UNIVERSITÉ LOUIS PASTEUR STRASBOURG

Laboratoire des Sciences
de l'Image, de l'Informatique et de la
Télédétection

UMR 7005 CNRS/ULP

Équipe Apprentissage et
Fouille de Données

N° d'ordre : 4997

Thèse présentée pour obtenir le grade de
Docteur de l'Université Louis Pasteur
Strasbourg I

Discipline : INFORMATIQUE

par **Arnaud Quirin**

Découverte de règles de classification par approche évolutive : application aux images de télédétection

Soutenue publiquement le 12 décembre 2005

Membres du jury

Rapporteurs externes : M. Alexandre CAMINADA, Professeur
Université de Technologie de Belfort-Montbéliard
M. Pierre COLLET, Habilité à Diriger des Recherches
Université de Littoral Côte d'Opale de Calais
Rapporteur interne : Mme Christiane WEBER, Directrice de Recherche
Université Louis Pasteur de Strasbourg
Co-directeurs : M. Jerzy KORCZAK, Professeur
Université Louis Pasteur de Strasbourg
M. Massimo MENENTI, Directeur de Recherche
Université Louis Pasteur de Strasbourg
Invité : M. David E. GOLDBERG, Professeur
University of Illinois at Urbana-Champaign, USA

Notes :

Document généré le 19 janvier 2006. Version finale.
Pour toute remarque ou information, pour signaler une erreur ou pour avoir la version définitive, merci de contacter l'auteur :
Arnaud Quirin <quirin@lsiit.u-strasbg.fr>

À toi, papi...

Remerciements

Me voici arrivé à la dernière page que je rédige et la première que vous lisez. Il se trouve que ce n'est pas la plus facile à rédiger, loin de là, compte tenu des nombreux contacts, échanges et secours que j'ai eu la chance d'obtenir de vous.

Je commencerai par remercier mes deux directeurs de thèse, Jerzy Korczak et Massimo Menenti, pour leur confiance, leur aide et leur soutien. Les membres de mon équipe ont toujours répondu présent quand j'avais besoin d'eux, notamment Pierre Gançarski, Cédric Wemmert, David Sheeren, Alexandre Blansché et Sébastien Dérivaux, sans oublier Agnès Braud pour ses précieux conseils et corrections, y compris celles de dernière minute !

Je remercie très chaleureusement toutes celles et tous ceux qui m'ont accueilli dans leurs laboratoires respectifs pour de courts séjours durant ma thèse. Je pense à David Goldberg, Pier-Luca Lanzi, Martin Butz, Kumara Sastry et Xavier Llorà du laboratoire IlliGAL dans la petite ville d'Urbana-Champaign aux Etats-Unis pour leur grande disponibilité. Merci aussi à toute l'équipe de géographes de Padoue de Venise pour m'avoir fait travailler toutes ces années sur une superbe région et de très belles images. Je pense à Marco Marani, Cheng Wang, Enrica Belluco, Monica Camuffo et Sergio Ferrari du projet *Tidal Inlets Dynamics and Environment* avec qui j'ai passé de merveilleux moments à remuer la vase de San Felice pour faire avancer la Science ! Merci encore à Christiane Weber, Anne Puissant et Jacky Hirsch du laboratoire Image et Ville de Strasbourg pour leurs nombreux conseils, remarques et à-propos concernant les données de télédétection, particulièrement dans le cadre de l'action concertée incitative *FoDoMust*. Les quelques heures du cours de géographie de Abdelaziz Serradj que j'ai suivi ont pu éclairer ma recherche d'un point de vue différent.

Mes remerciements vont aussi à mes rapporteurs externes, Pierre Collet et Alexandre Caminada, pour leurs remarques pertinentes par courriers ou lors de la soutenance, qui ont beaucoup contribué à améliorer la qualité de mon manuscrit.

