « 3 ».***

Les chiffres de 0 à 9 sont utilisés, pour un total de 10 classes (*Voir* figure 4.8).

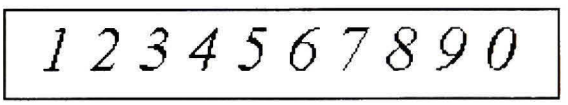


Figure 4.8 ***Images segmentées des dix classes de chiffres.***

Le nombre d'échantillon pour cette base de données est petit, en particulier considérant le nombre de classes.

4.3 Segmentation des objets d'intérêt

Pour les bases de données de céréales et de raisins, nous avons développé un script Matlab qui nous permet de segmenter les objets individuellement à partir d'une photo numérique qui comprend plusieurs objets. Cette technique permet un processus rapide, car une seule photo suffit à capturer un grand nombre d'objets. Une version du script de segmentation peut être téléchargée sur internet (Levasseur, 2008b). Voici une brève description de ce script.

Premièrement, nous devons départager les pixels du fond des objets. Dans le cas du script présenté, toutes les valeurs de pixels en dessous d'un certain seuil font partie du fond. Les pixels au-dessus font partie des objets. Cette opération est appelée seuillage; la valeur du seuil dépend des valeurs de pixels de l'image et peut être déterminée automatiquement au besoin à l'aide de fonctions de Matlab. Ensuite, chaque groupe de pixels disjoints est

considéré comme un objet distinct. Les objets trop petits, qui ne comptent que peu de pixels, sont éliminés alors que les autres sont conservés.

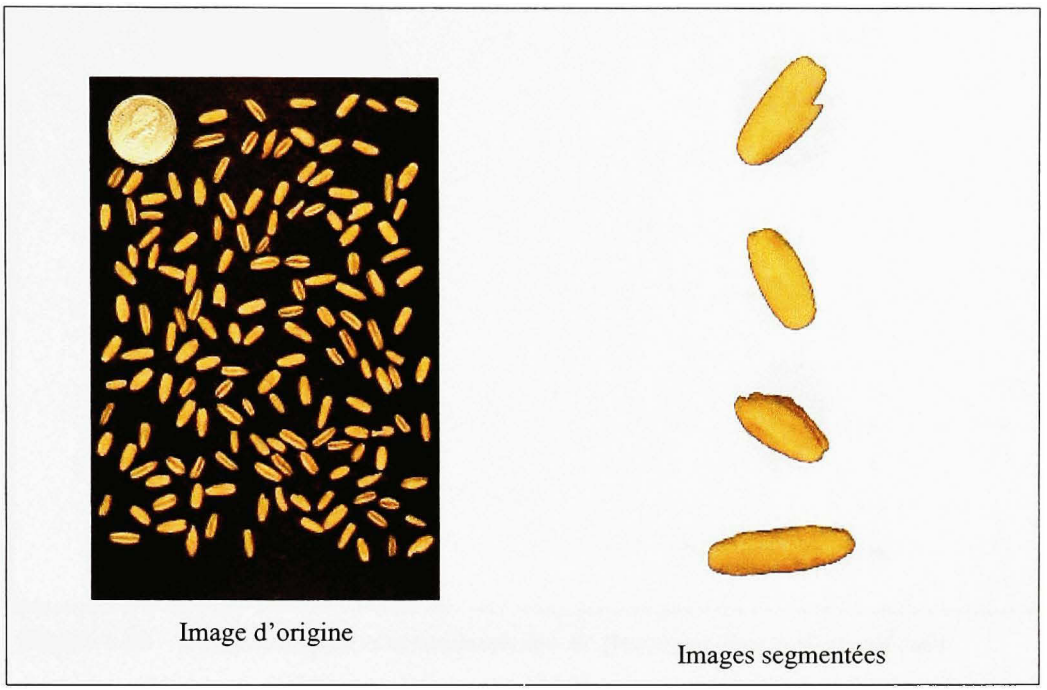


Figure 4.9 ***Image réduites de céréales et de quelques résultats de segmentation.***

Pour les bases de nœuds et de pollen, les images se présentent déjà de façon individuelle. La segmentation est réalisée de la même façon que pour les bases de céréales, à l'exception que seul l'objet comptant le plus de pixels est conservé à partir d'une image.

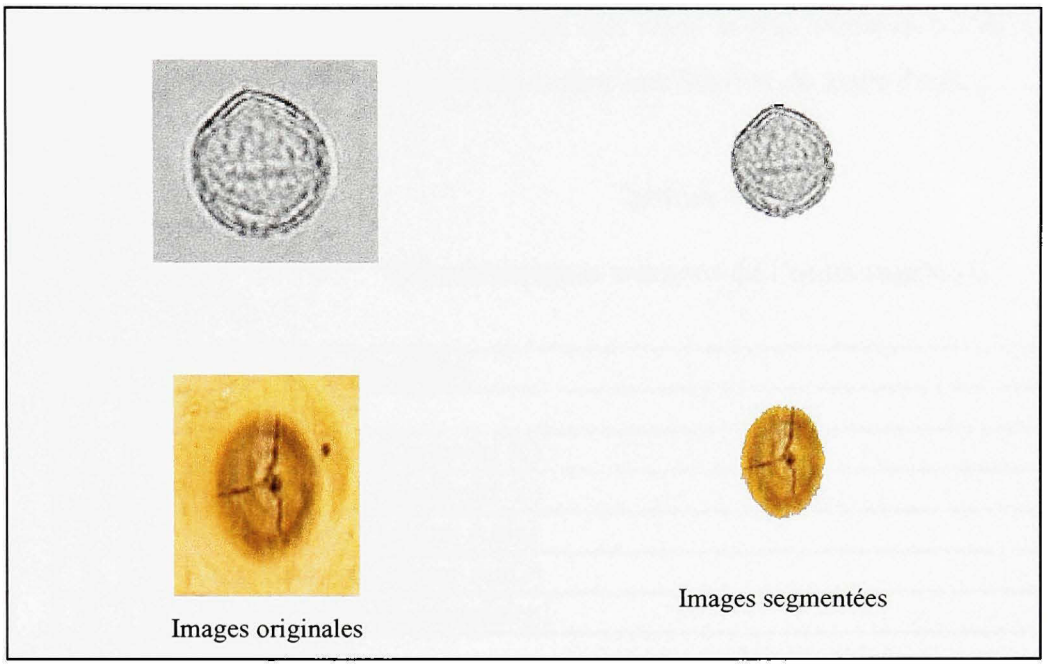


Figure 4.10 ***Images originales et segmentation de Quercus robor et de nœud sain.***

La base de données de feuilles n'a simplement pas été segmentée. La base de feuilles rognées a été segmentée par un opérateur humain. L'opérateur a sélectionné et sauvegardé une région d'intérêt carrée à partir de chaque photo de feuille (*Voir* figures 4.5 et 4.6).

4.4 Extraction des caractéristiques

Une fois les images préparées, il nous faut en extraire les informations qui nous permettront de classifier les objets qu'elles présentent. Notre choix de caractéristiques à extraire a partir des images tient compte de deux critères. Le premier est la facilité avec laquelle on peut utiliser ou programmer les algorithmes d'extraction pour ces caractéristiques avec Matlab. Le second critère concerne le nombre suffisant et la variété des caractéristiques. Nous désirons utiliser plusieurs caractéristiques différentes pour chaque type.

Voici un tableau des caractéristiques que nous avons retenues, parmi celles fournies par la littérature, pour traiter les problèmes de classification de notre étude :

Tableau 4.3

Caractéristiques extraites de l'objet segmenté

No	T	Nom (nom anglais)
1	M	Aire (Area)
2	M	Centroïde X (Centroid X)
3	M	Centroïde Y (Centroid Y)
4	M	Axe majeur (Major Axis)
5	M	Axe mineur (Minor Axis)
6	M	Excentricité (Eccentricity)
7	M	Orientation
8	M	Aire convexe (Convex Area)
9	M	Aire remplie (Filled Area)
10	M	Nombre d'Euler (Euler Number)
11	M	Diamètre équivalent (Equivalent Diameter)
12	M	Solidité (Solidity)
13	M	Étendue (Extent)
14	M	Périmètre (Perimeter)
15	M	Hauteur (Height)
16	M	Largeur (Width)
17	M	Complexité (Complexity)
18	T	Moyenne (Gray Mean)
19	T	Écart type (Standard Deviation)
20	T	Moment - valeur R
21	T	Asymétrie (Skewness)
22	T	Coefficient d'aplatissement (Kurtosis)
23	T	Uniformité (Uniformity)
24	T	Entropie du niveau de gris (Gray Entropy)
25	T	Contraste dans 3 directions : horizontal, vertical et diagonal (Contrast)
26	T	Corrélation dans 3 directions : horizontal, vertical et diagonal (Correlation)
27	T	Énergie dans 3 directions : horizontal, vertical et diagonal (Energy)
28	T	Homogénéité dans 3 directions : horizontal, vertical et diagonal (Homogeneity)
29	T	Moment R de la forme binaire locale (Local Binary Pattern Moment R)
30	T	Asymétrie de la forme binaire locale (Local Binary Pattern Skewness)
31	T	Coefficient d'aplatissement de la forme binaire locale (Local Binary Pattern Kurtosis)
32	T	Origine de la droite pour transformée de Fourier (Fourier Origin)

No	T	Nom (nom anglais)
33	T	Pente de la droite pour transformée de Fourier (Fourier Slope)
34	T	Corrélation de la droite pour transformée de Fourier (Fourier Correlation)
35	C	Maximum de l'histogramme pour un canal (Histogram Max)
36	C	Moyenne d'un canal (Mean)
37	C	Maximum d'un canal (Maximum)
38	C	Minimum d'un canal (Minimum)
39	C	Somme du résultat d'un filtre Sobel sur un canal (Sobel)
40	C	Entropie sur un canal (Entropy)

La deuxième colonne du tableau 4.2 mentionne le type de chaque caractéristique. *M* est utilisé pour Morphologique, *T* pour Texture et *C* pour Couleur.

La caractéristique *17*, la *complexité*, correspond à l'aire divisée par le périmètre au carré. Cette caractéristique informe sur l'irrégularité du contour de l'objet.
La caractéristique *20*, *Moment – valeur R*, est une valeur normalisée du second moment.
Les caractéristiques *25-28*, *valeurs de co-occurrences*, utilisent la co-occurrence de valeurs de pixels juxtaposés (soit vertical, horizontal ou diagonal) pour l'image transformée en 8 niveaux de gris. À partir de la matrice de co-occurrence, de taille 8 pixels par 8 pixels, le contraste, la corrélation, l'énergie et l'homogénéité sont calculés, pour les trois directions.
Les caractéristiques 29-*31*, *valeurs de la forme binaire locale*, utilisent un voisinage de 16 points sur une zone circulaire avec un rayon mesurant 2 pixels (*Voir* section 4.4.1).

Les calculs de caractéristiques concernant la forme binaire locale et la transformée de Fourier sont détaillés respectivement aux sections 4.4.1 et 4.4.2.

Étant donnée les caractéristiques choisies, nos bases de données auront un nombre différent de caractéristiques :

- Les base d'images de couleurs segmentées (céréales, raisins, nœuds de bois) auront 60 caractéristiques soit 17 morphologiques, 25 sur la texture et 18 sur la couleur;
- Les bases d'images en niveaux de gris et segmentées (grains de pollen et chiffres) auront 48 caractéristiques soit 17 morphologiques, 25 sur la texture et 6 sur la couleur;

- Les bases d'images en couleur mais non segmentées (feuilles et feuilles rognées) auront 43 caractéristiques soit 25 sur la texture et 18 sur la couleur (aucune morphologique).

4.4.1 Forme binaire locale

La forme binaire locale, tirée de l'anglais « Local Binary Pattern » (LBP), est un opérateur d'analyse de texture de niveaux de gris considéré invariant, obtenu de la définition de la texture dans un voisinage rapproché (Ojala, Pietikäinen et Harwood, 1996). Le concept du LBP est simple : un code binaire qui décrit la texture locale d'une forme est calculé à partir du seuillage d'un voisinage par la valeur en niveau de gris de son centre.

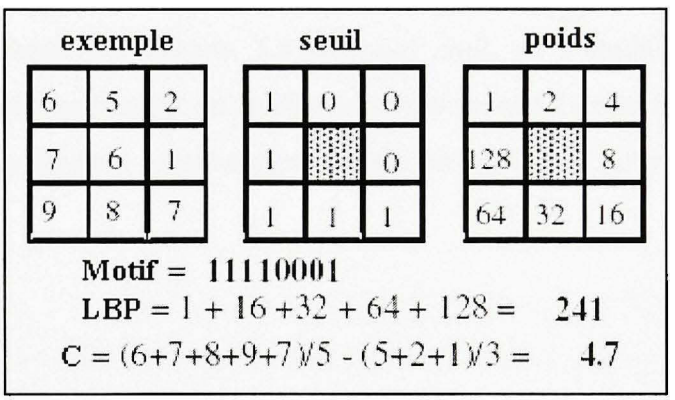


Figure 4.11 ***Ancienne définition du LBP.***

(Tiré de (University of Oulu Machine Vision Group) [notre traduction])

Source : traduction d'une figure tirée de la page 1 du document *LBP Methodology*, produit par le Machine Vision Group de l'Université d'Oulu, en Finlande.

La valeur *C* représente une mesure locale du contraste. Elle correspond à la moyenne des niveaux de gris au dessus ou égaux à la valeur centrale moins celle des valeurs en dessous. Les distributions de *LBP* et *C* sur deux dimensions sont utilisées comme caractéristiques.

Le LBP a connu plusieurs avancées depuis son introduction pour l'analyse d'images. Les améliorations du LBP ont fait apparaître plusieurs particularités recherchées : invariance à la rotation, invariance à un changement de niveau de gris monotonique et simplicité de calcul.

On a démontré (Ojala, Pietikäinen et Mäenpää, 2002) que l'équation 4.1 suffit à calculer un LBP avec un voisinage de P pixels, sur un rayon R, simplement en comptant la somme de niveaux de gris g plus grands que la valeur centrale. Un critère d'uniformité U indique si le motif est notable, si tel n'est pas le cas, une valeur par défaut $P+1$ est attribuée.

$$LBP_{P,R}^{riu2} = \begin{cases} \sum_{p=0}^{P-1} s(g_p - g_c), U(G_P) \leq 2 \\ P+1, \textit{autrement} \end{cases} \tag{4.1}$$

Pour utiliser l'opérateur LBP, l'usager doit choisir le nombre de voisins ainsi que la taille du rayon servant à localiser les voisins. La notation *riu2*, de l'anglais « rotation invariant uniform », indique que l'opérateur est uniforme et invariant à la rotation (Ojala, Pietikäinen et Mäenpää, 2002). La figure 4.12 présente des exemples de choix pour P et R.

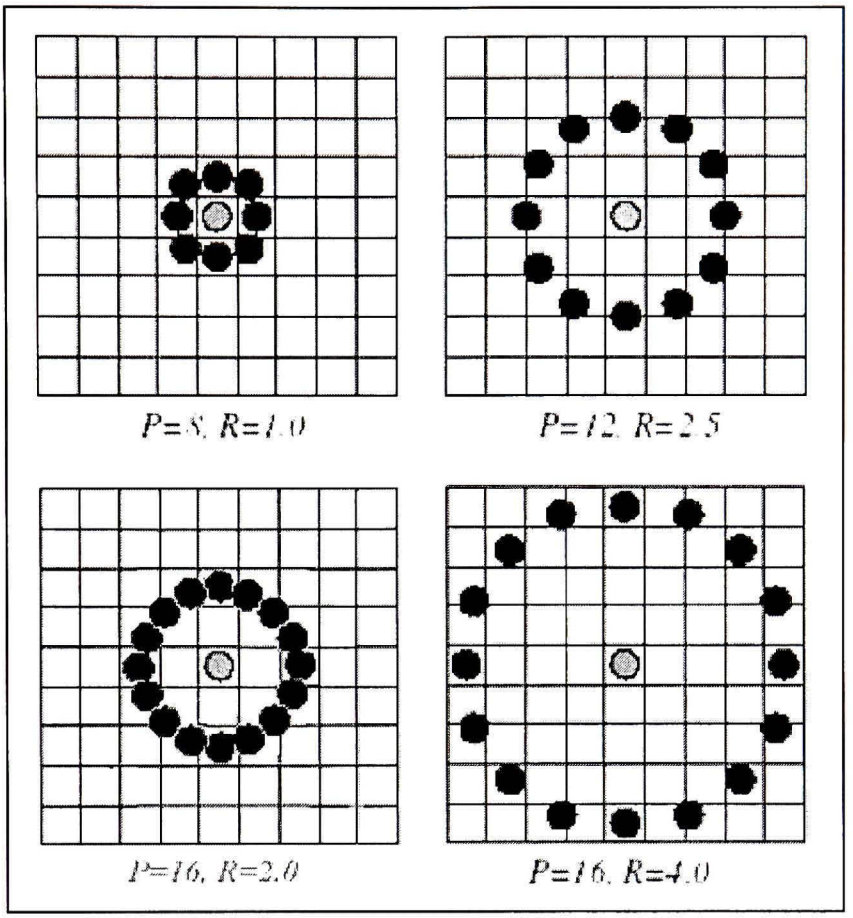


Figure 4.12 ***Voisinages circulaires et symétriques pour LBP.***

(Tiré de University of Oulu Machine Vision Group)

Source : modification d'une figure tirée de la page 2 du document *LBP Methodology*.

Le calcul du LBP sur une image nous fournit un histogramme des motifs (représentés par leur code binaire) observés dans le voisinage choisi. Nous avons décidé d'utiliser 3 valeurs statistiques issues de cet histogramme comme caractéristiques, soit la valeur R (valeur normalisée obtenue de la variance), l'asymétrie et le coefficient d'aplatissement.

4.4.2 Transformée de Fourier

L'analyse en fréquence d'images fournit des informations pertinentes relatives à la régularité ou à la redondance des valeurs de pixels de l'image. Ces informations sont utiles pour la détection de bruit. Elles servent également à mettre au point des filtres dans le domaine des

fréquences, qui peuvent adoucir ou « aiguiser » (en référence aux filtres passe-bas ou passe-haut) une image, rendant plus flous ou plus contrastés les contours des objets qui s'y retrouvent (Forsyth et Ponce, 2003). L'analyse en fréquence permet aussi de révéler la présence de patrons réguliers comme les textures.

La Transformée de Fourier Discrète (TFD), une version échantillonnée de la transformée originale, est utilisée pour traiter les images numériques. Le nombre de fréquences correspond au nombre de pixels de l'image dans le domaine spatial. L'équation 4.2 présente le calcul de la TFD pour une image de *M* par *N* pixels :

$$F(k,l) = \frac{1}{MN} \sum_{a=0}^{M-1} \sum_{b=0}^{N-1} f(a,b) \cdot e^{-j2\pi\left(\frac{ka}{M}+\frac{lb}{N}\right)} \tag{4.2}$$

Dans l'équation 4.2, *f(a,b)* est l'image dans le domaine spatial, puis le terme exponentiel est la fonction de base correspondant à chaque point *F(k,l)* dans l'espace de Fourier. L'équation peut être interprétée comme suit : la valeur de chaque point *F(k,l)* est obtenue en multipliant l'image spatiale avec la fonction de base et en sommant les résultats. Comme les fonctions de base sont des sinus et cosinus pour différentes fréquences, *F(0,0)* représente la composante continue de l'image, soit le niveau de gris moyen, puis *F(M-1, N-1)* représente la fréquence la plus élevée.

La suite de cette section présente la méthode que nous avons utilisée pour extraire un nombre limité de caractéristiques à partir du résultat de la transformée de Fourier d'une image.