Je n'oublierais pas non plus le « petit noyau dur » qui se reconnaîtra, Jean Hommet et Aurélie Bertaux pour leur bonne humeur, les sourires de Anne-Lise, Louise, Raphaële, Stéphanie et Véronique qui ont adouci les moments difficiles, ni toutes celles et ceux avec qui j'ai passé de chouettes soirées entre deux lignes de code.

Enfin merci à ma famille, tout spécialement à mes grands-parents, qui m'ont aidé jusqu'au bout à faire de cette thèse ce qu'elle est finalement devenue.

"LCS are a quagmire – a glorious, wondrous and inviting quagmire, but a quagmire nonetheless."

D. E. Goldberg, 1992

Table des matières

Introduction

La découverte de connaissances dans les bases de données apparaît à l'heure actuelle comme l'une des thématiques les plus dynamiques en Intelligence Artificielle (IA). Elle est mise en œuvre par un processus complexe dont le but est d'extraire, à partir d'un entrepôt de données, de l'information qui a un *sens* pour un utilisateur humain. Les connaissances découvertes doivent avoir un haut niveau d'abstraction par rapport aux données initiales. En effet, les données élémentaires, hétérogènes et volumineuses – en un mot incompréhensibles, sont transformées en une représentation plus compacte (par exemple, une approximation, un modèle expliquant les données) et/ou plus utile (un modèle cohérent des données, un modèle prédictif, ...). Au cœur de ce processus, l'étape d'extraction proprement dite, est appelée fouille de données (ang., *Data Mining*). La contribution apportée par tout processus de fouille de données est la possibilité d'expliquer les connaissances extraites. Il existe de multiples façons de formaliser ces connaissances, certaines pouvant être traduites sous forme de règles lisibles alors que d'autres sont de type « *boîte noire* ». Nous avons choisi de travailler sur des techniques permettant d'engendrer des connaissances compréhensibles, sous la forme de *règles* s'appliquant sur les données et renvoyant une expertise de plus haut niveau (classe, statistiques sur l'échantillon, ...). Nous nommons ce processus d'extraction *découverte de règles de classification*.

La fouille de données, appliquée à des domaines très divers, vise particulièrement les très grandes bases de données, à l'image de celles rencontrées dans le monde réel, comme dans les télécommunications, l'astronomie, la thérapeutique ou le e-commerce. Le cadre de travail dans lequel s'inscrit ce mémoire est la télédétection dont les applications sont la classification, la cartographie, l'évaluation et la reconnaissance des caractéristiques du terrain, la gestion d'environnement naturel, l'aménagement de territoire urbain, etc. Dans ce domaine, les chercheurs sont confrontés à des images de très grande taille (par exemple, 12000 x 8000 pixels pour le capteur ROSIS), bruitées (notamment par les conditions atmosphériques ou par celles de la prise de vue) et complexes car ce sont des données réelles. Ces données sont aussi de plus en plus fastidieuses à manipuler puisque leur quantité, leur complexité et leur taille suivent l'amélioration constante de la technologie de prise de vue et de la sensibilité des capteurs, que ce soient ceux de satellites ou d'avions volant à haute altitude. L'apprentissage et l'extraction automatique de connaissances se révèlent adaptés à ce domaine compte tenu de leurs innombrables applications et des difficultés présentées par les données réelles.

L'approche évolutionnaire

Les difficultés évoquées suggèrent l'emploi d'algorithmes évolués, déjà utilisés dans d'autres domaines qui engendrent des masses de données complexes, comme le commerce, la production industrielle ou l'ingénierie. Les algorithmes évolutionnaires sont connus pour être robustes au bruit et peu soucieux du nombre de données, donc à même de traiter un problème de classification d'imagerie satellitaire. Un bref point de terminologie doit être formulé ici. Dans notre cas, un *classifieur* est une règle issue du processus d'extraction de connaissance considéré et une *classification* est le résultat de son application. Les algorithmes évolutionnaires comme les systèmes de classifieurs [Holland, 1975; Wilson, 1995] sont entièrement dédiés à la découverte de classifieurs. Les systèmes de classifieurs (ang., *Learning Classifier Systems* ou *LCS*) engendrent des populations de règles de classification simples, lisibles et généralisantes (par la présence de caractères *joker* ou d'intervalles de confiance permettant de modéliser de nombreux cas similaires en une seule règle). Les règles apprises constituent une base de connaissances qui peut être