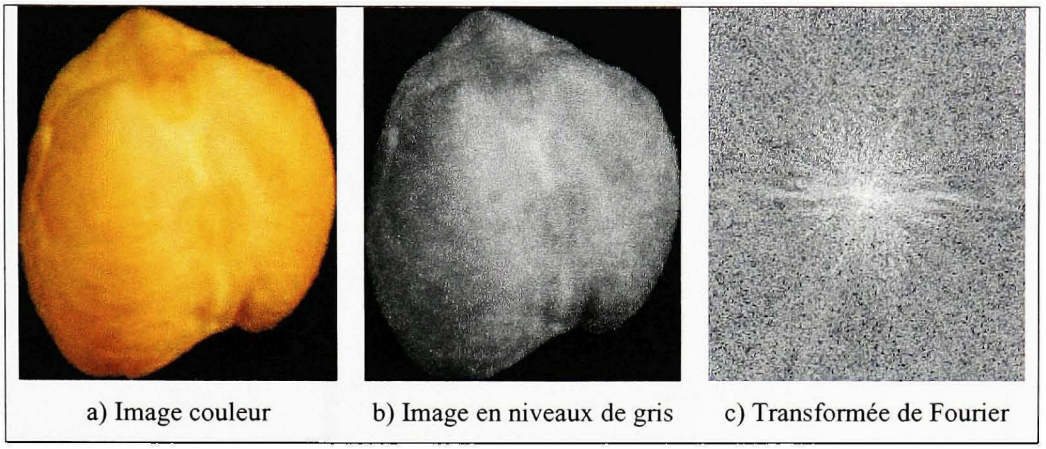


Figure 4.13 ***Transformée de Fourier sur une image.***

La première étape est d'effectuer une TFD, en ne conservant que la portion réelle (le spectre des fréquences) de la réponse. L'information sur l'angle de phase est ignorée. On calcule alors, à partir de l'image en fréquence centrée (*Voir* figure 4.13c), les sommes des valeurs de pixels sur un cercle centré à l'origine.

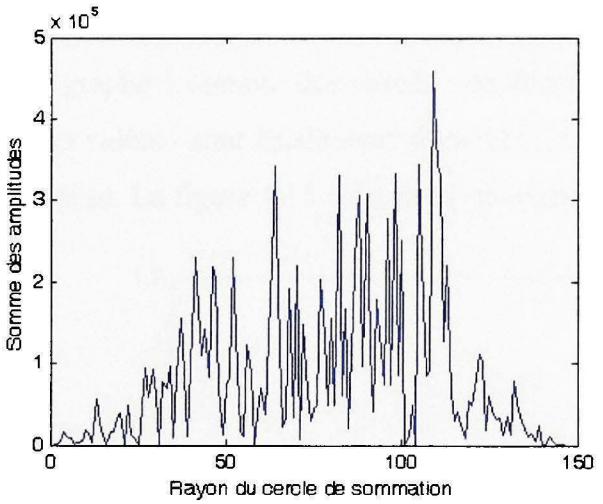


Figure 4.14 ***Somme des amplitudes de la transformée de Fourier sur un cercle.***

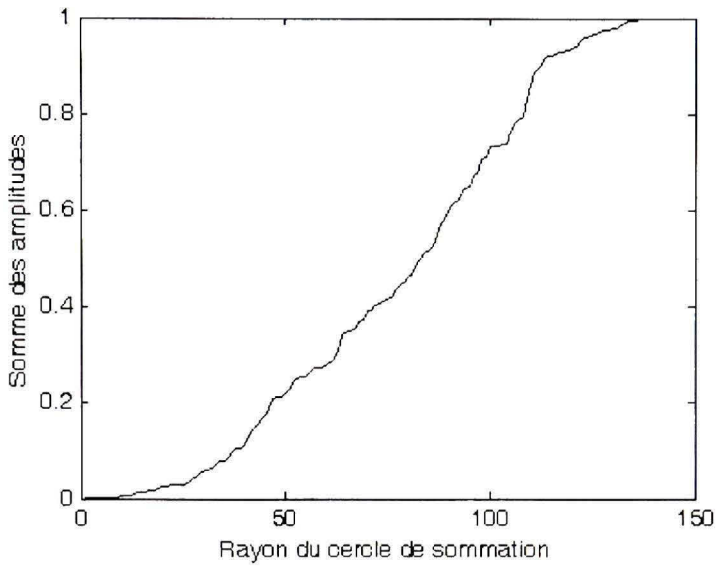


Figure 4.15 ***Graphe cumulatif normalisé de la somme des amplitudes.***

Les sommes pour un cercle d'un rayon de longueur 0 jusqu'au plus grand rayon possible sont conservées (un graphe « somme des pixels » en fonction du rayon est ainsi construit (*Voir* figure 4.14). Ces valeurs sont finalement sommées, puis normalisées pour obtenir un graphe cumulatif normalisé. La figure 4.15 en montre un exemple.

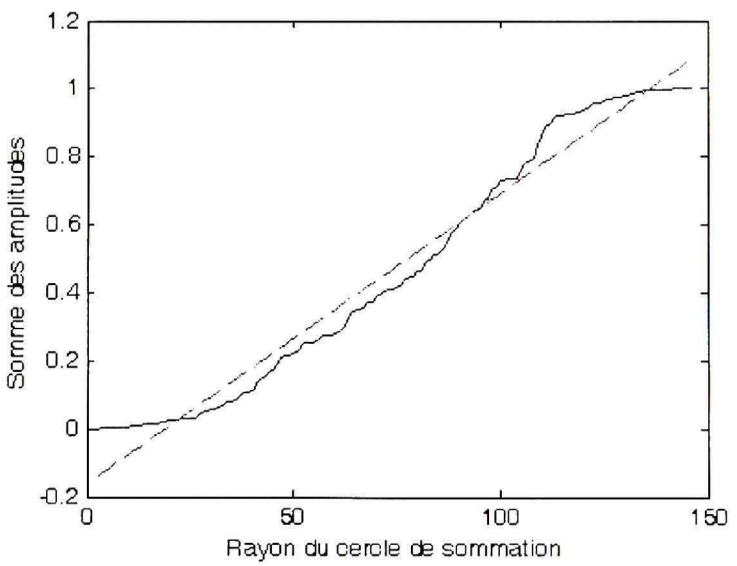


Figure 4.16 ***Approximation par régression linéaire de la courbe cumulative.***

À la toute fin, à partir d'une approximation par régression linaire (*Voir* figure 4.16) de la courbe cumulative, nous calculons les trois caractéristiques suivantes : la pente, l'ordonnée à l'origine et la corrélation.

4.4.3 Préparation des bases de données

L'extraction des caractéristiques est donc réalisée avec Matlab. Nous avons élaboré deux procédures différentes, une pour les images segmentées, qui extrait toutes les caractéristiques présentées au tableau 4.2, et une pour les images de feuilles, non segmentées, qui ne tient pas compte des caractéristiques morphologiques (notées *M* au tableau 4.2), puisque la forme de l'objet est inconnue.

Weka utilise un seul format de base de données appelé « arff » pour Attribute-Relation File Format (Reutemann, 2007). C'est un format simple entièrement en texte qui permet à l'usager de faire des modifications rapide sans interface autre qu'un éditeur de texte. L'entête est composée des caractéristiques et des classes, identifiées par mots-clé, puis les données suivent, groupées par échantillon. Nous avons produit un script Matlab qui écrit les valeurs des caractéristiques calculées pour les échantillons dans un fichier texte qui respecte le format arff. Les scripts utilisés sont disponibles pour téléchargement sur internet (Levasseur, 2008b). Les bases de données arff produites à partir de chaque lot d'images présenté à la section 4.3 peuvent aussi être téléchargées (Levasseur, 2008a).

À partir des caractéristiques extraites, nous produirons de nouvelles bases de données de dimension moins élevée à l'aide de l'ACP. Weka offre un algorithme d'ACP dans sa section dédiée à la sélection de caractéristiques. Les classes suivantes permettent de réaliser l'ACP avec Weka :

- *filters.supervised.attribute.AttributeSelection*
- *attributeSelection.PrincipalComponents*
- *attributeSelection.Ranker*.

La procédure d'ACP de Weka réalise une transformation des données selon le principe présenté à la section 2.5.1. La réduction de la dimensionnalité est accomplie par la sélection (la procédure *Ranker* de Weka permet cette sélection) de suffisamment de vecteurs propres afin de tenir compte d'un pourcentage de la variance des données originales. La quantité de variance conservée est 95%. L'ACP de Weka procède aussi par défaut à une normalisation des données.

4.5 Séparation des bases de données

Les bases de données arff contenant les caractéristiques seront séparées en deux portions, une pour l'apprentissage, l'autre pour le test. Deux tiers des données seront utilisés pour la base de données d'apprentissage et le tiers restant sera utilisé pour la base de test. Le découpage en deux bases sera réalisé par le filtre suivant offert par Weka :

- *supervised.instance.StratifiedRemoveFolds*

Ce filtre permet de séparer une base en deux. Une option nous permet de séparer en fractions et d'affecter les fractions à la base voulue en sortie (ce qui permet notre découpage deux tiers un tiers). Ce filtre conserve la proportion des classes dans les bases de sorties, ce qui contribue à stabiliser les résultats. Finalement, on peut découper une même base plusieurs fois et obtenir des bases d'apprentissage et de test différentes en changeant le paramètre « seed » du filtre.

4.6 Classification

Les classificateurs utilisés sont ceux présentés à la section 2.6. Ces classificateurs sont très souvent utilisés dans des articles ou des algorithmes de classification sont comparés (Duda, Hart et Stork, 2000; Jain, Duin et Mao, 2000). Les paramètres utilisés pour chaque classificateur sont les paramètres par défaut de Weka, ou une sélection qui s'est montrée plus performante que les valeurs par défaut lors d'une évaluation non exhaustive. L'information sur les algorithmes utilisés est présentée au tableau 4.3.

Tableau 4.4

Information sur l'utilisation des classificateurs de Weka

Nom (identifiant)	**Appel avec Weka (*weka.classifiers.*)**	**Options choisies**
Classificateur Naïf de Bayes (Bayes)	*bayes.NaiveBayes*	Aucune
Séparateur à vastes marges : noyau linéaire (SVM)	*functions.SMO*	Normalisation et fonction de noyau linéaire : $k(x_i,x) = x_i \cdot x$
Séparateur à vastes marges : noyau polynomial de deuxième degré (SVM2)	*functions.SMO*	Normalisation et fonction de noyau polynomiale de deuxième degré : $k(x_i,x) = (x_i \cdot x + 1)^2$
K plus proches voisins (KPPV)	*lazy.IBk*	Normalisation et $K = 3$ voisins
Perceptron multicouche (PM)	*functions.MultilayerPerceptron*	Par défaut
Arbre de décision C4.5 (C4.5)	*trees.J48*	Par défaut
Programmation génétique avec orchestration (PG)	*functions.GeneticProgramming*	Options présentées dans le tableau 3.5, à la section 3.7.

Dans le tableau 4.3, l'*identifiant* est le mot qui sera utilisé dans les graphes de résultats pour un algorithme. *Appel avec Weka* fait référence au nom de la classe à l'intérieur de la librairie *weka.classifiers*. Weka fournit les références utilisées pour la programmation de chaque algorithme dans sa documentation.

Pour ce qui est du perceptron multicouche, le nombre de neurones de la couche cachée dépend de la base de données : il correspond à la moitié de la somme du nombre de caractéristiques et du nombre de classes. Le critère d'arrêt sera toujours 500 époques. L'ensemble d'entraînement n'est pas divisé en ensemble de validation pour éviter le sur-apprentissage.

Dans le cas de la programmation génétique avec orchestration, les paramètres choisis sont les paramètres par défaut (*Voir* section 3.7, tableau 3.5) à l'exception des suivants :

Tableau 4.5

Paramètres de la PG avec orchestration

Paramètre	**Valeur**
Taille de la population	3000
Profondeur maximale	10
Critères d'arrêt	Adéquation = 0.999 Générations = 60 Temps = 2400 minutes
Scénario évolutif	Sélection après reproduction Taille de la nouvelle population = 5000
Taille de l'élite	25 (par classe, utilisés pour l'orchestration)

4.7 Classification avec boosting

Tous les algorithmes, à l'exception du KPPV – inapproprié puisque il ne comprend pas de phase d'apprentissage -, seront utilisés une seconde fois, en combinaison avec le méta algorithme de boosting (*Voir* section 2.7.2). Dans le cas de la PG, le boosting sera incorporé au processus de PG tel que décrit à la section 3.5.1.

L'appel de l'algorithme de boosting fourni par Weka se fait à l'aide de la classe suivante :

- *weka.classifiers.meta.AdaBoostM1*

Cet algorithme de boosting est basé sur les travaux de Freund et Schapire (1996). Nous permettrons un nombre maximal de dix itérations pour le boosting. Finalement, lorsque nous utiliserons le boosting avec la méthode AdaBoost pour combiner un type de classificateur, son identifiant sera précédé de la lettre 'A' (ainsi SVM avec AdaBoost devient ASVM, ainsi de suite). Voici les séquences d'appels nous permettant de combiner les classificateurs pour le boosting :

Tableau 4.6

Séquences d'appels pour classificateurs avec boosting

Nom (Identifiant)	**Appel avec Weka (*weka.classifiers.*)**
Boosting pour classificateur Naïf de Bayes (ABayes)	*meta.AdaBoostM1 + bayes.NaiveBayes*
Boosting pour Séparateur à vastes marges : noyau linéaire (ASVM)	*meta.AdaBoostM1 + functions.SMO*
Boosting pour Séparateur à vastes marges : noyau polynomial de degré 2 (ASVM2)	*meta.AdaBoostM1 + functions.SMO*
Boosting pour Perceptron multicouche (APM)	*meta.AdaBoostM1 + functions.MultilayerPerceptron*
Boosting pour Arbre de décision C4.5 (AC4.5)	*meta.AdaBoostM1 + trees.J48*
Programmation génétique avec boosting (PGB)	*functions.GeneticProgramming*

Les classificateurs utilisés avec l'algorithme de boosting AdaBoost ont un identifiant précédé de la lettre 'A'. La PG avec boosting est quand à elle identifiée « PGB », car elle n'utilise pas AdaBoost mais un boosting intégré que nous avons programmé. En ce qui concerne la programmation génétique avec boosting, les paramètres choisis sont les suivants :

Tableau 4.7

Paramètres de la PGB

Paramètre	**Valeur**
Taille de la population	250 programmes
Profondeur maximale	5 niveaux
Critères d'arrêt	Adéquation = 0.9 Générations = 80 Temps = 720 minutes
Scénario évolutif	Sélection après reproduction Taille de la nouvelle population = 500
Taille de l'élite	25 (par classe, utilisés pour le boosting)

4.8 Déroulement des expérimentations

Nous désirons comparer les résultats de classification selon les deux critères suivants :

1. Taux d'erreur de classification : il correspond au nombre d'erreurs de classification sur le nombre total d'échantillons dans la base de test. Il sera représenté en pourcentage;

2. Temps d'apprentissage du classificateur : c'est le temps total dédié à la phase d'apprentissage du classificateur, divisé par le nombre d'échantillons compris dans la base d'apprentissage. Cette valeur est donc calculée en seconde par échantillon.

Pour évaluer ces critères, nous procéderons à 50 séparations aléatoires et différentes de chacune des bases de données utilisées selon la méthode discutée à la section 4.5. Pour chaque classificateur, 50 valeurs de taux d'erreur et de temps d'apprentissage seront donc disponibles, ce qui permettra une analyse statistique significative.

Nos expérimentations se dérouleront en deux parties. Lors d'une première série, chacun de nos classificateur sans boosting (avec orchestration dans le cas de la PG) réalisera la classification des échantillons provenant des bases de données de céréales (grande), de feuilles, de grain de pollen, de nœuds de bois et de raisins, chaque base avec et sans transformation par ACP.

Après avoir obtenu les résultats de la première série, nous mettrons sur pied la version de PG avec boosting intégré et recommenceront les expériences. Chaque classificateur, avec et sans boosting (boosting intégré dans le cas de la PG) fournira des prédictions pour toutes les bases de données sans transformation par ACP, sauf pour celle des céréales (grande), qui ne sera plus utilisée. Les bases de chiffres, feuilles rognées et la petite base de céréales seront

introduites pour cette série d'expérimentations. Un tableau en Annexe I présente les expérimentations qui seront réalisées lors de la première et de la deuxième série.

Finalement, notons que l'ordinateur utilisé pour les expérimentations possède deux processeurs Xeon double coeur à 2.66 MHZ et 5 Giga octets de mémoire vive. Le système d'exploitation utilisé est « Mac OS X v10.4 Tiger ».

4.9 Mesures de performance

Plusieurs termes statistiques seront utilisés pour comparer les résultats. Voici les définitions selon lesquelles nous avons utilisé ces termes :

- La **médiane** est une valeur unique qui sépare l'ensemble des valeurs mesurées en deux parties égales. Une partie est inférieure à la médiane et l'autre lui est supérieure. Dans le cas d'un nombre pair de valeurs (la médiane ne pourra alors être une des valeurs de l'ensemble et respecter sa définition), la médiane sera la valeur à mi-chemin entre les deux valeurs centrales de l'ensemble. On peut interpréter la médiane comme la valeur typique des mesures. La médiane est un indicateur plus stable que la moyenne face à des valeurs extrêmes. La lettre M sera utilisée pour identifier la médiane dans les tableaux de résultats.

- Les **quartiles** sont les valeurs qui séparent les mesures en quatre parties égales. Le premier quartile sépare le quart des mesures les plus faibles de l'ensemble. Le deuxième quartile correspond à la médiane. Le troisième sépare le quart des mesures les plus élevées. La zone délimitée par le premier et le troisième quartile couvre la moitié de toutes les mesures de l'ensemble. Cette zone présente une bonne esquisse des valeurs attendues pour de nouvelles expériences dans les mêmes conditions.

- L'**écart interquartile** E_i correspond à la différence entre le troisième et le premier quartile. Cette valeur est utilisée pour évaluer la dispersion des mesures.

- Les **valeurs extrêmes** sont des mesures qui se trouvent particulièrement loin de la médiane. On considère comme extrême une valeur dont la différence avec la médiane est supérieure à $E_i \bullet c$. La constante c varie selon les auteurs.

- La **moyenne** $\bar{x}$ est la valeur unique que devraient avoir tous les n échantillons d'un ensemble de mesures pour que leur total reste inchangé. Elle se calcule ainsi :

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{4.3}$$

- L'**écart type** $\hat{\sigma}$ caractérise la répartition des mesures autour de leur moyenne. Il sert à mesurer la dispersion d'un ensemble de données. Plus l'écart type est faible, plus les valeurs sont homogènes. Nous calculons l'écart type ainsi :

$$\hat{\sigma} = \sqrt{\frac{1}{(n-1)}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{4.4}$$

4.9.1 Représentation des résultats

Une partie des résultats sera représentée graphiquement à l'aide de « boîtes à moustaches » (mieux connues sous le nom anglais de « box and whiskers plot »). Cette méthode de représentation a été introduit par Tukey (Tukey, 1977) afin de comparer visuellement des valeurs statistiques comme la médiane et les quartiles. Nous utiliserons une représentation par boîte à moustache telle que décrite par Vellemean (Velleman et Hoaglin, 1984), illustrée à l'aide de la figure 4.17.

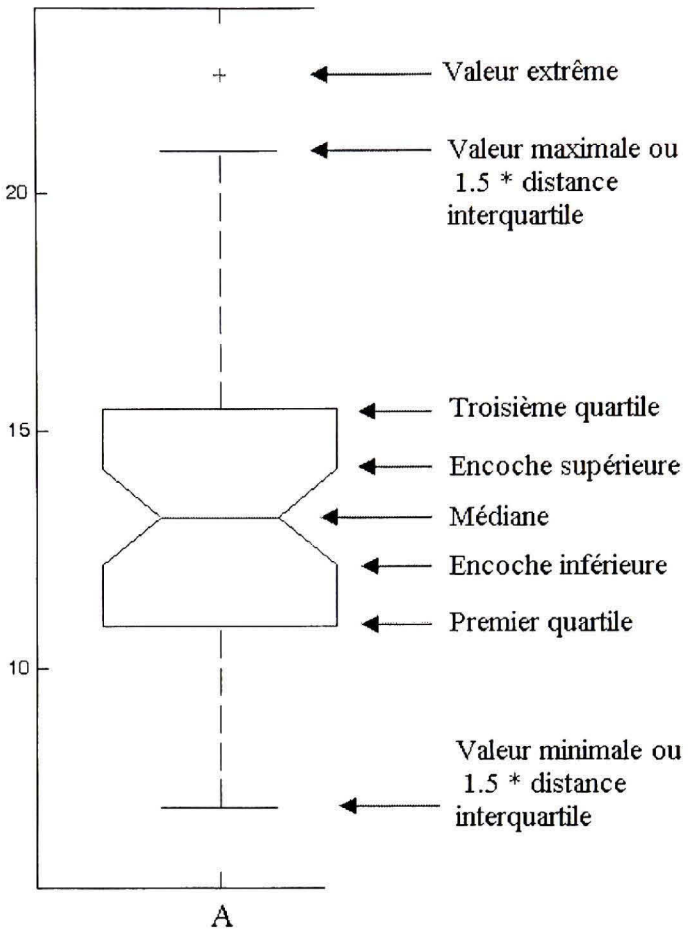


Figure 4.17 ***Interprétation de boîte à moustache.***

La figure 4.17 révèle une boîte à moustache pour l'ensemble de données *A*. La boîte présente des lignes horizontales pour le premier quartile, la médiane, et le troisième quartile. Les moustaches sont les prolongements de la boîte, qui indiquent l'étendue du reste des données. La constante *c* est ici fixée à 1.5. Les moustaches s'étirent donc jusqu'à la dernière valeur (borne inférieure ou supérieure) de l'ensemble ou jusqu'à un maximum de $1.5 \bullet E_i$ à partir de la médiane. Les valeurs extrêmes sont illustrées par des plus (+), situés en dehors des moustaches. La boîte comporte également deux encoches. Ces encoches sont une estimation robuste de l'incertitude de la médiane, utile pour la comparaison de deux boîtes. La médiane réelle est située, avec une certitude de 95%, quelque part entre l'encoche inférieure et supérieure.

Ce type de boîtes à moustaches permet une interprétation rapide d'un ensemble de mesures. La boîte elle-même présente la moitié des résultats. L'étendue de la boîte et des moustaches indique la dispersion des mesures. D'une moustache à l'autre, pratiquement toutes les mesures sont représentées (il y aura à l'occasion des valeurs extrêmes à l'extérieur des moustaches).

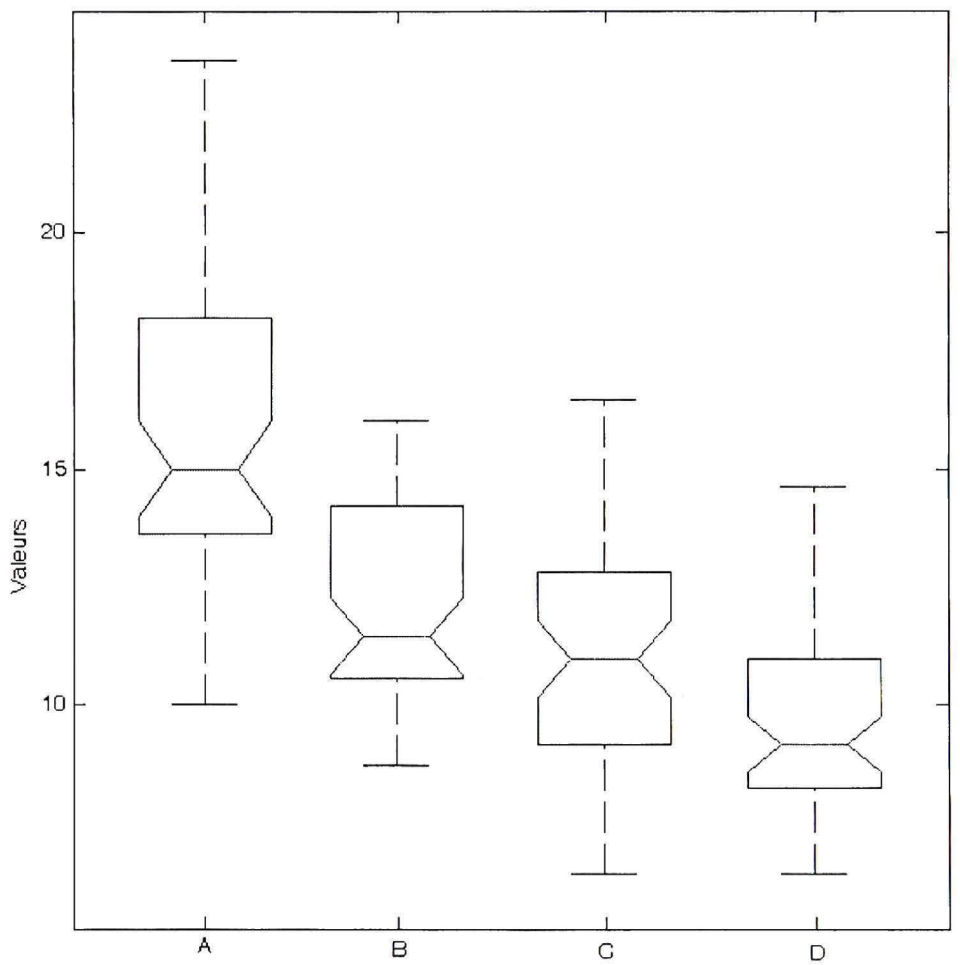


Figure 4.18 ***Comparaisons de boîtes à moustache.***

Afin de comparer la performance de prédiction des classificateurs, nous utiliserons, en tant que premier critère, la médiane. Les encoches présentent en plus la certitude avec laquelle on peut comparer deux médianes. Un deuxième critère consiste à comparer les premiers et

troisièmes quartiles. La figure 4.18 illustre quatre ensembles de mesures pour les classificateurs fictifs *A*, *B*, *C* et *D*. À l'aide de cette figure, nous présentons dans le tableau 4.8 notre interprétation des ensembles.

Tableau 4.8

Exemples d'interprétation des résultats pour la figure 4.18

Ensembles à comparer	**Exemple d'interprétation**
A et B	La médiane de *A* est au-dessus de la boîte de *B*. La valeur typique de *A* est supérieure au trois quarts des valeurs de *B*.
A et C	La boîte de *C* est entièrement en dessous de la boîte de *A*. Trois quart des valeurs de *C* sont inférieures au trois quarts des valeurs de *A*. Les valeurs de *C* sont nettement au dessous de *A*.
B et C	Les encoches de *B* chevauchent celles de *C*. Les valeurs typiques de *B* et de *C* sont comparables.
C et D	Les boîtes de *C* et de *D* se recouvrent, mais pas leurs encoches. La vraie valeur typique de *D* est inférieure à celle de *C*, avec une grande probabilité.

Dans le but de faciliter la lecture des graphiques, les valeurs extrêmes seront omises. La présence de ces valeurs dans nos ensembles de mesures est de toute façon un phénomène d'exception.

Finalement, lorsque les données seront trop nombreuses pour être représentées graphiquement de façon conviviale, nous emploierons des tableaux présentant trois valeurs statistiques utiles pour l'interprétation des données : la médiane, la moyenne et l'écart type.

CHAPITRE 5

RÉSULTATS ET ANALYSE

Ce chapitre est divisé en trois parties. Premièrement, nous soumettons les résultats pour le taux d'erreur de classification. La deuxième partie détaille les temps d'apprentissage. La dernière section comporte une discussion et une analyse générale des résultats.

Dans les parties un et deux, les résultats seront présentés en deux blocs, soit un pour la première série d'expériences (algorithmes sans boosting) et un pour la deuxième série d'expériences (algorithmes avec boosting). Pour chaque problème, nous rappellerons quelques informations sur la base de données utilisée, introduirons les graphiques, puis présenterons une courte analyse des résultats.

5.1 Taux de reconnaissance

La présente section est divisée en deux parties. La première détaille les résultats de classification pour la première série, sans transformation des données par ACP. La seconde révèle les résultats avec l'ACP et les compare avec les résultats présentés précédemment.

Les identificateurs utilisés pour les graphiques sont ceux utilisés dans les tableaux 4.3 et 4.5 (section 4.6 et 4.7). Un tableau sommaire des erreurs de classification pour la première série est présenté à l'Annexe II.

5.1.1 Première série : bases de données sans ACP

La première série d'expérimentations comprend 50 expériences pour les bases de céréales (grande), nœuds, feuilles, pollen et raisins secs (toutes sans ACP). Les résultats sont présentés dans cet ordre. Les classificateurs utilisés sont Bayes, KPPV, C4.5, PG orchestrée, SVM, SVM2 et PM.

Grande base de céréales

La grande base de céréales contient 950 images par classe. Pour les six classes, on obtient un total de 5700 images. Comme pour toutes les bases de données, deux tiers sont utilisés pour l'entraînement (3800 images) et le tiers restant pour le test (1900 images). Nous avançons qu'il n'y a aucune erreur de classification dans la base de données (l'ayant réalisée nous-mêmes).

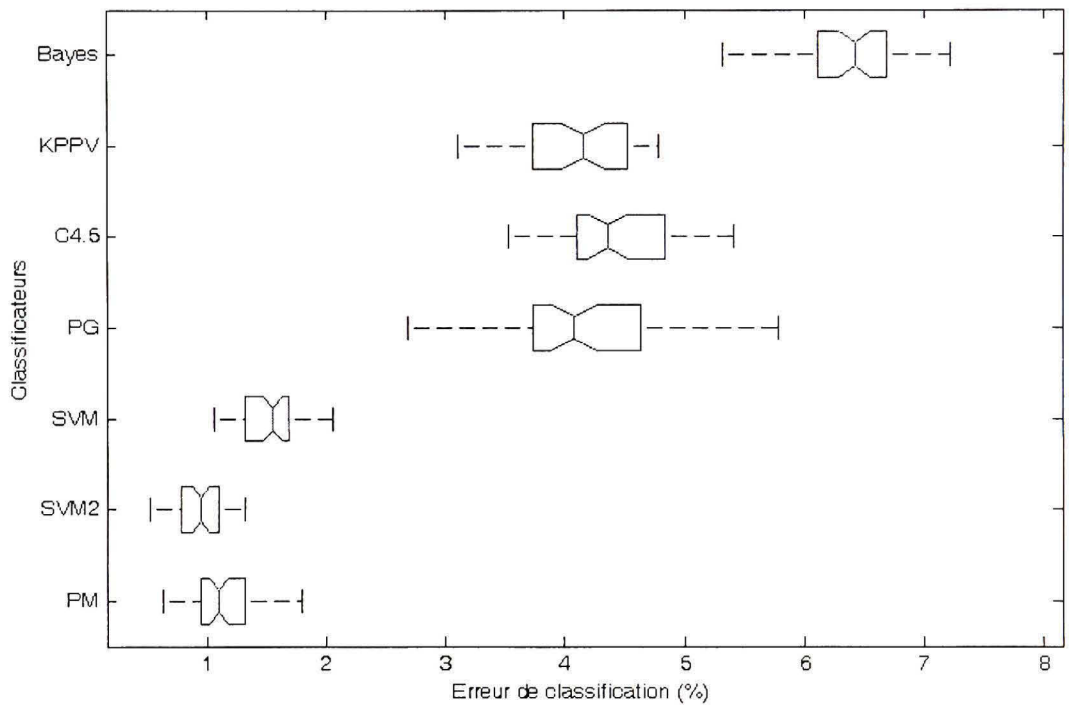


Figure 5.1 ***Erreurs de classification pour la grande base de céréales : première série.***

Les erreurs de classification pour ce problème sont très faibles. On peut distinguer trois groupes de classificateurs en fonction de leur erreur. Bayes est seul de son côté, avec une médiane de 6.42%. Ensuite, KPPV, C4.5 et la PG orchestrée ont une médiane qui varie entre 4 et 4.5% d'erreur. En particulier, KPPV et la PG présentent des médianes comparables. La PG produit des résultats qui fluctuent plus que KPPV et C4.5, à travers les 50 expériences. Finalement, le troisième groupe est constitué des deux versions de SVM, puis du PM. Le SVM2 offre la meilleure performance de classification, avec une médiane de 0.95% d'erreur et la plus petite variation de résultats. L'erreur de reconnaissance typique du SVM2 (SVM à

fonction de noyau polynomiale) est inférieure au trois quarts des erreurs obtenues avec le PM, ce qui démontre une certaine supériorité de classification pour cette base de données.

L'analyse des résultats nous permet de faire quelques observations additionnelles. Le classificateur naïf de Bayes présente une plus forte erreur que les autres. Ce résultat peut probablement s'expliquer du fait que la distribution de chaque caractéristique ne suit pas une distribution normale, comme l'assume l'algorithme. Ensuite, les performances du deuxième groupe sont satisfaisantes, mais pas aussi généralisantes que celles du troisième groupe. Finalement, nous croyons que le haut taux de classification pour ce problème s'explique par quatre qualités de la base de données :

1. La grande taille de la base d'apprentissage permet une bonne généralisation;

2. Il y a peu de chevauchement des données entre les classes. Quelques classes peuvent être confondues, mais restent relativement faciles à distinguer;

3. Il n'y a pas d'erreur de classification dans la base de données. L'inverse biaise l'évaluation de la généralisation du classificateur;

4. La segmentation des objets est appropriée, ce qui offre une grande quantité d'information pertinente pour la classification.

Base de nœuds de bois

La base de nœuds comprend 27 images par classe et six classes. De ces 162 images, 108 sont utilisées pour l'apprentissage et 54 pour le test. Il s'agit, dans le cadre de nos recherches, de la base de données ayant le plus petit nombre d'images par classe.

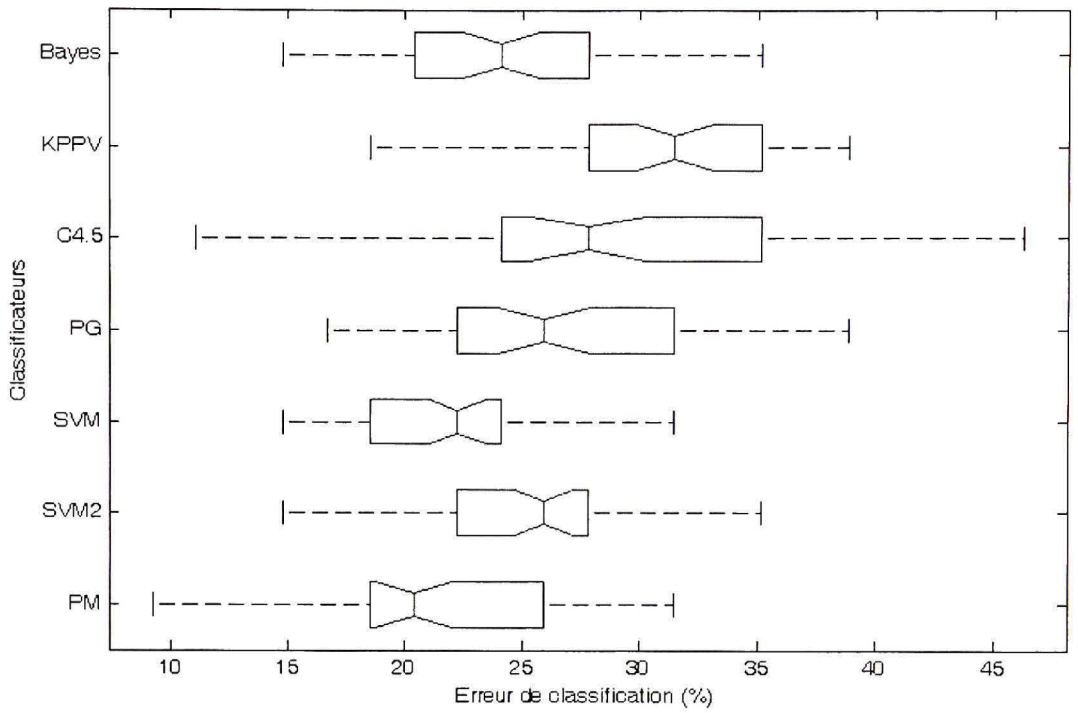


Figure 5.2 ***Erreurs de classification pour la base de nœuds de bois : première série.***

Les résultats de classification présentés à la figure 5.2 illustrent la difficulté du problème de classification des nœuds de bois : la plupart des classificateurs présentent de faibles performances de reconnaissance. De plus, les écarts interquartiles sont très grands, indiquant une variabilité de la performance de classification en fonction des exemples d'apprentissage présentés. KPPV obtient la plus grande médiane de l'erreur avec 31.48%. C4.5 présente quant à lui une variabilité notable, passant de plus de 45% d'erreur à près de 10%. Les meilleures performances sont obtenues par le PM, avec des résultats qui frôlent, dans certains cas, 32% d'erreur. La meilleure expérience de PM obtient un taux d'erreur d'environ 9%. Les valeurs de taux d'erreur observées à la figure 5.2 contrastent fortement avec celles provenant de la classification de la grande base de céréales (*Voir* figure 5.1).

Nous expliquons la faible performance de KPPV par le petit nombre d'échantillons par classe pour l'entraînement. En effet, il est difficile de trouver trois voisins significatifs à partir d'une base d'entraînement qui ne compte que 36 échantillons par classe et six classes. Plus globalement, les résultats indiquent la possibilité de sur apprentissage ou la difficulté de

généralisation. Cette dernière peut apparaître puisque le problème est défini par un faible nombre d'échantillons, contre un très grand nombre de caractéristiques (donc de dimensions). Il n'est pas surprenant d'observer cette problématique, connue sous le nom de fléau de la dimension (Bellman, 1961), puisque le problème de nœuds de bois comprend seulement 108 échantillons d'apprentissage pour 60 caractéristiques. Nous identifions aussi d'autres causes de la difficulté de classification pour la base de nœuds :

1. Nous jugeons qu'il s'agit d'un problème difficile, puisque la classification par l'humain offre de faibles résultats. De plus, le chevauchement de données entre plusieurs classes est grand;

2. La segmentation des nœuds a été réalisée sans directives d'experts du domaine. Il est possible que l'information pertinente à la classification ne fasse pas partie de l'image à partir de laquelle nous avons extrait les caractéristiques;

3. L'analyse des images à l'œil révèle une grande probabilité d'erreurs de classification dans la base originale.

Base de feuilles

La base de données de feuilles est composée de trois classes. De ses 180 images, nous en utilisons 120 pour l'apprentissage et 60 pour le test. Rappelons que nous n'avons pas réalisé la segmentation pour ce problème. Les caractéristiques morphologiques énoncées à la section 4.4 sont donc absentes de la base de données.

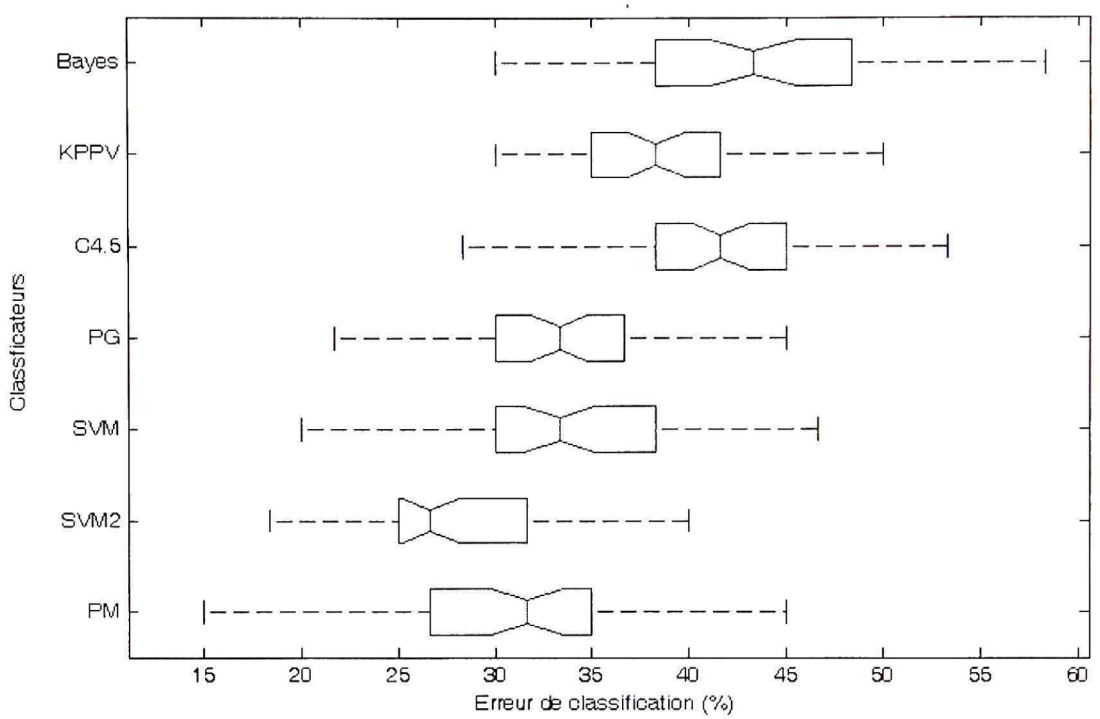


Figure 5.3 ***Erreurs de classification pour la base de feuilles : première série.***

Les résultats varient énormément, passant de 60% d'erreur à 15%. Notons qu'un taux de 60% d'erreur est très grand puisque choisir une classe au hasard, pour un problème ayant trois classes et le même nombre d'échantillons pour chaque classe, devrait obtenir en moyenne 66.7% d'erreur. On retrouve à peu près le même ordre des classificateurs que celui observé pour la grande base de céréales (*Voir* figure 5.1), mais avec plus de chevauchement. Pour tous les classificateurs, l'écart interquartile des résultats est très grand. Bayes produit la plus grande erreur médiane, suivi de C4.5. Ensuite, la PG apparaît supérieure à KPPV. L'ensemble des résultats de la PG a une médiane semblable à celle du SVM, pour une dispersion plus petite. Le classificateur offrant la plus petite médiane de l'erreur pour ce problème est SVM2, présentant du même coup une dispersion plus petite que les autres classificateurs et semblable à celle de KPPV. PM reste le second classificateur le plus performant, même si les trois quarts de ses erreurs sont supérieurs à la valeur typique de SVM2. Le PM réussit toutefois à obtenir quelques performances en dessous de 17.2%, qui correspond au meilleur résultat offert par SVM2.