appliquée sur une autre partie de l'image, voire une nouvelle image, en garantissant un taux correct de vrais positifs et vrais négatifs pour la classification de nouveaux exemples. Cette approche n'a cependant pas été explorée dans le domaine de la télédétection. À notre connaissance, aucun travail n'a été publié dans les principales conférences relatives à ce sujet (GECCO[1] et IWLCS[2]). En réalité, la recherche dans ce domaine est en plein essor, comparée à celle gouvernée par les algorithmes génétiques. Peu d'applications en télédétection ont pu à ce jour être conjuguées avec les *LCS*, malgré les nombreux travaux théoriques qui leur ont été consacrés, dont certains sont récents [Wilson, 2000b; Butz et Wilson, 2002; Studley et Bull, 2005].

Pour l'apprentissage de nos classifieurs, nous disposons de connaissances préalables assez abondantes, très bien renseignées et documentées par les experts de terrain. Pour les exploiter, nous nous sommes tournés vers une approche d'apprentissage supervisée. Dans ce contexte, la découverte de classes précises et exactes compte parmi les objectifs essentiels que se fixent les thématiciens. Pour un algorithme, répondre à cette requête est une mission relativement ardue.

En effet, pour extraire les concepts thématiques étudiés par l'expert, nous disposons souvent de sources d'information nombreuses (images satellitaires, images à haute altitude, mesures aérostatiques, mesures laser, validation terrain, validations spectrométriques, informations expertes informelles), de différentes natures et peu cohérentes entre elles. Une part de notre étude est ainsi consacrée à la mise en place d'un formalisme de représentation de la connaissance adapté à l'extraction de ces concepts. Cette problématique est au cœur d'un autre projet de recherche plus vaste, initié par l'équipe Apprentissage et Fouille de Données du LSIIT dans le cadre de l'Action Concertée Incitative (ACI) *Masse de données* : le projet *FoDoMust*[3]. L'objectif de ce projet est de proposer une stratégie permettant l'utilisation conjointe de différentes sources d'images en vue d'en extraire des objets qui dépassent le niveau du pixel, en utilisant notamment des connaissances de haut niveau comme des ontologies.

Nous sommes aussi en étroite collaboration avec le laboratoire Image et Ville de Strasbourg [LIV, 2005] qui nous a fourni des documents de travail sur Strasbourg, et avec le projet européen TIDE[4] [TIDE, 2005; Marani et al., 2003; Marani et al., 2004] pour des images de Venise. Dans ce projet, nous avons participé à un groupe de travail (*Work Package 6*), visant à développer et valider des modèles dynamiques et complets de systèmes marécageux incorporant à la fois des processus écologiques et physiques. Ce projet européen, réunissant des équipes italiennes, anglaises, allemandes et françaises, a nourri la majorité de notre base de données, par des images multispectrales et hyperspectrales provenant des capteurs QuickBird, CASI ou ROSIS, autorisant une résolution au sol de 60 cm à 3 m, sur plusieurs centaines de canaux spectraux. Grâce à ce projet, nous disposons aussi de nombreuses expertises sur ces données.

Problématiques de recherche

Les travaux présentés dans ce mémoire se divisent principalement en trois problématiques qui sont la représentation des règles, le processus de leur découverte et la classification floue. Nos travaux sur la découverte de règles de classification nous ont conduits à explorer l'**influence de la représentation** sur la capacité des règles à façonner de nouvelles connaissances et à intégrer celles existantes. La représentation a aussi une influence significative sur la qualité de reconnaissance et donc sur le pouvoir de *prédiction* des règles découvertes. Cette problématique centrale nous a guidés lors de la définition d'une liste de critères permettant de qualifier les propriétés d'un « bon » classifieur. Cette liste nous a servi de base de travail lors des mesures de lisibilité, de simplicité ou de performance des règles expliquant la présence ou l'absence de divers attributs proposés par les experts en télédétection.