Le problème de feuilles est difficile à résoudre en raison de la proportion de l'image qui ne concerne pas l'objet que nous voulons classifier. Nous estimons que l'objet d'intérêt représente environ 15% de l'image à partir de laquelle nous avons extrait les caractéristiques. Certaines caractéristiques comme la moyenne d'un canal ou son écart type sont dans la plupart des cas inutilisables pour la classification de la feuille puisqu'ils qualifient davantage le fond de l'image que la feuille. Finalement, nous observons aussi le fléau de la dimensionnalité, puisque seulement 120 images sont utilisées pour l'apprentissage, contre 43 caractéristiques (le nombre de caractéristiques est présenté au tableau 5.2).

Base de pollen

Sept types de grains de pollen, à raison de 196 échantillons par classe, forment cette base. Des 1372 échantillons, 915 sont utilisés pour l'entraînement, et les 457 autres pour le test. Comme les images de pollen ne sont pas en couleur mais en niveaux de gris, seulement 48 caractéristiques ont été extraites.

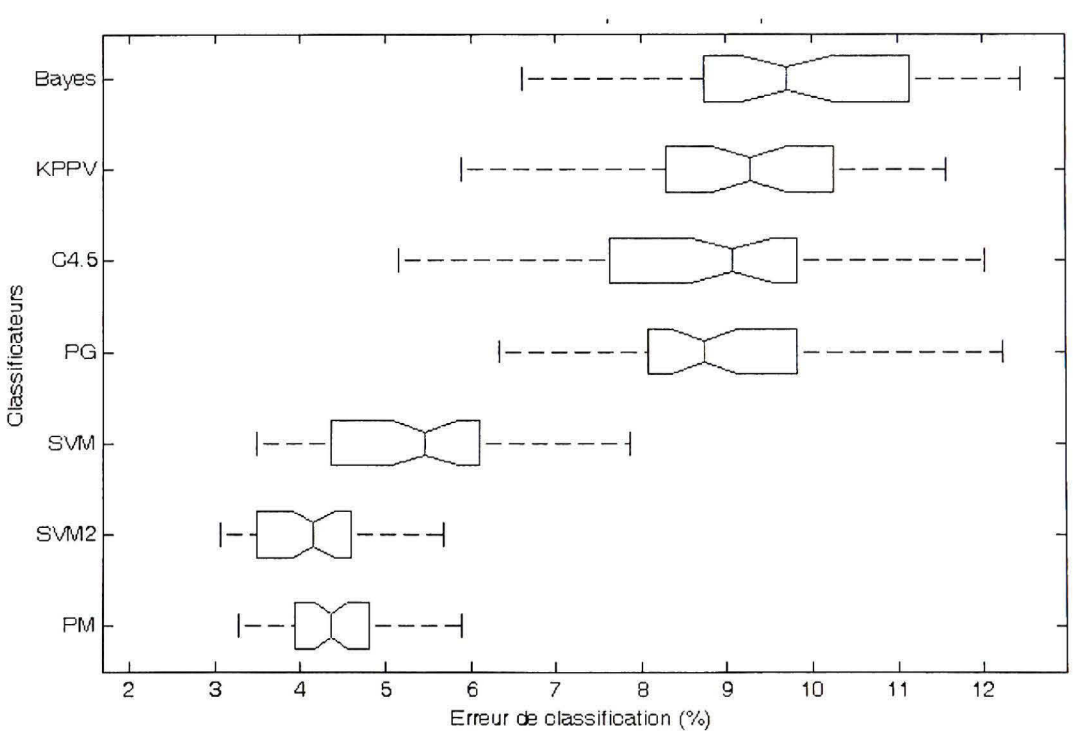


Figure 5.4 ***Erreurs de classification pour la base de pollen : première série.***

On observe pour ce problème une séparation marquée de deux groupes de classificateurs. Le premier est composé de Bayes, KPPV, C4.5 et de la PG orchestrée, tous les quatre avec une erreur médiane située entre 8.5% et 9.75%. Leur dispersion est également semblable, variant grossièrement entre 6% et 12% d'erreur. Le deuxième groupe est de beaucoup supérieur au premier groupe, tant par ses performances de classification que par sa dispersion. Le SVM, et en particulier le SVM2 et le PM, offrent les taux d'erreur les plus petits. Le SVM2 présente une médiane de 4.15% d'erreur, suivi de près par le PM avec 4.37%.

Ces résultats sont très similaires à ceux obtenus pour la grande base de céréales. Les classificateurs arrivent à une bonne généralisation, en particulier ceux du deuxième groupe. Quant à la légère baisse de performance par rapport au problème de céréales, nous l'expliquons ainsi :

1. Il y a sept classes, soit une de plus que pour la base de céréales, et beaucoup de chevauchement des données entre les classes;

2. La taille de la base d'apprentissage est inférieure à celle des céréales.

3. L'inspection de la base de données d'images nous a révélé qu'il est probable que la base originale contienne des erreurs de classification de grains : les différents classificateurs obtiennent des erreurs d'identification pour les mêmes échantillons suspects;

Base de raisins secs

Cette base contient 450 échantillons par classes et trois classes, pour un total de 1350 échantillons. On en utilise 900 pour l'apprentissage contre 450 pour le test. Comme la base de céréales, celle de raisins a été entièrement réalisée par nous.

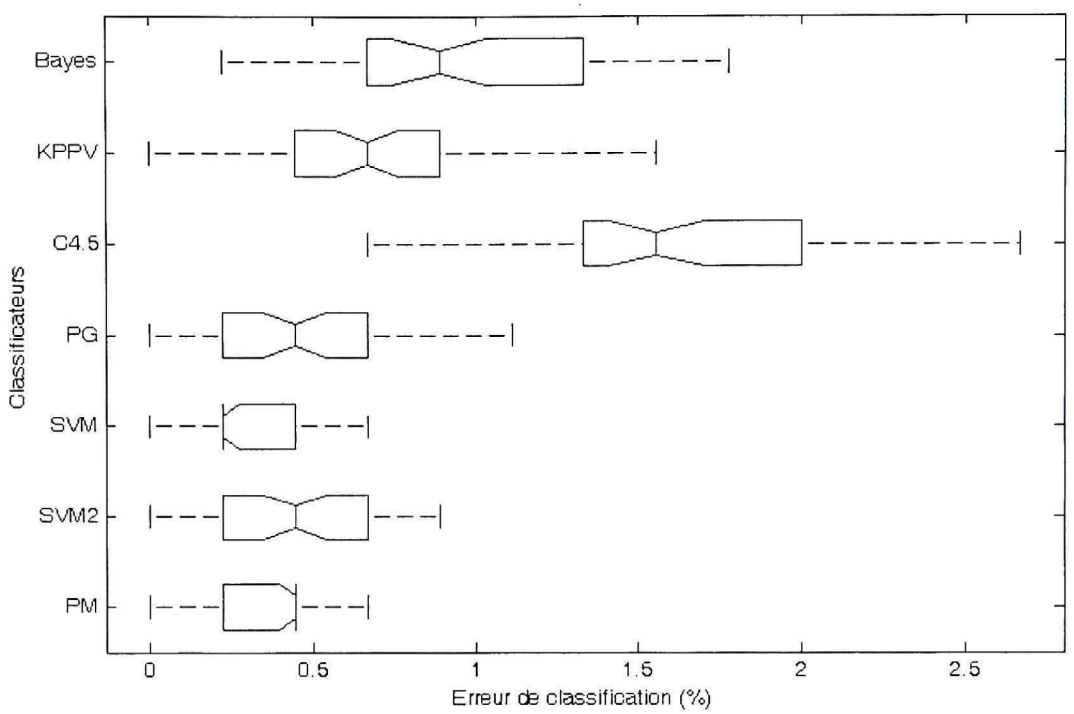


Figure 5.5 ***Erreurs de classification pour la base de raisins secs : première série.***

Tous les taux d'erreur pour ce problème sont extrêmement faibles. Le classificateur offrant les moins bonnes performances, C4.5, obtient une erreur médiane de seulement 1.56%. Bayes et KPPV le suivent avec respectivement 0.89% et 0.67% d'erreur. Trois autres algorithmes, la PG, le SVM2 et le PM fournissent des résultats similaires avec une médiane de 0.44% d'erreur. Finalement, le SVM obtient la meilleure médiane avec 0.22%

Ces résultats illustrent la facilité avec laquelle les classificateurs utilisés arrivent à résoudre le problème de reconnaissance des raisins secs. Mentionnons que les PG, SVM, SVM2 et PM ne peuvent être statistiquement différenciés à partir de la figure 5.5 puisque l'ensemble de leurs performances sont similaires (la différence de 0.22% entre SVM et SVM2 ne représente qu'un échantillon de test sur 450).

L'aisance de classification des raisins par un opérateur humain laisse envisager que la classification peut être réalisée en tenant compte de très peu de caractéristiques. En effet, les solutions présentées par C4.5 et la PG utilisent souvent moins d'une dizaine de

caractéristiques décrivant surtout la couleur et quelques fois la taille du raisin sec. Finalement, il semble que la grande quantité d'échantillons ait contribué au pouvoir de généralisation des classificateurs.

Observations générales

Ces expériences sur les bases de céréales, nœuds, feuilles, pollen et raisins nous informent à la fois sur la difficulté d'interprétation de chaque base de données et sur le pouvoir de généralisation des classificateurs pour ces bases. Voici un tableau récapitulatif des résultats de classification :

Tableau 5.1

Meilleures médianes de l'erreur de classification pour la première série

Base de données	**Meilleure erreur médiane**	**Meilleur classificateur**	**Deuxième meilleur classificateur**
Grande base de céréales	0.95%	SVM2	PM
Nœuds de bois	20.37%	PM	SVM
Feuilles	26.67%	SVM2	PM
Pollen	4.15%	SVM2	PM
Raisins secs	0.22%	SVM	PM, SVM2, PG

En premier lieu, on remarque que l'erreur de classification est très forte pour les bases de nœuds et de feuilles. Ces deux bases ont en commun le faible nombre d'échantillons disponibles pour l'entraînement. De plus, nous savons que la segmentation de la base de nœuds est imparfaite et que la base de feuilles n'a tout simplement pas été segmentée.

Ensuite, on remarque que les classificateurs les plus performants pour les cinq bases sont plus ou moins toujours les mêmes : SVM2, PM et SVM. De plus, les taux d'erreur de ce groupe de classificateurs se distinguent particulièrement de ceux des autres algorithmes pour les bases de céréales, de pollen et de raisins, où l'erreur de classification est faible.

La généralisation présentée par le module de PG orchestrée que nous avons mis au point se situe à mi-chemin entre celle du groupe formé des SVM2, PM et SVM et celles des KPPV, C4.5 et Bayes.

5.1.2 Première série : bases de données avec ACP

Nous comparons maintenant les résultats de la classification produits par les mêmes classificateurs qu'auparavant, mais sur les cinq bases de données transformées. Les nouvelles bases de données produites par l'ACP (avec conservation de 95% de la variance, tel que décrit à la section 4.4.3) comprennent un plus petit nombre de caractéristiques que les bases de données initiales :

Tableau 5.2

Nombre de caractéristiques des bases de données avant et après ACP

Base de données	**Nombre de caractéristiques initiales**	**Nombre de caractéristiques après ACP**
Grande base de céréales	60	14
Nœuds	60	14
Feuilles	43	13
Pollen	48	13
Raisins secs	60	14

Les résultats de la grande base de céréales avec et sans transformation par ACP seront examinés. Comme les effets de l'ACP sur le taux de classification sont relativement uniformes pour tous les problèmes, nous n'analyserons pas chaque cas individuellement, mais présenterons plutôt des observations générales. Un tableau des erreurs de classification pour la première série d'expériences est disponible à l'Annexe II.

Grande base de céréales et observations générales

Afin de faciliter la comparaison, les erreurs de classification pour la grande base de céréales avec et sans ACP sont présentées sur la même figure. Une boîte à moustache identifiée avec le nom du classificateur précédé de « ACP » indique que la base de données utilisée a été préalablement transformée par la procédure de ce nom.

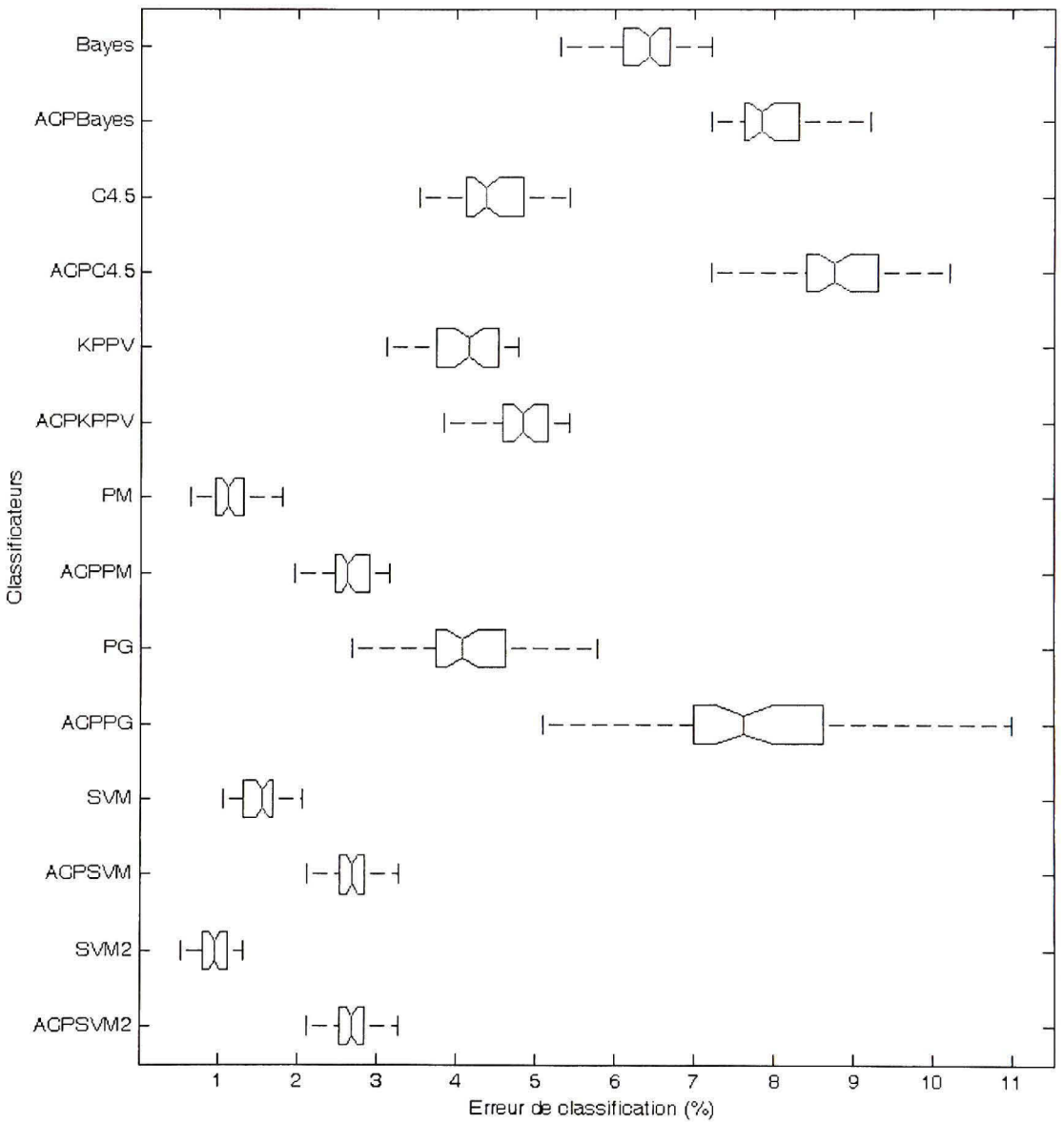


Figure 5.6 ***Erreurs de classification pour la grande base de céréales avec et sans ACP.***

Pour les sept classificateurs, l'erreur de classification médiane ainsi que trois quarts ou plus des mesures d'erreur obtenues sont plus grandes lorsque la base de céréales a été transformée par l'ACP. Le tableau suivant présente plus précisément la médiane de l'erreur pour chaque classificateur, avant et après ACP :

Tableau 5.3

Erreurs médianes pour la grande base de céréales

Classificateur	**Médiane de l'erreur sur la base initiale (1)**	**Médiane de l'erreur sur la base transformée par ACP (2)**	**Ratio $\frac{2}{1}$**
Bayes	6.42%	7.84%	1.22
C4.5	4.37%	8.76%	2.00
KPPV	4.16%	4.84%	1.16
PG	4.08%	7.63%	1.87
SVM	1.55%	2.68%	1.72
SVM2	0.95%	2.26%	2.37
PM	1.11%	2.63%	2.37

Dans le tableau 5.3, le ratio de la quatrième colonne correspond à l'erreur médiane avec ACP divisée par l'erreur médiane avec la base initiale. On peut former deux groupes de classificateurs selon le ratio présenté, interprété comme un indice (proportionnel à la médiane de l'erreur) de l'influence de l'ACP sur les performances de classification. Bayes et KPPV sont les moins affectés, dans un rapport qui dépasse légèrement 1.0. Tous les autres classificateurs présentent un ratio qui se rapproche de 2.0, indiquant un doublement approximatif de l'erreur de classification après transformation par l'ACP. Alors que l'ordre de performance des classificateurs est différent suite à l'application de l'ACP pour les Bayes, C4.5, KPPV et PG pour le problème de céréales, il reste le même pour les classificateurs produisant les plus petites erreurs, soit SVM, SVM2 et PM. Les observations issues de la figure 5.6 et du tableau 5.3 se confirment rapidement suite à l'examen des effets de l'ACP pour les autres bases de données (*Voir* Annexe II). Dans tous les cas, la transformation d'une base de données par ACP (selon les paramètres que nous avons choisis) ne permet pas à nos

algorithmes d'obtenir de meilleurs taux d'erreur de classification comparativement à ceux produits à partir de la base initiale.

5.1.3 Deuxième série : classificateurs avec boosting

Nous reprenons pour la deuxième série les sept classificateurs que nous avons utilisés pour la première. Nous ajoutons à cet ensemble les versions « boostées » de Bayes, C4.5, SVM, SVM2 et PM. Finalement, nous complétons le groupe avec une nouvelle version de la PG, celle avec boosting intégré, que nous identifions PGB. Les bases de données sans ACP de la première série sont de nouveau utilisées, sauf que la grande base de céréales est remplacée par la petite, et les bases de chiffres par ordinateur et de feuilles rognées sont ajoutées. Pour faciliter la comparaison, les résultats avec et sans boosting seront présentés sur les mêmes figures. Évidemment, les résultats des expériences déjà réalisées et présentées à la section 5.1.1 ne sont que reportés sur les graphiques. Nous examinerons plus en détail les résultats pour les bases de céréales (petite), de chiffres et de feuilles rognées. Nos observations générales pour l'ensemble des bases de données complèteront l'analyse de la deuxième série.

Petite base de céréales

La petite base de céréales est une version réduite de la grande, qui ne contient que 500 images par classe. Encore une fois, les deux tiers sont utilisés pour l'entraînement (2000 échantillons) et le tiers restant pour le test (1000 échantillons).