La seconde problématique concerne le **processus de découverte** de ces règles. Nous avons conçu des algorithmes évolutifs comprenant divers opérateurs génétiques pour créer, manipuler et évaluer les règles, ainsi que pour détecter la convergence de l'apprentissage. Nous abordons aussi le problème du raffinement des règles ainsi produites. On reproche souvent aux systèmes de classifieurs de produire une base comprenant trop de classifieurs, pour des problèmes supposés en nécessiter beaucoup moins.

[1] Genetic and Evolutionary Computation Conference
[2] International Workshop on Learning Classifier Systems
[3] Fouille de données multi-stratégie, http://lsiit.u-strasbg.fr/afd/fodomust
[4] Tidal Inlets Dynamics and Environment, http://www.istitutoveneto.it/tide/project/tide.php

Nous proposons une solution consistant à post-traiter la population de règles obtenue pour la raffiner, améliorer la qualité de classification et réduire le nombre de règles. Pour cela, différentes approches ont été testées. La première est basée sur les algorithmes génétiques et consiste à créer des individus représentant des sous-populations de la population initiale. La seconde approche est basée sur la réutilisation des règles de la base comme prédicats principaux pour former les nœuds d'un arbre de décision, découvert avec une adaptation de l'algorithme inductif C4.5. Dans cette approche, l'algorithme de construction de l'arbre de décision permet une simplification automatique (en nombre de règles) de la base ainsi qu'une hiérarchisation de ces règles, ce qui tend à donner à l'arbre un pouvoir de généralisation plus important.

Enfin, la troisième problématique porte sur l'un des derniers enjeux de la classification en télédétection. Dans de nombreux cas il peut s'avérer nécessaire d'examiner – en plus du problème habituel de classification – l'idée qu'un même pixel peut appartenir à plusieurs classes. Les thématiciens considèrent que les algorithmes permettant d'**associer plusieurs classes à un pixel** sont les seuls pouvant s'approcher davantage d'une modélisation correcte de la réalité spectrale du terrain. En fait, chaque pixel n'est qu'une mixture des valeurs de réflectance spectrale de plusieurs types de terrain différents, dont les différentes abondances caractérisent la forme finale du spectre observé. On parle alors de *classification floue* et sa résolution fait appel à la notion d'*unmixing*. Ces nouvelles méthodes sont très prometteuses, d'autant plus que l'utilisation de multiples canaux rend aujourd'hui ces approches envisageables. Différents problèmes de classification réclament différents formalismes de règles. À cette intention, nous avons créé ou adapté, par des modifications parfois simples, des représentations permettant le renvoi d'une valeur réelle correspondant à une information quantitative plutôt que d'un simple attribut nominatif. À la lumière de certaines mesures de qualité validant les résultats, certaines de ces modifications nous ont semblé bien appropriées pour la résolution des problèmes de classification floue.

Structure de la thèse

Ce mémoire suit le plan suivant. Le premier et le second chapitre se proposent d'effectuer un état de l'art des différentes techniques supportant la production de règles de classification. Nous y étudions à la fois des méthodes non évolutives comme les systèmes inductifs ou les réseaux de neurones, et des méthodes évolutives comme les algorithmes génétiques, les systèmes à base de renforcement ou la programmation génétique. Nous avons aussi construit une liste de critères caractéristiques des classifieurs, que nous présentons en premier.

Le troisième chapitre développe la problématique de la classification d'images hyperspectrales que nous n'avons qu'effleurée jusqu'à présent et déploie, sous forme de figures, de diagrammes et de tableaux récapitulatifs, notre matériel de travail. Les diverses images brutes que nous avons utilisées sont présentées en complément des données validées par l'expert. Nous en profitons également pour préciser certains aspects ayant trait à la complexité des données, suivis de quelques pré-traitements qui leurs sont habituellement réservés. Nous terminons par deux études, courtes mais essentielles pour poursuivre, l'une sur la réduction des données, l'autre sur la robustesse d'une méthode évolutive face au bruit d'une image.

Les trois chapitres suivants composent le chaînon central. Le premier des trois présente les concepts théoriques de la découverte de nos règles. Les différents types de problèmes de classification y sont abordés, ainsi qu'une terminologie et une architecture générique, que nous avons définies et qui nous servent de cadre de travail. Cette architecture comprend notamment un algorithme intervenant dans le processus de la découverte des règles, ainsi qu'une série de mesures de qualité courantes ou nouvelles, utiles pour nos expérimentations. Nous avons distingué les mesures spécifiques au domaine d'étude, par rapport aux mesures génériques, et celles calculées durant le processus d'apprentissage par rapport à celles utilisées en validation. Le chapitre suivant détaille les algorithmes, que nous avons conçus, adaptés à la classification standard et à la classification floue. Enfin, nous proposons dans la troisième partie de nouveaux algorithmes de post-traitement des règles de classification.

Les études de cas du dernier chapitre éprouvent et comparent les différents algorithmes mis au point durant nos travaux. Divers protocoles de validation, mesures de qualité et mesures comparatives ont été utilisés pour juger de l'efficacité et de la compréhensibilité des règles de classification en environnement rural ou urbain. Pour permettre à notre logiciel de s'adresser à de nombreux utilisateurs, experts ou non en données de télédétection, nous présentons trois méthodes offrant une visualisation graphique de

la population de règles et des résultats. Nous montrons que chacun des algorithmes de classification s'adresse à une problématique particulière. Cependant, nous proposons en conclusion un système à base de *vote consensuel* surmontant les différences conceptuelles de ces algorithmes et permettant d'élire une classification finale.

Nous terminons par un résumé de notre contribution, suivi d'un tour d'horizon des perspectives offertes par nos travaux. Pour que ce mémoire puisse aussi être repris par des lecteurs non initiés à divers algorithmes basiques en IA, nous avons rassemblé les informations nécessaires en annexe. Les deux premières annexes sont consacrées à l'étude de certains algorithmes connexionnistes et des machines à vecteur support, ainsi que leur apport en télédétection. Ces algorithmes sont connus pour être robustes dans ce domaine, nous les avons donc pris comme étalon pour nos comparaisons. La troisième annexe détaille le fonctionnement des opérateurs génétiques. Les deux annexes suivantes exposent les objectifs des deux projets dans lesquels nos travaux s'inscrivent (le projet TIDE et le projet FoDoMust). Enfin, la dernière annexe décrit notre plate-forme d'expérimentation **VPlat**.

Chapitre 1

État de l'art

1.1 Introduction

Le traitement de grandes bases de données, qu'elles soient textuelles, économiques (assurances), médicales ou géographiques comme dans notre cas, fait traditionnellement appel au domaine de la fouille de données (ang., *data mining*), car elle permet grâce à un processus itératif d'extraction de connaissance de traiter des données souvent volumineuses, provenant de sources physiques (capteurs) diverses et regroupant des données de différentes natures. Actuellement il existe de nombreuses méthodes de fouille supervisée (inductives, connexionnistes, évolutives, ...) [Mitchell, 1997; Goldberg, 1989; Haykin, 1999]. Nous nous intéressons particulièrement à celles capables de découvrir des règles (ang., *rule discovery*) car elles participent à la réduction et à l'explication de la connaissance dans ces grandes bases de données. Le processus d'extraction est complexe car il faut préparer les données brutes, adapter des algorithmes spécialisés puis éventuellement raffiner les règles obtenues pour obtenir une qualité de généralisation suffisante.