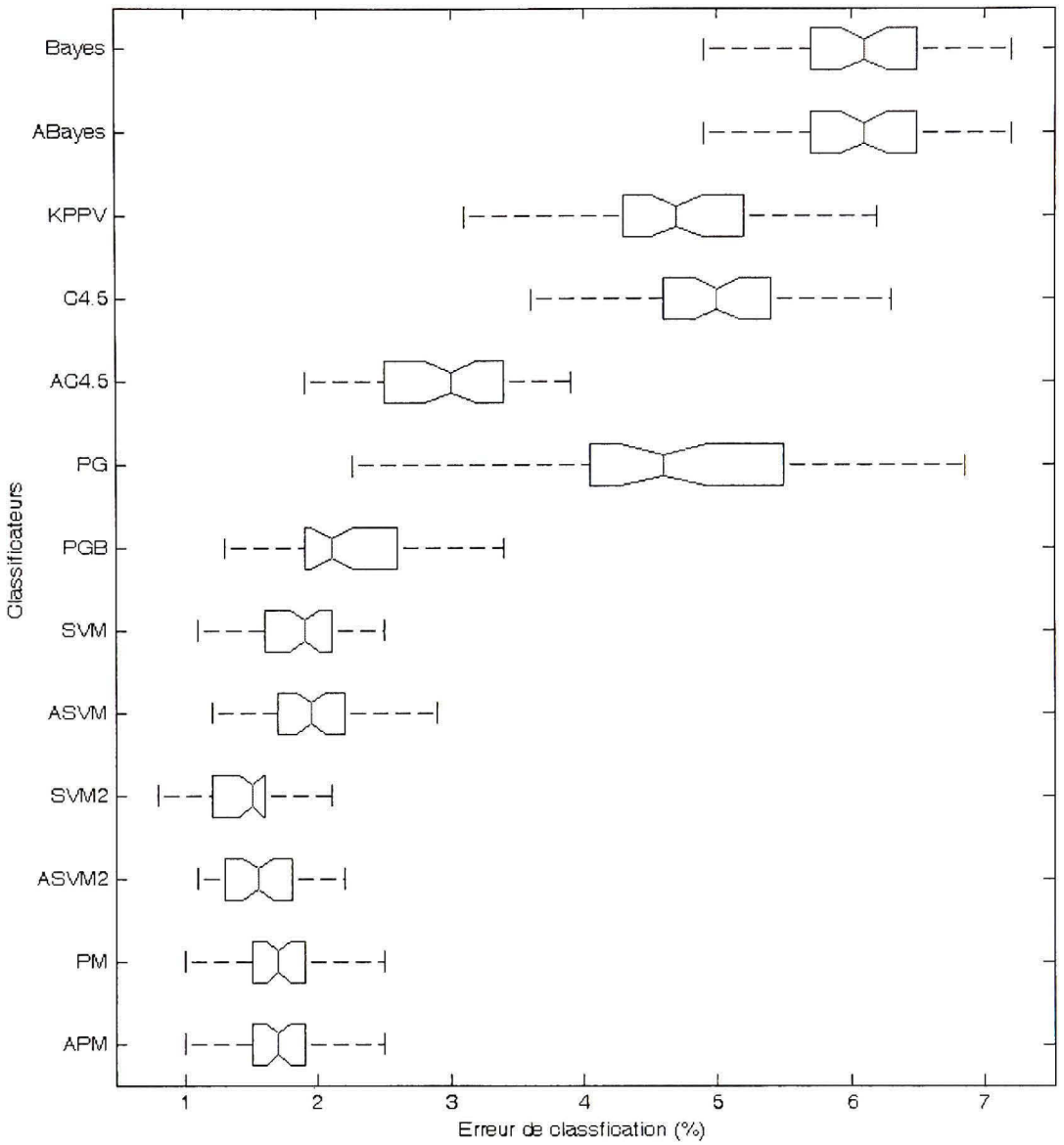


Figure 5.7 ***Erreurs de classification pour la petite base de céréales : deuxième série.***

Les résultats de classification pour la petite base de céréales ressemblent fort à ceux obtenus pour la grande base. Les médianes des erreurs varient entre 1.50% pour SVM2 à 6.10% pour Bayes. Le boosting des algorithmes semble produire deux effets différents, pour deux groupes de classificateurs. Chez Bayes, SVM, SVM2 et PM, le boosting ne semble pas avoir d'effet significatif. En effet, la médiane de l'erreur n'est pas statistiquement différente de celle obtenue sans boosting. Par contre, dans le cas du groupe formé de C4.5 et la PG, l'utilisation du méta algorithme permet une réduction significative de l'erreur médiane de

classification. Alors que C4.5 offre une médiane de 5.00%, celle d'AC4.5 est réduite à 3.00%. La PG passe de 4.61%, sous sa forme orchestrée, à 2.10% lorsque intégrée avec le boosting. La PGB se rapproche du groupe de classificateurs qui a montré les meilleures performances de généralisation pour la première série, soit les SVM et le PM. Ces derniers algorithmes ne semblent pas être en mesure de bénéficier du méta algorithme de boosting. De plus, la PGB obtient un écart interquartile plus petit que la PG orchestrée.

Base de chiffres

La base de chiffres a été produite à partir d'images contenant les chiffres zéro à neuf. 33 échantillons par classe sont disponibles, chaque chiffre étant représenté avec différents attributs (gras, italique, tailles variées). 220 échantillons sont utilisés pour l'apprentissage contre 110 pour le test.

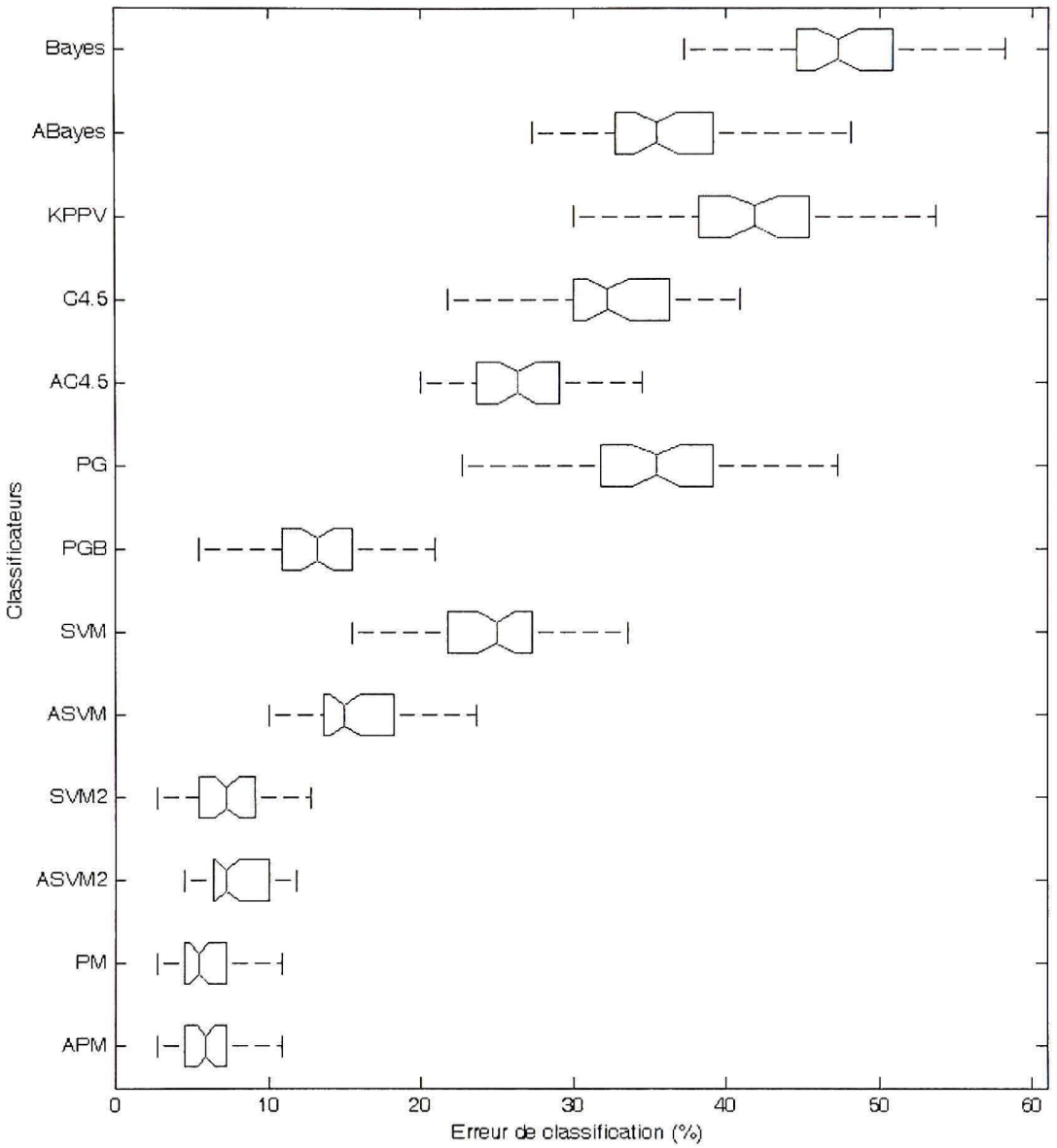


Figure 5.8 ***Erreurs de classification pour la base de chiffres : deuxième série.***

La dispersion des résultats, tous classificateurs confondus, est très grande. PM obtient la médiane de l'erreur la plus faible avec 5.45%, alors que Bayes présente une médiane approximativement neuf fois supérieure : 47.27%. Les effets du boosting sont particulièrement visibles alors que Bayes, C4.5, la PG et même le SVM réduisent significativement leurs erreurs de classification lors de l'utilisation du méta algorithme. La différence entre les médianes de l'erreur de la PG et de la PGB est particulièrement élevée, à 22.27%. Le boosting semble encore une fois ne pas avoir d'effet sur les SVM2 et PM.

Bayes, KPPV, C4.5, et la PG n'arrivent pas à obtenir un taux de classification adéquat pour ce problème. La difficulté éprouvée par ces algorithmes est en partie due au petit nombre d'échantillons disponibles pour l'apprentissage. Pourtant, l'écart particulièrement grand entre les performances de ces quatre classificateurs et celles des autres nécessite un éclaircissement. Nous expliquons cette différence par la présence de plusieurs caractéristiques qui ne fournissent pas d'information importante, individuellement, dans la base de données. En effet, nous croyons que la capacité de combiner les caractéristiques observées chez les PGB, SVM, SVM2 et PM offre un grand avantage dans la résolution de ce problème. Selon cette analyse, les algorithmes produisant les résultats les plus faibles pourraient améliorer leurs performances à l'aide de la sélection des caractéristiques les plus informatives (donc le retrait des caractéristiques qui nuisent à la classification) ou de leur transformation. De plus, il est probable que l'ajout de caractéristiques décrivant plus précisément la forme de l'objet puisse augmenter le pouvoir de généralisation des classificateurs.

Base de feuilles rognées

Nous avons produit la base de feuilles rognées à partir de la base de feuilles, au moyen d'une segmentation partielle par rognage (réalisée par un opérateur humain). Cette transformation augmente la proportion de l'image qui représente l'objet à classifier, passant de 15% à environ 40% après rognage. La nouvelle base est constituée du même nombre d'échantillons que sa source, soit 60 échantillons par classe pour un total de 180.

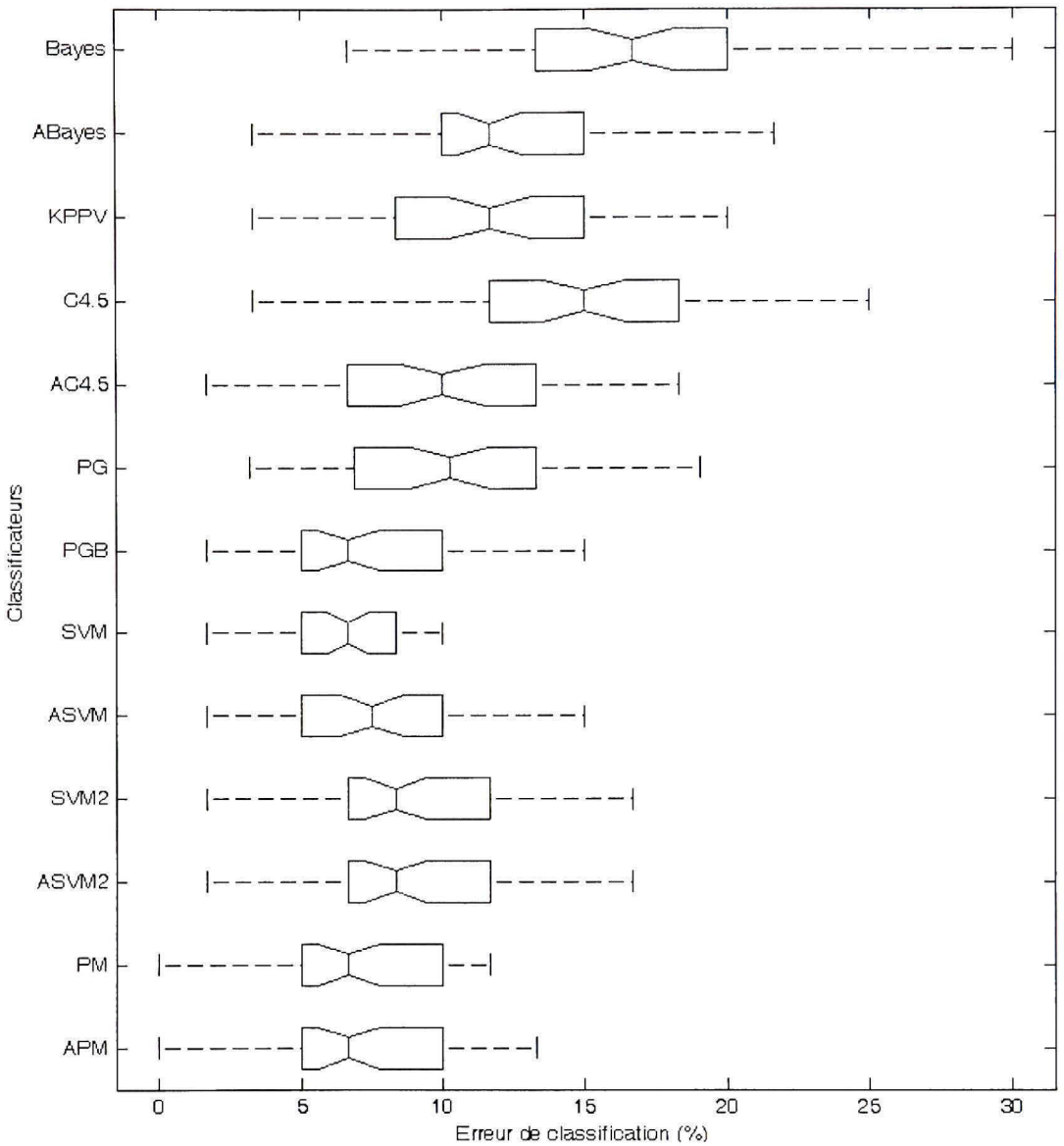


Figure 5.9 ***Erreurs de classification pour la base de feuilles rognées : deuxième série.***

Nous remarquons à la figure 5.9 à peu près le même ordre des classificateurs que celui qui a été observé pour la base de feuilles (*Voir* figure 5.5). Les combinaisons de classificateurs par boosting de Bayes, C4.5 et la PG produisent respectivement des médianes d'erreurs plus faibles que leur version originale. SVM, SVM2 et PM classifient aussi bien avec ou sans boosting. En comparant aux performances obtenues pour la base de feuille sans rognage, on observe une baisse généralisée des taux d'erreurs accompagnée d'une réduction de la

dispersion des erreurs. Le tableau 5.4 permet d'apprécier les différences entre les médianes de l'erreur avec et sans le rognage des images de feuilles.

Tableau 5.4

Médiane de l'erreur de classification pour les bases de feuilles et de feuilles rognées

Classificateur	**Médiane de l'erreur Base de feuilles**	**Médiane de l'erreur Base de feuilles rognées**
Bayes	43.33	16.67
ABayes	41.67	11.67
KPPV	38.33	11.67
C4.5	41.67	15.00
AC4.5	38.33	10.00
PG	33.33	10.76
PGB	31.67	6.67
SVM	33.33	6.67
SVM2	26.67	8.33
PM	31.67	6.67

Ces résultats illustrent à quel point il est plus facile de procéder à la classification lorsque la segmentation, même partielle, est incorporée au processus. Même si la feuille correspond à moins de la moitié de l'image, dans le cas du rognage, la classification faite à partir de l'extraction de caractéristiques de cette image obtient une médiane de l'erreur qui peut atteindre 6.67%, pour les PGB, SVM et PM. Toutefois, les taux d'erreurs varient énormément au long des 50 réplications, passant parfois de 0% à 12% d'erreur (PM). Nous avançons que le faible nombre d'échantillons de la base de données d'apprentissage est une cause de cette grande dispersion.

Observations générales

L'étude des résultats de classification pour les problèmes de céréales, chiffres et feuilles rognées, puis pour les autres problèmes de la deuxième série (*Voir* Annexe III) nous permet de constater que certains classificateurs sont plus « réceptifs » au boosting. En premier lieu, C4.5 et la PG, pour tous les problèmes, diminuent leur taux d'erreur respectif au moyen du

boosting. Cette amélioration des performances est considérable et toujours significative, sauf pour le problème de raisins secs, où il est très difficile d'améliorer la classification (la PG et la PGB obtiennent une médiane de l'erreur de seulement 0.44%). Ensuite, il semble que Bayes puisse profiter de l'effet généralisant du boosting dans la majorité des cas. Dans deux cas, lorsque le nombre d'échantillons est élevé (céréales et pollen), le boosting de Bayes n'apporte pas de contribution significative. Finalement, il semble que les SVM, SVM2 et PM soient déjà trop optimisés pour être combinés par boosting. Cette observation supporte l'hypothèse que le boosting permet d'améliorer le taux de reconnaissance de classificateurs « faibles », tandis qu'il ne permet pas d'améliorer celui des classificateurs « forts » (Freund, 1990).

L'algorithme de PGB présente des résultats particulièrement intéressants. Malgré le peu de recherche réalisée pour ajuster ses paramètres, les taux de classification observés se rapprochent des classificateurs les plus performants pour les problèmes étudiés, soit les SVM et le PM.

5.2 Temps d'apprentissage

Nos observations concernant le temps d'apprentissage des classificateurs pour la première série seront présentées. Ensuite, nous illustrerons les résultats temporels de la deuxième série à l'aide d'un graphique des temps de classification pour la petite base de céréales. Nous compléterons notre analyse avec des observations générales.

5.2.1 Première série : temps d'apprentissage avec et sans ACP

Un tableau récapitulatif de tous les temps d'apprentissage pour la première série est présenté en Annexe IV. Afin de mieux cerner l'effet de l'ACP sur le temps d'apprentissage des différents types de classificateurs, nous avons extrait quelques données et préparé le tableau suivant :

Tableau 5.5

Temps d'apprentissage médian pour cinq classificateurs, en 10^{-3} secondes par échantillon

Base / Classificateur	Bayes	C4.5	PG	SVM2	PM
Céréales (grande)	0.05	0.49	6656.21	1.07	254.59
Céréales (grande) avec ACP	0.02	0.21	7434.08	0.77	26.50
Feuilles	0.25	0.75	5916.04	1.33	125.33
Feuilles avec ACP	0.17	0.50	5257.38	1.00	17.33
Raisins secs	0.07	0.26	2059.12	0.29	227.62
Raisins secs avec ACP	0.03	0.11	2487.31	0.27	17.92

On observe trois effets différents de l'ACP sur le temps d'apprentissage. Le PM semble profiter énormément de la réduction de dimensionnalité, ses temps d'apprentissage étant généralement réduits au cinquième du temps nécessaire avec la base de données originale. Les Bayes (puis KPPV), C4.5 et SVM profitent aussi de la réduction du nombre de caractéristiques, mais dans une moindre mesure : de l'ordre de 40%. Finalement, le temps d'apprentissage de la PG n'est généralement pas réduit lorsque l'ACP est utilisée pour transformer la base de données. Cela s'explique par les critères d'arrêt de la PG, relatifs au taux de reconnaissance, au temps écoulé et au nombre de générations. En effet, si le premier critère n'est pas atteint, c'est le temps qui détermine la fin de l'évolution de la PG.