L'état de l'art que nous proposons suit 2 axes différents en parallèle. Il met en valeur les méthodes et les contributions théoriques qui permettent de produire des règles ou des systèmes à base de règles simples et efficaces (extraction depuis les réseaux de neurones, systèmes inductifs, ...), ainsi que les études théoriques qui ont porté sur l'amélioration de l'efficacité de ces règles. Le second axe concerne l'application des algorithmes, principalement évolutifs, en télédétection. Nous montrons, à chaque fois que cela est possible, ce qu'a apporté la génétique pour le traitement de ces images. Cette structuration en deux axes distincts vient du fait qu'ils sont tout deux intégrés à notre problématique. Toutefois, il n'existe que peu d'études combinant les deux. Sinon, lorsque les domaines se sont recouverts, nous signalons ces études et ce dont la partie applicative a tiré parti de la partie théorique.

Le chapitre possède la structure suivante. La première section dresse une liste des propriétés attendues des règles de classification. Les sections suivantes sont structurées par classe de méthodes. La section 1.3 présente les algorithmes inductifs travaillant directement avec une représentation arborescente, la section 1.4 évoque les algorithmes qui permettent l'extraction de règles depuis les réseaux neuronaux, les deux suivantes concernent les systèmes évolutifs, l'apprentissage par renforcement en particulier et nous conclurons dans la dernière section.

1.2 Propriétés des classifieurs

1.2.1 Critères d'évaluation concernant les classifieurs extraits

Considérant le nombre impressionnant de techniques permettant de produire une base de règles, qu'elles soient évolutives ou non, nous devons nous résoudre à aborder le problème de manière pragmatique. Le choix de la technique à adopter pour la résolution d'un problème de classification ne doit pas simplement tenir compte de sa performance ou sa robustesse. Les algorithmes que nous allons évoquer

ont été évalués sur des données très diverses et ils présentent eux-même des approches différentes. Il est donc difficile de déterminer la technique idéale, simplement en examinant les algorithmes d'extraction ou de découverte de règles. Pour cela, nous avons commencé par tirer la liste des critères caractérisant un « bon » classifieur. [Zhou, 2003] identifie deux types de critères d'évaluation : ceux concernant les classifieurs extraits et ceux concernant l'algorithme d'apprentissage lui-même.

La qualité de généralisation. Il s'agit d'une classe d'indices permettant de savoir si l'algorithme va maintenir ses performances ou s'améliorer sur des données non apprises. Par exemple, l'indice PA (ang., *Predictive Accuracy*) est défini comme suit :

$$PA = \frac{\textit{Nombre d'exemples non appris classifiés correctement}}{\textit{Nombre total d'exemples dans le jeu de test}}$$

La compréhensibilité. Comprendre une base de règles de manière intelligible est très importante pour l'expert dans le cadre d'une validation, ou tout simplement lors d'une consultation de la connaissance apprise par le système. Malheureusement, il n'est pas facile de mesurer un tel critère notamment à cause de son caractère subjectif mais aussi du fait qu'il est dépendant de la représentation. Par exemple, on peut considérer la complexité syntaxique de la règle mesurée par la longueur de son expression, la *spécificité* de l'expression, le nombre de règles dans la base ou le nombre de conditions dans chacune des règles comme étant des mesures adaptées au cas des systèmes de classifieurs (SC), dont nous allons parler dans la suite.

La robustesse face aux données. En télédétection comme en base de données nous rencontrons souvent des données bruitées, erronées, redondantes ou manquantes. Quel est la robustesse du classifieur face à de tels ensembles ? De tests peuvent être effectués par ajout de bruit *de laboratoire* (il s'agit d'un bruit dont la distribution est connue) ou sur des images contenant des artefacts réels. De plus, le bruit étant une augmentation de l'entropie moyenne des données, on peut noter qu'un classifieur plus simple (délivrant ou stockant moins d'informations) représente une solution plus robuste car ce classifieur ne pourra pas être placé en situation de sur-apprentissage.

L'indépendance algorithmique. Un classifieur est indépendant de l'algorithme qui l'a créé s'il peut être interprété ou amélioré à l'aide d'un autre algorithme ou d'une technique qui ne reposent pas sur les mêmes fondements. Par exemple, un classifieur dont les paramètres ont été découverts par un algorithme génétique et qui est suffisamment souple pour être ensuite optimisé par un recuit simulé est dit avoir une indépendance algorithmique élevée. Cette notion est totalement liée à l'interopérabilité de la connaissance entre les différents algorithmes, et donc d'une certaine façon, à la simplicité de la représentation.