5.2.2 Deuxième série : temps d'apprentissage avec et sans boosting

Rappelons que la petite base de céréales contient 500 échantillons par classe, pour un total de 3000. Les taux d'erreur de classification pour ce problème sont présentés à la figure 5.7. Nous avons encore un fois utilisé cette base pour illustrer les résultats de temps d'apprentissage pour la deuxième série :

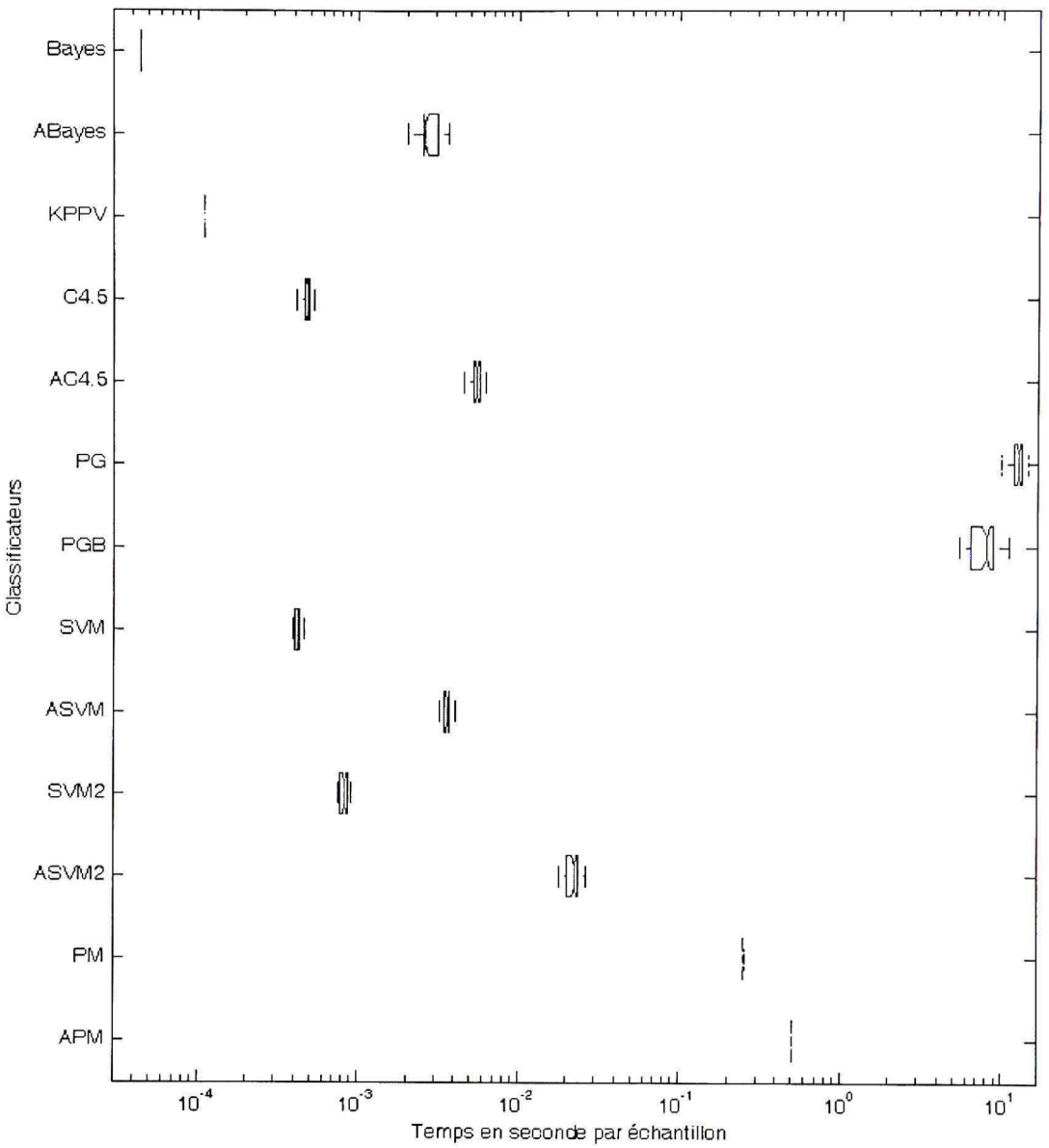


Figure 5.10 ***Temps d'apprentissage pour la petite base de céréales.***

La figure 5.10 présente un graphique dont l'échelle de temps (sur l'axe horizontal) est logarithmique de base 10. Notre première observation est l'extrême dispersion des temps d'apprentissage. Le classificateur qui apprend le plus rapidement est plus de 10^5 fois plus rapide que le classificateur qui met le plus de temps pour l'apprentissage. Dans l'ordre, de la plus petite médiane du temps à la plus grande, on compte Bayes (4.0×10^{-5} s/éch.), KPPV (1.1×10^{-4} s/éch.), SVM (4.2×10^{-4} s/éch.), C4.5 (4.8×10^{-4} s/éch.), SVM2 (8.4×10^{-4} s/éch.), PM (2.5×10^{-1} s/éch.) et PG (12.86 s/éch.). Le boosting augmente le temps d'apprentissage

d'un facteur qui varie entre 10 et 30, dans la plupart des cas. Deux classificateurs, le PM et la PG réagissent différemment. Premièrement, le PM, combiné par boosting, augmente son temps de classification médian d'un facteur d'environ 3. Le PM obtient souvent un taux de classification de 100% à l'apprentissage. Cette situation arrête prématurément l'algorithme de boosting car il est impossible de poursuivre à l'étape développement d'un nouveau classificateur "spécialisé" pour la reconnaissance des échantillons problématiques s'il n'y en a pas (*Voir* le pseudo code à la section 2.7.2). Finalement, nous observons que la PGB nécessite un temps d'apprentissage plus court que la PG orchestrée. Ce gain temporel est principalement dû à deux facteurs : la PGB recherche des programmes moins performants (le critère pour l'adéquation est fixé à 0.9 – *Voir* tableau 4.6 de la section 4.7) et elle conserve la population lors de la recherche d'un nouveau programme pour la même classe (*Voir* pseudo code à la section 3.5.1).

Le tableau 6.5 de l'annexe V présente tous les résultats de temps d'apprentissage de la deuxième série. Les observations tirées de la figure 6.5 se répètent pour les autres problèmes, à quelques différences près. Les gains temporels pour la PGB par rapport à la PG orchestrée sont particulièrement visibles avec de plus petites bases de données comme celles de chiffres, de nœuds et de feuilles rognées. Dans ces cas, le temps d'apprentissage diminue d'un facteur d'environ huit. Même avec ces temps d'apprentissage plus rapides, la PGB reste plus lente en apprentissage que son plus proche concurrent, le PM. Cette différence est particulièrement grande lorsque la base de données contient beaucoup d'échantillons.

5.3 Discussion sur les résultats

Dans cette section nous observerons les résultats dans leur ensemble. En premier lieu, nous étudierons les effets de l'ACP, ensuite nous ferons une analyse par problème (base de données), puis nous terminerons avec les résultats par classificateur.

Effets de l'ACP

Nos expériences avec l'ACP ne nous permettent pas d'observer l'avantage qu'elle pourrait offrir. Puisque les caractéristiques que nous avons extraites support ent déjà une grande discrimination des classes, nous pensons que le paramètre de conservation de la variance utilisé (conservation de 95% de la variance) a été fixé à une valeur trop petite. Il est probable qu'avec une conservation de 96% à 99% de la variance, l'ACP puisse maintenir le taux de bonne classification des algorithmes en aval, tout en réduisant leur temps d'apprentissage (puisqu'il réduit le nombre de caractéristiques).

Résultats de classification par problème

Les taux d'erreur des classificateurs semblent extrêmement liés aux problèmes particuliers présentés par une base de données. En effet, les taux de reconnaissance qu'il est possible d'atteindre sont contraints par l'information contenue dans la base de données qui contribue à fournir un portrait général des classes. Il semble que le choix d'un classificateur soit, dans une certaine mesure, d'une importance beaucoup moins grande que la qualité et la taille de la base de données. Le tableau 5.6 présente les meilleurs résultats pour toutes les bases de données utilisées et les caractéristiques qui rendent difficile la création d'un classificateur généralisant. La colonne *Erreur* donne la meilleure médiane de l'erreur obtenue par un classificateur pour la base en question. L'algorithme ayant obtenu ce résultat est identifié sous la colonne *Algo*.

Tableau 5.6

Meilleur classificateur, difficulté(s) et solution(s) proposée(s) par problème

Base de données	**Erreur**	**Algo**	**Éch. / classe**	**Difficulté(s)**	**Solution(s) proposée(s)**
Céréales	1.50%	SVM2	950 ou 500	Faible confusion sur 4 classes	Extraction de plus de descripteurs
Raisins	0.22%	SVM	450	Aucune	-
Pollen	4.15%	SVM2	195	Faible confusion sur 3 classes	Images en couleur ou extraction de plus de descripteurs
Feuilles	26.67%	SVM2	60	Absence de segmentation, peu d'échantillons	Circonscrire les données importantes : segmentation
Feuilles rognées	6.67%	PGB, SVM, PM (égaux)	60	Peu d'échantillons, segmentation partielle	Plus d'échantillons
Chiffres par ordinateur	5.45%	PM	33	Peu d'échantillons	Plus d'échantillons
Nœuds de bois	20.37%	PM	27	Peu d'échantillons, erreurs dans la base	Plus d'échantillons, éliminer les erreurs de la base

Les meilleurs résultats obtenus, pour les bases de raisins secs, de céréales et de pollen par exemple, sont un bon indice que les caractéristiques que nous avons extraites des images fournissent suffisamment d'information pour la classification d'objets biologiques. Les autres résultats, présentant un plus haut taux d'erreur de classification, s'expliquent soit par une trop petite quantité d'échantillons d'apprentissage, soit par une difficulté inhérente au problème (erreurs dans la base de données ou segmentation incomplète). Dans le cas d'une base de données contenant des caractéristiques pertinentes, le nombre d'échantillons est un facteur déterminant pour le taux de reconnaissance qu'un algorithme peut obtenir à la suite de l'étude des échantillons.

Les résultats pour la base de feuilles rognées sont encourageants et indiquent qu'il est possible d'obtenir de bons taux de reconnaissance pour des objets biologiques dans une image avec peu ou pas de segmentation.

Résultats par classificateur

Certains classificateurs se sont démarqués par leur performance de généralisation ou pour leur court temps d'apprentissage. Le tableau 5.7 montre notre évaluation de chaque classificateur (ASVM2 et APM ont été omis car ils sont pratiquement toujours semblables à leur version sans boosting) pour les problèmes traités au cours de cette étude.

Tableau 5.7

Évaluation de chaque classificateur utilisé

Classificateur	Taux d'erreur de classification	Temps d'apprentissage
Bayes	C-	A+
ABayes	C-	C+
KPPV	C	A
C4.5	C-	A-
AC4.5	B-	C+
PG	B-	D
PGB	A-	D+
SVM	A-	B+
ASVM	B+	C+
SVM2	A	B
PM	A+	C-

Les scores du tableau 5.7 sont basés sur l'ordre de performance de chaque classificateur pour les bases de céréales (petite), chiffres, nœuds de bois, feuilles, feuilles rognées, pollen et raisins secs. Si les médianes de deux classificateurs ne sont pas significativement différentes (selon l'analyse par boîte à moustache), leur rangs sont considérés similaires. Les résultats représentent la moyenne du rang d'un classificateur pour tous les problèmes (1^{er} = A+, $2^{ième}$ = A, etc., puis $11^{ième}$ rang = D). La méthode d'évaluation des notes n'étant pas basée sur des critères quantitatifs, les recommandations à tirer du tableau 5.7 ne s'appliquent que dans les

cas les plus généraux, où il n'y a pas de critères quant à la performance de taux de classification ou de temps d'apprentissage désirée.

À l'aide du tableau 5.7, nous pouvons de nouveau observer que le boosting de Bayes et du SVM ne font que dégrader les résultats, que ce soit pour le taux de classification ou pour le temps d'apprentissage, en comparaison avec les mêmes méthodes sans boosting. Globalement, les algorithmes les plus rapides en apprentissage (Bayes, KPPV et C4.5) obtiennent des taux d'erreur de classification plus élevés que les autres algorithmes utilisés. Les meilleures combinaisons pour le taux d'erreur de classification et le temps d'apprentissage sont SVM, SVM2 et PM. PM maintient des taux d'erreur plus réguliers que les SVM, qui eux démontrent une plus grande variance pour les différentes bases de données.

CONCLUSION

Dans ce mémoire, nous avons étudié les étapes menant à la classification automatisée d'objets dans des images numériques. Nous nous sommes particulièrement attardés au processus d'extraction des caractéristiques et aux algorithmes de classification. En pratique, nous avons mis sur pied un système de segmentation, d'extraction de caractéristiques et de classification à l'aide de Matlab et de Weka, en plus d'avoir développé un algorithme de PG modulaire pour Weka. Finalement, nous avons comparé les résultats de classification pour deux processus d'extraction des caractéristiques, avec six classificateurs et un méta algorithme, pour la reconnaissance d'objets présentés dans six bases de données d'images dont trois ont été créées par nous.

L'extraction de 40 caractéristiques à partir des images permet la classification automatisée d'objets de différentes classes. La transformation des caractéristiques par ACP influence le taux de classification des algorithmes en aval. En effet, si l'ACP à 95% de conservation de variance permet de réduire le temps d'apprentissage de la plupart des classificateurs (à l'exception de la PG), elle entraîne également une augmentation considérable du taux d'erreur de classification, en comparaison avec l'extraction de caractéristiques sans transformation. Cette observation nous a découragé d'utiliser l'ACP pour la suite de notre analyse.

L'étude des résultats de classification révèle que certaines bases de données, en particulier celles qui contiennent peu d'échantillons pour l'apprentissage, ne permettent pas d'obtenir des taux de classification suffisants à l'aide des algorithmes que nous avons utilisés. La segmentation pauvre ou inexistante, dans certains cas, rend le problème encore plus corsé. Néanmoins, l'ordre des classificateurs, selon leur taux d'erreur médian, est consistant pour la plupart des problèmes rencontrés. Dans le cas des bases de données étudiées, le PM et les deux variantes de SVM obtiennent généralement les meilleurs taux de classification. Ces taux sont particulièrement élevés pour les bases de données contenant un grand nombre

d'échantillons. Les meilleurs temps d'apprentissage sont produits par des algorithmes moins performants en classification, soit Bayes, C4.5 et KPPV.

Les résultats obtenus lors de nos expérimentations présentent une variabilité qui semble reliée au type de problème rencontré, défini par les caractéristiques et leur variance parmi la base de données. Nos expérimentations ne nous permettent pas d'émettre une recommandation fiable sur l'utilisation prioritaire d'un classificateur dans tous les cas. Cependant, nous croyons que l'utilisation d'une heuristique pour l'analyse des données ou l'essai de plusieurs classificateurs puisse offrir une piste intéressante pour le choix d'un algorithme, en autant que l'on dispose d'une base de données d'apprentissage suffisamment grande.

L'étude de la classification d'objets non segmentés pour la base de données de feuilles révèle un certain potentiel, en particulier si l'objet à identifier occupe une bonne portion de l'image de laquelle les caractéristiques sont extraites et ce, même si le fond de l'image varie considérablement. Étant donné le peu d'échantillons de la base de données et le faible nombre d'expériences réalisées, nous aimerions réaliser de nouvelles expérimentations pour confirmer et préciser les performances pouvant être atteintes par cette méthode.

Le méta algorithme de boosting montre son utilité lorsqu'il combine des programmes en arbres comme ceux produits par l'algorithme C4.5 et la PG. La combinaison de programmes permet, pour ces classificateurs, de réduire considérablement le taux d'erreur obtenu pour tous les problèmes étudiés. Cette qualité n'est généralement pas observée dans le cas des autres classificateurs utilisés.

Finalement, l'intégration du boosting à la PG aura offert un double avantage par rapport à la PG orchestrée : non seulement les taux d'erreurs de reconnaissance ont diminué, mais le temps d'apprentissage a aussi chuté. La PGB semble avoir le potentiel de se hisser au rang des SVM et PM, à condition d'optimiser ses paramètres.

Nous estimons que les taux de reconnaissance obtenus par notre système, pour des bases de données variées, sont suffisamment élevés pour susciter l'intérêt et démontrer la possibilité de développer un algorithme de reconnaissance de formes dans des images qui soit générique et qui nécessite peu d'intervention humaine. Nous sommes d'avis qu'un tel système, faisant l'extraction de multiples caractéristiques à partir d'images segmentées et ne préférant pas l'usage d'un classificateur en particulier, pourrait obtenir des taux de classification suffisants, supérieurs à la classification humaine et, à l'occasion, supérieurs à la classification produite par un algorithme ajusté par un expert. Pour le démontrer, il nous faudrait comparer nos résultats avec ceux de l'humain et de l'expert. Nous proposons de réaliser ces expériences dans un travail subséquent.

Nous nous proposons de continuer ces recherches parallèles à propos de la PGB et du processus de reconnaissance dans son ensemble (segmentation, extraction et classification) d'une façon libre, ouverte et sur la base de la collaboration. Selon cette idée, les recommandations qui suivent pourraient être appliquées par un réseau composé de programmeurs et de chercheurs. Ces réseaux « computationnels » humains sont selon nous le meilleur outil pour la résolution de problèmes que les machines n'arrivent pas tout à fait à solutionner pour le moment, faute d'exemples d'apprentissage adéquatement mis en forme.

RECOMMANDATIONS

Nous présentons des recommandations concernant trois portions de notre travail. En premier lieu, nous suggérons des avenues en vue du développement d'un outil automatique et générique pour la reconnaissance de formes dans des images. Ensuite, nous discutons de nouvelles expérimentations à réaliser pour étoffer l'analyse présentée dans ce travail. Finalement, nous proposons des améliorations ainsi que de nouvelles options à apporter à l'algorithme de PG développé pour Weka.

Outil automatique et générique pour la reconnaissance de formes dans des images

Nos conclusions mènent à trois recommandations majeures. Tout d'abord, nous avons constaté qu'il est extrêmement difficile, voire impossible, de certifier qu'un classificateur sera meilleur qu'un autre dans toutes les situations. Ayant fait cette constatation, nous suggérons plusieurs façons de procéder en vue d'obtenir le meilleur taux de classification pour un problème :

- Comparer les résultats de plusieurs classificateurs à l'aide d'une base d'apprentissage et d'une base de test et sélectionner le plus prometteur;
- Combiner plusieurs classificateurs à l'aide de méta algorithmes comme par exemple un comité d'experts ou un arbre neuronal (Jain, Duin et Mao, 2000);
- Automatiser la sélection ou la combinaison d'algorithmes à utiliser en fonction des données au moyen d'une métaheuristique évolutive.

Des recherches suggèrent que la performance des classificateurs dépend de la distribution des données du problème (van der Walt et Barnard, 2006). Nous sommes donc d'avis que l'utilisation d'une méta heuristique évolutive, qui « apprend » à choisir ou à combiner les bons classificateurs à partir des données (à l'aide d'exemples de problèmes) soit une avenue prometteuse. C'est d'ailleurs la voie que proposent d'autres groupes de chercheurs indépendants (Gagné et al., 2007) (Chandra et Yao, 2006).

En second lieu, nous croyons qu'il serait bénéfique d'optimiser les paramètres de certains algorithmes, par exemple le K du KPPV, le nombre de neurones de la couche cachée du PM et le choix de la fonction de noyau du SVM, afin d'obtenir la meilleure performance possible pour chacun d'eux. Considérant la rapidité d'apprentissage de la plupart de ces classificateurs, le coût computationnel de l'optimisation des paramètres est souvent faible.

Finalement, nous rappelons que la pertinence des données fournies à un système de reconnaissance détermine la performance de reconnaissance maximale que ce dernier pourra obtenir. Ainsi, il est primordial de disposer de caractéristiques qui mettent en valeur les différences entre les classes d'un problème de classification. Bien que les 40 caractéristiques utilisées dans ce projet constituent une bonne base pour un éventail de problèmes de reconnaissance visuelle, nous croyons que l'addition d'autres caractéristiques, par exemple l'extraction à l'aide de filtres de Gabor (Kong, Zhang et Li, 2003), nous permettraient d'atteindre une plus grande généralisation en terme de domaine d'utilisation. En présence d'un grand nombre de caractéristiques, il devient important d'éliminer celles qui ne présentent pas d'information utile (ayant une trop grande ou trop petite variance) : ceci peut être réalisé automatiquement au moyen de l'algorithme « attributeSelection » de Weka.

Poursuite des expérimentations

De façon générale, nos analyses et conclusions seraient consolidées par de nouvelles expérimentations avec un plus grand nombre de bases de données concernant des domaines différents. Des problèmes de reconnaissance d'écriture manuscrite, de visages, d'iris, d'empreintes ou d'anomalies biologiques pourraient être abordés afin de mieux tester la performance de généralisation des algorithmes d'extraction et de classification.

Nos expériences concernant la transformation des caractéristiques par ACP ne sont pas concluantes. Il apparaît que le choix d'un autre taux de conservation de la variance pourrait offrir un avantage à la fois au niveau du taux de reconnaissance et du temps d'apprentissage. En conservant les mêmes 40 caractéristiques de base, nous suggérons de faire de nouvelles expériences avec des taux variant entre 96.0% et 99.0% de conservation de la variance.

Les temps de classification, incluant le temps nécessaire à l'extraction de chacune des caractéristiques utilisées, pourraient être évalués, pour chaque classificateur. Ce faisant, nous pourrions produire des recommandations à propos du choix d'un classificateur dans le cas où une limite de temps est imposée pour le traitement de chaque échantillon. Prendre en compte le temps d'extraction est important, en particulier dans le cas de classificateurs qui n'utilisent pas nécessairement toutes les caractéristiques fournies en entrée.

Algorithme de PG pour Weka

L'algorithme développé pour Weka a pour but de permettre la recherche coopérative afin d'améliorer les performances de classification de la PG. Nous proposons ici des variantes qui pourraient inciter les chercheurs de la PG à utiliser notre algorithme pour leur recherche respective.

En premier lieu, nous croyons important de permettre l'utilisation d'autres structures pour les programmes générés par la PG. Nous proposons donc d'intégrer les structures en graphe (Niehaus, Igel et Banzhaf, 2007) et linéaire (Brameier et Banzhaf, 2007), deux options populaires dans la recherche actuelle en PG.

Ensuite, il est important de faciliter l'utilisation de réseaux d'ordinateurs pour la computation parallèle de la PG (Weise, 2008), lorsque le nombre de données à traiter est important. De plus, nous aimerions incorporer des méthodes de traitement de populations plus ou moins isolées comme le modèle de l'île à espèce (Gustafson et Burke, 2006).

Nous croyons primordial d'améliorer l'efficacité et la rapidité du code utilisé pour le boosting, pour le calcul de la fitness et pour la sélection des programmes reproducteurs car ces opérations occupent la plus grande part de temps d'utilisation du processeur lors de la phase d'apprentissage de la PG. Certaines méthodes, comme les unités de traitement graphiques (Harding et Banzhaf, 2007), pourraient être implémentées afin d'obtenir encore plus d'efficacité computationnelle.

Nous désirons aussi ajouter une option pour le traitement des données bruitées par la méthode de relaxation (Da Costa, Landry et Levasseur, 2007), ce qui nous permettra de poursuivre les recherches entamées par des collègues du LIVIA. Dans le même ordre d'idée, il serait utile de permettre à Weka d'utiliser la PG pour construire des caractéristiques, selon le principe de co-évolution présenté par un autre collègue (Bourgouin, 2007).

Nous pensons que la performance de classification de la PGB pourrait être considérablement augmentée à l'aide de la détermination automatique des paramètres génétiques en fonction des données d'apprentissage. Le nombre de programmes à utiliser pour le boosting pourrait lui aussi être optimisé en fonction des données.

Nous croyons que la recherche sur la PG pourrait bénéficier de plus de visibilité si un algorithme de PG était disponible pour le logiciel libre RapidMiner (Mierswa et al., 2006). Ce logiciel offre une plus grande bibliothèque d'algorithmes que Weka, est plus modulaire, et semble dépasser Weka en popularité depuis quelque temps. Heureusement, RapidMiner supporte les algorithmes développés pour Weka (de plus, les deux logiciels utilisent le langage Java). Nous proposons donc de rendre notre algorithme disponible pour ce logiciel.

Finalement, nous suggérons une dernière option. Il s'agit d'ajouter un critère de temps d'exécution pour les programmes, en tenant compte du temps d'extraction des caractéristiques. Cette option permettrait de construire un classificateur qui respecte une limite de temps pour la classification de chaque objet. Cette contrainte est importante, par exemple, pour l'inspection en temps réel sur une chaîne de production.

ANNEXE I

EXPÉRIMENTATIONS

Tableau 6.1

Expérimentations de la première et de la deuxième série

BdD / Classificateur	Bayes	C4.5	KPPV	PM	PG	SVM	SVM2	ABayes	AC4.5	APM	PGB	ASVM	ASVM2
Céréale (grande) ACP	1	1	1	1	1	1	1	x	x	x	x	x	x
Nœuds ACP	1	1	1	1	1	1	1	x	x	x	x	x	x
Feuilles ACP	1	1	1	1	1	1	1	x	x	x	x	x	x
Pollen ACP	1	1	1	1	1	1	1	x	x	x	x	x	x
Raisins ACP	1	1	1	1	1	1	1	x	x	x	x	x	x
Céréales (grande)	1	1	1	1	1	1	1	x	x	x	x	x	x
Nœuds	1	1	1	1	1	1	1	2	2	2	2	2	2
Feuilles	1	1	1	1	1	1	1	2	2	2	2	2	2
Pollen	1	1	1	1	1	1	1	2	2	2	2	2	2
Raisins	1	1	1	1	1	1	1	2	2	2	2	2	2
Céréales (petite)	2	2	2	2	2	2	2	2	2	2	2	2	2
Feuilles rognées	2	2	2	2	2	2	2	2	2	2	2	2	2
Chiffres	2	2	2	2	2	2	2	2	2	2	2	2	2

Dans le tableau 6.1, la première colonne identifiée *BdD* indique le nom des bases de données, puis la première ligne identifiée *Classificateur* utilise les identifiants des classificateurs. Les identifiants sont présentées dans les tableaux *4.3* et *4.5*. Le chiffre *1* indique la première série d'expérimentation, *2* la deuxième série et les *x* indiquent que ces expérimentations n'ont pas été réalisées.

ANNEXE II

ERREURS DE CLASSIFICATION : PREMIÈRE SÉRIE

Tableau 6.2

Pourcentages d'erreurs de classification de la première série avec et sans ACP

BdD / Classificateur	S	Bayes	KPPV	C4.5	PG	SVM	SVM2	PM
Céréales (grande)	M	6.42	4.16	4.37	4.08	1.55	0.95	1.11
	$\bar{x}$	6.39	4.13	4.44	4.37	1.53	0.95	1.13
	$\hat{\sigma}$	0.53	0.43	0.47	0.98	0.26	0.19	0.26
Céréales (grande) avec ACP	M	7.84	4.84	8.76	7.63	2.68	2.26	2.63
	$\bar{x}$	7.96	4.83	8.84	7.92	2.72	2.30	2.66
	$\hat{\sigma}$	0.56	0.42	0.63	1.33	0.31	0.30	0.37
Nœuds	M	24.07	31.48	27.78	25.93	22.22	25.93	20.37
	$\bar{x}$	24.30	30.78	29.52	27.07	22.11	25.11	21.48
	$\hat{\sigma}$	4.93	5.61	7.86	5.87	4.68	5.08	5.12
Nœuds avec ACP	M	27.78	38.89	37.04	35.19	31.48	29.63	31.48
	$\bar{x}$	29.22	37.74	37.44	34.48	30.67	28.89	32.22
	$\hat{\sigma}$	5.69	5.83	5.75	5.55	5.58	4.60	4.73
Feuilles	M	43.33	38.33	41.67	33.33	33.33	26.67	31.67
	$\bar{x}$	43.20	38.87	40.90	33.07	33.70	28.50	30.77
	$\hat{\sigma}$	6.61	4.61	6.46	5.66	5.45	5.00	6.33
Feuilles avec ACP	M	45.83	38.33	48.33	41.67	37.50	35.00	36.67
	$\bar{x}$	47.43	39.70	48.20	42.17	37.90	35.70	37.27
	$\hat{\sigma}$	6.65	5.38	6.47	5.99	4.55	4.52	5.50
Pollen	M	9.72	9.28	9.06	8.73	5.46	4.15	4.37
	$\bar{x}$	9.78	9.18	8.78	8.82	5.37	4.11	4.30
	$\hat{\sigma}$	1.45	1.24	1.37	1.44	1.05	0.71	0.66
Pollen avec ACP	M	9.83	11.79	14.63	12.23	8.30	6.77	7.42
	$\bar{x}$	9.93	11.87	14.56	12.16	8.38	6.85	7.51
	$\hat{\sigma}$	1.17	0.99	1.67	1.62	0.97	0.95	0.95
Raisins secs	M	0.89	0.67	1.56	0.44	0.22	0.44	0.44
	$\bar{x}$	0.92	0.60	1.60	0.56	0.30	0.46	0.36
	$\hat{\sigma}$	0.41	0.33	0.52	0.39	0.19	0.28	0.24
Raisins secs avec ACP	M	1.56	1.44	2.00	1.33	0.44	0.44	0.44
	$\bar{x}$	1.67	1.49	1.88	1.34	0.48	0.52	0.42
	$\hat{\sigma}$	0.52	0.55	0.59	0.45	0.25	0.27	0.29

La colonne S indique laquelle des trois valeurs statistiques (soit M pour Médiane, $\bar{x}$ pour moyenne et $\hat{\sigma}$ pour écart type) est utilisée pour la ligne correspondante.

ANNEXE III

ERREURS DE CLASSIFICATION : DEUXIÈME SÉRIE

Tableau 6.3

Pourcentages d'erreur de classification pour la deuxième série avec et sans boosting

DdB / C	S	Bayes	ABayes	KPPV	C4.5	AC4.5	PG	PGB	SVM	ASVM	SVM2	ASVM2	PM	APM
Céréales (petite)	M	6.10	6.10	4.70	5.00	3.00	4.61	2.10	1.90	1.95	1.50	1.55	1.70	1.70
	$\bar{x}$	6.12	6.12	4.72	5.05	2.93	4.72	2.23	1.88	1.94	1.46	1.56	1.67	1.67
	$\hat{\sigma}$	0.58	0.58	0.70	0.66	0.53	1.06	0.48	0.36	0.37	0.37	0.32	0.32	0.32
Chiffre	M	47.27	35.45	41.82	32.27	26.36	35.45	13.18	25.00	15.00	7.27	7.27	5.45	5.91
	$\bar{x}$	47.36	36.16	41.51	32.82	26.91	35.27	12.91	24.91	15.96	7.18	7.71	5.98	6.18
	$\hat{\sigma}$	4.92	5.01	4.87	4.10	3.56	5.32	4.11	4.10	3.44	2.35	2.16	2.00	2.19
Nœuds	M	24.07	27.78	31.48	27.78	22.22	25.93	24.07	22.22	25.93	25.93	25.93	20.37	22.22
	$\bar{x}$	24.30	26.00	30.78	29.52	22.59	27.07	25.74	22.11	25.30	25.11	25.00	21.48	22.26
	$\hat{\sigma}$	4.93	5.02	5.61	7.86	6.17	5.87	7.09	4.68	4.96	5.08	5.11	5.12	5.30
Feuilles	M	43.33	41.67	38.33	41.67	38.33	33.33	31.67	33.33	31.67	26.67	31.67	31.67	31.67
	$\bar{x}$	43.20	42.33	38.87	40.90	37.97	33.07	31.53	33.70	32.73	28.50	30.80	30.77	31.53
	$\hat{\sigma}$	6.61	6.19	4.61	6.46	5.54	5.66	5.06	5.45	4.76	5.00	4.63	6.33	5.34
Feuilles rognées	M	16.67	11.67	11.67	15.00	10.00	10.26	6.67	6.67	7.50	8.33	8.33	6.67	6.67
	$\bar{x}$	16.23	12.27	12.13	15.27	10.23	10.23	7.90	7.03	7.77	9.10	9.10	6.70	6.73
	$\hat{\sigma}$	4.48	4.11	4.08	4.79	3.98	3.72	3.10	2.53	3.00	3.32	3.32	3.10	3.16
Pollen	M	9.72	9.72	9.28	9.06	5.46	8.73	4.80	5.46	5.79	4.15	4.59	4.37	4.37
	$\bar{x}$	9.78	9.82	9.18	8.78	5.46	8.82	4.77	5.37	5.78	4.11	4.64	4.30	4.32
	$\hat{\sigma}$	1.45	1.39	1.24	1.37	0.97	1.44	0.95	1.05	0.99	0.71	0.79	0.66	0.60
Raisins secs	M	0.89	0.67	0.67	1.56	0.78	0.44	0.44	0.22	0.44	0.44	0.44	0.44	0.44
	$\bar{x}$	0.92	0.75	0.60	1.60	0.95	0.56	0.46	0.30	0.48	0.46	0.46	0.36	0.36
	$\hat{\sigma}$	0.41	0.40	0.33	0.52	0.57	0.39	0.28	0.19	0.33	0.28	0.28	0.24	0.24

La colonne *S* indique laquelle des trois valeurs statistiques (soit M pour Médiane, $\bar{x}$ pour moyenne et $\hat{\sigma}$ pour écart type) est utilisée pour la ligne correspondante.

ANNEXE IV

TEMPS D'APPRENTISSAGE : PREMIÈRE SÉRIE

Tableau 6.4

Temps d'apprentissage pour la première série avec et sans ACP

Base / Classificateur	S	Bayes	KPPV	C4.5	PG	SVM	SVM2	PM
Céréales (grande)	M	0.05	0.10	0.49	6656.21	0.37	1.07	254.59
	$\bar{x}$	0.05	0.10	0.56	6714.42	0.39	1.06	226.64
	$\hat{\sigma}$	0.01	0.01	0.11	678.22	0.05	0.06	57.61
Céréales (grande) avec ACP	M	0.02	0.04	0.21	7434.08	0.22	0.77	26.50
	$\bar{x}$	0.02	0.04	0.25	7243.49	0.24	0.77	23.93
	$\hat{\sigma}$	0.00	0.01	0.05	695.82	0.03	0.05	5.23
Nœuds	M	0.28	0.37	0.83	10001.02	3.98	3.43	255.28
	$\bar{x}$	0.29	0.38	0.87	9998.07	4.14	3.43	227.26
	$\hat{\sigma}$	0.04	0.05	0.15	1135.58	0.46	0.04	56.86
Nœuds avec ACP	M	0.19	0.28	0.46	9947.96	3.94	3.33	26.94
	$\bar{x}$	0.21	0.29	0.52	10187.06	4.06	3.33	24.48
	$\hat{\sigma}$	0.12	0.03	0.07	1476.09	0.50	0.04	5.01
Feuilles	M	0.25	0.33	0.75	5916.04	1.17	1.33	125.33
	$\bar{x}$	0.25	0.34	0.76	5805.28	1.22	1.39	111.75
	$\hat{\sigma}$	0.04	0.01	0.18	903.45	0.15	0.15	27.63
Feuilles avec ACP	M	0.17	0.25	0.50	5257.38	1.08	1.00	17.33
	$\bar{x}$	0.20	0.26	0.50	5346.29	1.15	0.98	15.88
	$\hat{\sigma}$	0.08	0.03	0.04	771.24	0.18	0.04	3.06
Pollen	M	0.05	0.11	0.47	12218.57	0.68	0.85	175.73
	$\bar{x}$	0.06	0.11	0.54	12131.60	0.71	0.85	156.35
	$\hat{\sigma}$	0.01	0.01	0.10	1102.71	0.08	0.02	39.28
Pollen avec ACP	M	0.03	0.07	0.22	10343.13	0.62	0.72	26.82
	$\bar{x}$	0.03	0.07	0.23	10373.47	0.65	0.73	24.29
	$\hat{\sigma}$	0.01	0.01	0.03	1108.90	0.08	0.01	5.17
Raisins secs	M	0.07	0.12	0.26	2059.12	0.24	0.29	227.62
	$\bar{x}$	0.07	0.13	0.27	2080.33	0.25	0.29	202.25
	$\hat{\sigma}$	0.02	0.01	0.05	566.32	0.02	0.01	51.40
Raisins secs avec ACP	M	0.03	0.07	0.11	2487.31	0.22	0.27	17.92
	$\bar{x}$	0.04	0.07	0.11	2505.84	0.23	0.27	16.24
	$\hat{\sigma}$	0.01	0.00	0.01	643.99	0.03	0.01	3.46

La colonne *S* indique laquelle des trois valeurs statistiques (soit M pour Médiane, $\bar{x}$ pour moyenne et $\hat{\sigma}$ pour écart type) est utilisée pour la ligne correspondante. Les mesures sont en 10^{-3} seconde par échantillon.

ANNEXE V

TEMPS D'APPRENTISSAGE : DEUXIÈME SÉRIE

Tableau 6.5

Temps d'apprentissage pour la deuxième série avec et sans boosting

DB / C	S	Bayes	ABayes	KPPV	C4.5	AC4.5	PG	PGB	SVM	ASVM	SVM2	ASVM2	PM	APM
Céréales (petite)	M	0.004	0.259	0.011	0.048	0.549	1286.367	805.752	0.042	0.362	0.084	2.263	25.317	51.278
	$\bar{x}$	0.005	0.278	0.011	0.048	0.548	1262.096	801.881	0.042	0.362	0.083	2.213	25.389	55.832
	$\hat{\sigma}$	0.000	0.051	0.001	0.003	0.031	128.022	150.764	0.002	0.021	0.004	0.212	0.170	9.852
Chiffre	M	0.009	0.648	0.018	0.059	0.452	4123.986	577.216	0.327	2.164	0.309	3.248	14.116	42.391
	$\bar{x}$	0.009	0.632	0.017	0.060	0.452	4190.755	601.724	0.332	2.172	0.312	2.414	14.128	32.820
	$\hat{\sigma}$	0.001	0.045	0.002	0.003	0.011	912.779	162.969	0.013	0.084	0.007	1.359	0.082	19.024
Nœuds	M	0.028	0.495	0.037	0.083	0.449	1000.102	199.264	0.398	1.602	0.343	0.370	25.528	76.250
	$\bar{x}$	0.029	0.517	0.038	0.087	0.454	999.807	212.650	0.414	1.605	0.343	0.581	22.726	98.140
	$\hat{\sigma}$	0.004	0.139	0.005	0.015	0.033	113.558	74.855	0.046	0.036	0.004	0.548	5.686	78.571
Feuilles	M	0.025	0.175	0.033	0.075	0.367	591.604	614.017	0.117	0.487	0.133	1.042	12.533	124.608
	$\bar{x}$	0.025	0.198	0.034	0.076	0.372	580.528	610.328	0.122	0.489	0.139	1.045	11.175	94.788
	$\hat{\sigma}$	0.004	0.074	0.001	0.018	0.038	90.345	93.455	0.015	0.051	0.015	0.079	2.763	45.166
Feuilles rognées	M	0.017	0.358	0.033	0.050	0.217	478.890	54.037	0.125	0.517	0.100	0.117	12.450	12.546
	$\bar{x}$	0.017	0.355	0.036	0.051	0.215	486.180	59.212	0.122	0.525	0.100	0.121	12.477	16.283
	$\hat{\sigma}$	0.001	0.024	0.016	0.007	0.044	110.103	29.244	0.006	0.029	0.002	0.016	0.072	8.060
Pollen	M	0.005	0.257	0.011	0.047	0.495	1221.857	913.545	0.068	0.441	0.085	1.983	17.573	35.072
	$\bar{x}$	0.006	0.258	0.011	0.054	0.499	1213.160	898.209	0.071	0.441	0.085	1.995	15.635	39.267
	$\hat{\sigma}$	0.001	0.045	0.001	0.010	0.022	110.271	175.250	0.008	0.009	0.002	0.092	3.928	7.551
Raisins secs	M	0.007	0.308	0.012	0.026	0.243	205.912	27.648	0.024	0.100	0.029	0.038	22.762	22.746
	$\bar{x}$	0.007	0.296	0.013	0.027	0.202	208.033	32.723	0.025	0.089	0.029	0.038	20.225	22.730
	$\hat{\sigma}$	0.002	0.040	0.001	0.005	0.088	56.632	19.540	0.002	0.038	0.001	0.003	5.140	0.051

La colonne *S* indique laquelle des trois valeurs statistiques (soit M pour Médiane, $\bar{x}$ pour moyenne et $\hat{\sigma}$ pour écart type) est utilisée pour la ligne correspondante. Les mesures sont en 10^{-2} seconde par échantillon.

BIBLIOGRAPHIE

Banzhaf, W., P. Nordin, R.E. Keller et F.D. Francone. 1998. *Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and Its Applications*.

Bayes, T. 1763. « An Essay towards solving a Problem in the Doctrine of Chances ». Philosophical Transactions of the Royal Society of London.

Bellman, R.E. 1961. *Adaptive Control Processes*. Princeton, NJ: Princeton University Press, 274 p.

Berlage, A.G., T.M. Cooper et R.A. Carone. 1984. « Seed sorting by machine vision ». Agricultural Engineering, vol. 65, p. 14-17.

Bernier, T. 1997. « Development of an algorithmic method for the recognition of biological objects ». Mémoire de maîtrise, Montréal, McGill University.

Bibeau, Joël. 2006. *Investigation des techniques de la vision artificielle pour la classification d'objets biologiques dans des images 2D : Projet synthèse en génie de la production automatisée*. Montréal: École de Technologie supérieure.

Bourgouin, B. 2007. « Construction de caractéristiques par programmation génétique pour un système d'optimisation multiclasse ». Mémoire de maîtrise, Montréal, École de Technologie Supérieure.

Brameier, M. F., et W. Banzhaf. 2007. *Linear Genetic Programming*, XIII. Coll. « Genetic and Evolutionary Computation ». 315 p.

Breiman, Leo. 1996. « Bagging predictors ». Machine Learning, vol. 24, n° 2, p. 123-140.

Brierley, P., et B. Batty. 1999. « Data Mining With Neural Networks: An Applied Example in Understanding Electricity Consumption Patterns ». In *Knowledge Discovery and Data mining*, sous la dir. de Bramer, M. Coll. « Iee Professional Applications Of Computing Series ».

Brown, J.H., V.K. Gupta, B-L. Li, B.T. Milne, C. Restrepo et G.B. West. 2002. « The fractal nature of nature: Power laws, ecological complexity and biodiversity ». Biological Sciences, vol. 357, p. 619-626.

Buntine, Wray. 1996. « A Guide to the Literature on Learning Probabilistic Networks from Data ». IEEE Transactions on Knowledge and Data Engineering, vol. 8, n° 2, p. 195-210.

Burges, Christopher J.C. 1998. « A Tutorial on Support Vector Machines for Pattern Recognition ». Data Mining and Knowledge Discovery, vol. 2, p. 121-167.

Camastra, Francesco, et Alessandro Vinciarelli. 2008. *Image and Video Analysis, Theory and Applications*. Coll. « Advanced Information and Knowledge Processing ». 510 p.

Chan, J., et D. Braggins. 1996. « Untiring eyes ». Manufacturing Engineer, vol. 75, n° 5, p. 233-235.

Chandra, Arjun, et Xin Yao. 2006. « Evolving Hybrid Ensembles of Learning Machines for Better Generalisation ». Neurocomputing, vol. 69, n° 7-9, p. 686-700.

Churchill, D. B, D. M. Bilsland et T. M Cooper. 1993. « Separation of mixed lots of tall fescue and ryegrass seed using machine vision ». Transactions of the ASAE, vol. 36, p. 1383-1386.

Côté, M., E. Lecolinet, M. Cheriet et C.Y. Suen. 1998. « Automatic Reading of Cursive Scripts Using a Reading Model and Perceptual Concepts ». The International Journal on Document Analysis and Recognition, vol. 1, n° 1, p. 1-17.

Da Costa, L. E., J.-A. Landry et Yan Levasseur. 2007. « Treating Noisy Data Sets with Relaxed Genetic Programming ». In. Tours, France: Evolution Artificielle 2007. 8th International Conference on Artificial Evolution.

Dasarathy, Belur V. 1991. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. Los Alamitos, CA: IEEE Computer Society Press, 550 p.

Davies, E. R. 2004. *Machine Vision : Theory, Algorithms, Practicalities*, 3ième édition. Morgan Kaufmann, 934 p.

Du, Ji-Xiang, Xiao-Feng Wang et D.S. Huang. 2004. *Automatic plant leaves recognition system based on image processing techniques*. Institute of Intelligent Machines, Chinese Academy of Sciences.

Du, Ji-Xiang, Xiao-Feng Wang et Guo-Jun Zhang. 2007. « Leaf shape based plant species recognition ». Applied Mathematics and Computation, vol. 185, n° 2, Special Issue on Intelligent Computing Theory and Methodology (15 February 2007), p. 883-893.

Duda, Richard O., Peter E. Hart et David G. Stork. 2000. *Pattern classification*, 2. Wiley-Interscience, 654 p.

Eaton, John W. 2004. *MATLAB*, version. 7.0.1.24704 (R14). Logiciel. MathWorks. <http://www.mathworks.fr/products/matlab/>.

Eclipse Foundation. 2006. *Eclipse*, version. 3.2.1. Logiciel libre gratuit. <http://www.eclipse.org/>.

Fayyad, U., G. Piatetsky-Shapiro et P. Smyth. 1996a. « The KDD Process for Extracting Useful Knowledge from Volumes of Data ». Communications of the ACM, vol. 39, n° 2, p. 27-34.

Forsyth, David A., et Jean Ponce. 2003. *Computer Vision, A Modern Approach*. Upper Saddle River, NJ: Prentice Hall, 693 p.

France, Ian. 2007. « University of Bangor/Aberystwyth Pollen Image Database (Wales) ». In. <http://www.informatics.bangor.ac.uk/~ian/pdbase/pollen_dbase.html>.

Freund, Yoav. 1990. « Boosting a weak learning algorithm by majority ». In *Proceedings of the Third Annual Workshop on Computational Learning Theory*.

Gagné, C., M. Sebag, M. Schoenauer et M. Tomassini. 2007. « Ensemble learning for free with evolutionary algorithms? ». In (London, England, July 07 - 11, 2007). p. 1782-1789. New York, NY: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation.

GNU. 2007. *GNU General Public License*. en ligne. <http://www.gnu.org/copyleft/gpl.html>. Consulté le 7 janvier 2008.

Gonzalez, R. C., R. E. Woods et S. L Eddins. 2003. *Digital Image Processing Using MATLAB*. Upper Saddle River, NJ: Prentice Hall, 782 p.

Groszmann, A., et R. Poli. 2001. « Robust mobile robot localisation from sparse and noisy proximity readings using Hough transform and probability grids ». Robotics and Autonomous Systems, vol. 37, n° 1, p. 1-18.

Gustafson, S., et E. K. Burke. 2006. « The speciating island model: an alternative parallel evolutionary algorithm ». Journal of Parallel and Distributed Computing, vol. 66, n° 8, p. 1025-1036.

Harding, Simon, et Wolfgang Banzhaf. 2007. « Fast Genetic Programming on GPUs ». In (Valencia, Espagne, Avril 2007), sous la dir. de Ebner, M., M. O'Neill, A. Ekart, L. Vanneschi, A. Esparcia-Alcazar et Springer. p. 90-101. Berlin: Proceedings of the 10th European Conference on Genetic Programming.

Haton, Jean-Paul, Christophe Cerisara, Dominique Fohr, Yves Laprie et Kamel Smaïli. 2006. *Reconnaissance automatique de la parole : Du signal à son interprétation*. Coll. « UniverSciences ». Paris: Dunod, 392 p.

Hotelling, H. 1933. « Analysis of a Complex of Statistical Variables into Principal Components ». Journal of Educational Psychology, vol. 24, p. 417-441.

Jain, A.K., R.P.W. Duin et J. Mao. 2000. « Statistical Pattern Recognition: A Review ». IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, n° 1, p. 4-37.

Jain, A.K., Jianchang Mao et K.M. Mohiuddin. 1996. « Artificial neural networks: a tutorial ». Computer, vol. 29, n° 3, p. 31-44.

Jolliffe, I. 2002. *Principal Component Analysis*, 2ième édition. New York: Springer-Verlag.

Kong, Wai Kin, David Zhang et Wenxin Li. 2003. « Palmprint feature extraction using 2-D Gabor filters ». Pattern Recognition, vol. 36, n° 10, p. 2339-2347.

Koza, J.R. 1990. *Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems*. Coll. « Computer Science Department technical report ». Stanford University.

Koza, J.R. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. The MIT Press, 840 p.

Koza, J.R. 1994. *Genetic Programming II: Automatic Discovery of Reusable Programs*. The MIT Press, 768 p.

Koza, John R. 2007. « 36 Human-Competitive Results Produced by Genetic Programming ». In. <http://www.genetic-programming.com/humancompetitive.html>.

Langdon, W. B., et R. Poli. 2002. *Foundations of Genetic Programming*. Springer, 260 p.

Laplace, Pierre-Simon. 1820. *Théorie analytique des probabilités, Tome VII des œuvres complètes*, 3. Paris. <http://gallica.bnf.fr/ark:/12148/bpt6k88764q>.

Lee, C., et D. Landgrebe. 1993. « Feature Extraction Based on Decision Boundaries ». IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 15 n° 4, p. 388-400.

Levasseur, Yan. 2005. *Étude des performances de la Programmation Génétique pour un problème de reconnaissance de formes dans une image : Rapport de projet de session dans le cadre du cours SYS821 - Reconnaissance de formes et inspection*. Montréal: École de Technologie Supérieure, 30 p. <http://www.leyan.org/tiki-download_file.php?fileId=23>.

Levasseur, Yan. 2008a. *Bases de données*. En ligne. <http://www.leyan.org/tiki-index.php?page=Bases%20de%20données>. Consulté le 8 janvier 2008.

Levasseur, Yan. 2008b. *Extraction de caractéristiques*. En ligne. <http://www.leyan.org/tiki-index.php?page=Extraction%20de%20caracéristiques>. Consulté le 8 janvier 2008.

Levasseur, Yan. 2008c. *Leyan : Accueil*. En ligne. <http://www.leyan.org/>. Consulté le 8 janvier 2008.

Levasseur, Yan. 2008d. *Programmation Génétique*. En ligne. <http://www.leyan.org/tiki-index.php?page=Programmation%20génétique>. Consulté le 8 janvier 2008.

Liu, H., et H. Motoda. 1998. *Feature Extraction, Construction and Selection: A Data Mining Perspective*. Springer, 440 p.

Majumdar, S., et D.S. Jayas. 2000. « Classification of cereal grains using machine vision: I Morphology Models ». Transactions of the ASAE, vol. 43, n° 6, p. 1669-1675.

Natterer, Michael, et Sven Neumann. 2007. *GNU Image Manipulation Program (GIMP)*, version. 2.4.0. Open source software. GIMP Team. <http://www.gimp.org/>.

Niehaus, J., C. Igel et W. Banzhaf. 2007. « Reducing the Number of Fitness Evaluations in Graph Genetic Programming Using a Canonical Graph Indexed Database ». Evolutionnary Computation, vol. 15, n° 2, p. 199-221.

Nordin, Peter, Wolfgang Banzhaf et Franck Francone. 1999. « Efficient Evolution of Machine Code for CISC Architectures using Blocks and Homologous Crossover ». In *Advances in Genetic Programming, Vol. 3 : Complex Adaptative Systems*, sous la dir. de Spector, Lee, William B. Langdon, Una-May O'Reilly et Peter J. Angeline. Vol. 3, p. chapitre 6. MIT Press.

Nordin, Peter, Frank Francone et Wolfgang Banzhaf. 1996. « Explicitly defined introns and destructive crossover in genetic programming ». In *Advances in genetic programming*, sous la dir. de Angeline, Peter J., et Jr K. E. Kinnear. Vol. 2, p. 111 - 134.

Object Management Group. 2007. « Unified Modeling Language (UML), version 2.1.1 ». In. <http://www.omg.org/technology/documents/formal/uml.htm>. Consulté le 20 août 2007.

Ojala, T., M. Pietikäinen et D. Harwood. 1996. « A comparative study of texture measures with classification based on featured distribution ». Pattern Recognition, vol. 29, n° 1, p. 51-59.

Ojala, T., M. Pietikäinen et T. Mäenpää. 2002. « Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns ». IEEE Transactions on Pattern Analysis and Machine Intelligence, p. 971-987.

Pearson, K. 1901. « On Lines and Planes of Closest Fit to Systems of Points in Space ». Philosophical Magazine, vol. 2, p. 559-572.

Pearson, T.C., et D.C. Slaughter. 1996. « Machine vision detection of early split pistachio nuts ». Transactions of the ASAE, vol. 39, p. 1203-1207.

Proulx, R. 2006. « Ecological complexity for unifying ecological theory across scales: A field ecologist's perspective ». Ecological Complexity, vol. in press.

Proulx, R., et L. Parrott. 2007. « Measures of structural complexity in digital images for monitoring the ecological signature of an old-growth forest ecosystem ». Ecological Indicators, vol. 8, n° 3, p. 270-284.

Quinlan, J. R. 1986. « Induction of Decision Trees ». Machine Learning, vol. 1, p. 81-106.

Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning*. Coll. « Machine Lerning ». Morgan Kaufmann, 302 p.

Reutemann, Peter. 2007. *en:ARFF (3.4.6)*. <http://weka.sourceforge.net/wekadoc/index.php/en:ARFF_%283.4.11%29>.

Rigney, M.P., G.H. Brusewitz et G.A Kranzler. 1992. « Asparagus defect inspection with machine vision ». Transactions of the ASAE, vol. 35, p. 1873-1878.

Rosenblatt, F. 1962. *Principles of Neurodynamics*. Washington: Spartan.

Saitoh, T., et T. Kaneko. 2000. « Automatic recognition of wild flowers ». Proceedings of Pattern Recognition, vol. 2, n° 2000, p. 507-510.

Schölkopf, Bernhard, Christopher J.C. Burges et Alexander J. Smola. 1998. *Advances in Kernel Methods: Support Vector Learning*. Cambridge, MA: The MIT Press, 386 p.

Sebastiani, Fabrizio. 2002. « Machine learning in automated text categorization ». ACM Computing Surveys, vol. 34, n° 1, p. 1–47.

Setiono, R., et H. Liu. 1998. « Feature Extraction via Neural Networks ». In *Feature Extraction, Construction and Selection: A Data Mining Perspective*, sous la dir. de Liu, H., et H. Motoda, 2. p. 191-204. Boston: Kluwer Academic Publishers.

Shaanker, R. Uma. 2001. « Data mining and knowledge discovery: Emerging fashions in science ». Current Science, vol. 80 n° 5, p. 603.

Shakhnarovich, Gregory, Trevor Darrell et Piotr Indyk. 2006. *Nearest-Neighbor Methods in Learning and Vision*. The MIT Press, 262 p.

Shannon, C.E. 1948. « A Mathematical Theory of Communication ». Bell System Technical Journal, vol. 27, p. 379-423, 623-656.

Shatadal, P., D.S. Jayas, J.L. Hehn et N.R. Bulley. 1995. « Seed classification using machine vision ». Canadian Agricultural Engineering, vol. 37, n° 3, p. 163-167.

Shawe-Taylor, John, et Nello Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 476 p.

Silven, O., M. Niskanen et H. Kauppinen. 2000. « Visual Inspection of Lumber ». In. <http://www.ee.oulu.fi/~olli/Projects/Lumber.Grading.html>.

Sobel, I., et G. Feldman. 1973. « A 3x3 Isotropic Gradient Operator for Image Processing ». In *Pattern Classification and Scene Analysis*, sous la dir. de Duda, R., et P. Hart. présenté lors du Stanford Artificial Project en 1968, non publié mais souvent cité. p. 271-272. John Wiley and Sons.

Sonnenburg, Soeren, Mikio Braun et Cheng Soon Ong. 2008. *MLOSS : Machine Learning Open Source Software*. en ligne. <http://mloss.org/>. Consulté le 7 janvier 2008.

Stathakis, D., et A. Vasilakos. 2006. « Comparison of computational intelligence based classification techniques for remotely sensed optical image classification ». IEEE Transactions on Geoscience and Remote Sensing, vol. 44, n° 8, p. 2305- 2318.

Teller, A., et M. Veloso. 1997. « PADO: A new learning architecture for object recognition ». In *Symbolic Visual Learning*, sous la dir. de Ikeuchi, K., et M. Veloso. p. 81-116. Oxford University Press.

Teredesai, A., et V. Govindaraju. 2004. « Issues in Evolving GP Based Classifiers for a Pattern Recognition Task ». In *IEEE Congress on Evolutionary Computation*. Vol. 1, p. 509-515.

Teredesai, A., J. Park et V. Govindaraju. 2001. « Active handwritten character recognition using genetic programming ». In, sous la dir. de EuroPG2001.

Tukey, J.W. 1977. *Exploratory Data Analysis*. Addison-Wesley, 688 p.

University of Oulu Machine Vision Group. *LBP Methodology*. Fichier pdf. <http://www.ee.oulu.fi/research/imag/texture/lbp/about/LBP%20Methodology.pdf>. Consulté le 15 juillet 2007.

Van Der Heijden, G.W.A.M., A.M. Vossepoel et G. Polder. 1996. « Measuring onion cultivars with image analysis using inflection points ». Euphytica vol. 87, n° 1, p. 19-31.

van der Werf, E. 2004. « AI techniques for the game of Go ». Thèse de doctorat, Maastricht, The Netherlands, Universiteit Maastrich.

Vapnik, Vladimir. 1998. *Statistical Learning Theory*. Wiley-Interscience, 736 p.

Velleman, P.F., et D.C. Hoaglin. 1984. *Applications, Basics, and Computing of Exploratory Data Analysis*. Wadsworth Pub Co, 292 p.

Visen, N.S., D.S. Jayas et N.D.G. White. 2004. « Image analysis of bulk grain samples using neural networks ». Canadian Biosystems Engineering, vol. 46, p. 7.11-7.15.

Weber, Markus. 2007. « Computational Vision: Archive ». In. <http://www.vision.caltech.edu/html-files/archive.html>.

Weise, Thomas. 2008. *Distributed Genetic Programming Framework*. En ligne. <http://dgpf.sourceforge.net/>. Consulté le 16 janvier 2008.

Weiss, S., et N. Indurkhya. 1998. *Predictive Data Mining, a Practical Guide*. San Francisco, California.

Weiss, S., et C. Kulikowski. 1991. *Computer Systems That Learn: Classification and Prediction Methods From Statistics, Neural Nets, Machine Learning and Expert Systems*.

Wikipédia. 2004. « Image:Cranberrys beim Ernten.jpeg ». In *Wikipédia : L'encyclopédie libre*. En ligne. <http://fr.wikipedia.org/wiki/Image:Cranberrys_beim_Ernten.jpeg>. Consulté le 26 novembre 2007.

Wikipédia. 2005. « Image:Go board.jpg ». In *Wikipédia : L'encyclopédie libre*. En ligne. <http://fr.wikipedia.org/wiki/Image:Go_board.jpg>. Consulté le 26 novembre 2007.

Wikipédia. 2006a. « Bild:Diskriminanzfunktion.png ». In *Wikipédia : L'encyclopédie libre*. En ligne. <http://de.wikipedia.org/wiki/Bild:Diskriminanzfunktion.png>. Consulté le 26 novembre 2007.

Wikipédia. 2006b. « Image:Artificial neural network.svg ». In *Wikipédia : L'encyclopédie libre*. En ligne. <http://en.wikipedia.org/wiki/Image:Artificial_neural_network.svg>. Consulté le 26 novembre 2007.

Wikipédia. 2006c. « Image:Man-full.png ». In *Wikipédia : L'encyclopédie libre*. En ligne. <http://fr.wikipedia.org/wiki/Image:Man-full.png>. Consulté le 26 novembre 2007.

Wikipédia. 2007a. « Image:Genetic Program Tree.png ». In *Wikipédia : L'encyclopédie libre*. En ligne. <http://en.wikipedia.org/wiki/Image:Genetic_Program_Tree.png>. Consulté le 26 novembre 2007.

Wikipédia. 2007b. « Image:KnnClassification.svg ». In *Wikipédia : L'encyclopédie libre*. En ligne. <http://en.wikipedia.org/wiki/Image:KnnClassification.svg>. Consulté le 26 novembre 2007.

Wikipédia. 2007c. « Image:Partly disassembled Lumix digital camera.jpg ». In *Wikipédia : L'encyclopédie libre*. En ligne. <http://en.wikipedia.org/wiki/Image:Partly_disassembled_Lumix_digital_camera.jpg >. Consulté le 26 novembre 2007.

Witten, Ian H., et Eibe Frank. 2005. *Data Mining: Practical machine learning tools and techniques*, 2. San Francisco: Morgan Kaufmann, 525 p.

Wolfe, R.R., et M. Swaminathan. 1987. « Determining orientation and shape of bell peppers by machine vision ». Transactions of the ASAE, vol. 30, p. 1853-1856.

Yang, Q. 1996. « Apple stem and calyx identification with machine vision ». Journal of Agricultural Engineering Research, vol. 63, p. 229-236.

Zayas, I., et P.W. Flinn. 1998. « Detection of insects in bulk wheat samples with machine vision ». Transactions of the ASAE, vol. 41, n° 3, p. 883-888.