1.2.2 Critères d'évaluation concernant l'algorithme d'apprentissage

Les critères que nous avons vus dans la section précédente ne sont pas suffisants : en effet, deux algorithmes différents peuvent produire des règles simples, par exemple d'une longueur d'expression faible. Les critères que nous allons voir ici permettent de discriminer les algorithmes d'apprentissage eux-mêmes.

La représentation de la connaissance. Est-ce que l'algorithme permet de réduire la complexité de la représentation de la connaissance ? Il apparaît évident que, pour les jeux de données de taille importante, un être humain a de la difficulté à comprendre les régularités et les relations entre les attributs de ces données. De nombreuses représentations sont sensées apporter une réponse à ce problème : nous allons évoquer les réseaux de neurones, les arbres de décision ou les bases de classifieurs. Même si le choix de la représentation reste subjectif, il est sans doute préférable que l'algorithme découvre les structures les plus simples. Cette simplicité peut, par exemple, se calculer à partir du nombre de neurones dans un réseau, de nœuds dans un arbre de décision ou de classifieurs dans le cadre des SC.

La vitesse de traitement. Importante lors du traitement de larges jeux de données, elle permet aussi, si elle est faible, de rendre l'algorithme éligible au *tuning*, permettant d'optimiser certains para-

mètres à travers un grand nombre d'apprentissages. Ou encore de traiter des contenus dynamiques (flux vidéos, ...).

La généralité de la connaissance apprise. Distincte du critère de *généralisation*, il s'agit d'observer si différents algorithmes produisent le même type de connaissance, par exemple en comparant les attributs sélectionnés par un arbre de décision par rapport aux attributs privilégiés par les poids de connexion d'un réseau de neurones. Une connaissance qui possède des formalismes similaires ou *stables* quel que soit l'algorithme d'apprentissage offre une bonne garantie d'authenticité et de véracité.

Le déterminisme. Ce critère est valable pour un bon nombre d'algorithmes non déterministes, principalement les algorithmes évolutifs. Un algorithme est dit *déterministe* s'il produit les mêmes résultats de classification sur les exemples non appris – si ce n'est les mêmes classifieurs dans le cas des SC – après différents cycles d'apprentissage. Un algorithme évolutif *déterministe* offre la garantie que son aspect stochastique n'influence par les résultats de son apprentissage.

L'expansibilité dénote la capacité de l'algorithme à s'adapter à des jeux de données de grande taille. Elle se mesure habituellement par sa complexité exprimée en fonction d'un ou plusieurs paramètres spécifiques à l'algorithme ou aux données (par exemple, est-il exécuté en temps linéaire ou quadratique par rapport au nombre de canaux ou de pixels de l'image ?).

L'expertise externe. L'algorithme a-t-il besoin, à part le jeu d'apprentissage, d'une expertise externe pour apprendre ? Il peut s'agir de jeux de données supplémentaires intervenant au départ ou à une étape précise de l'algorithme. Certains systèmes dit *interactifs* sont dépendants de l'expert lors d'une phase de validation ou de paramétrage intermédiaire. Un autre point de vue peut être abordé concernant l'expertise externe. L'algorithme nécessite-t-il une connaissance poussée de son propre domaine pour être manipulé ? Par exemple, les algorithmes qui possèdent de nombreux paramètres sont souvent connus comme étant plus difficiles à comprendre et donc à maîtriser.

Les critères comme la vitesse de traitement ou l'expansibilité ne sont pas très importantes concernant les classifieurs eux-mêmes, car ces derniers sont, dans la plupart des cas, évaluables en temps constant en exploitation et donc linéairement dépendant de la taille du jeu de données.

Cette liste nous servira de fil conducteur pour ce chapitre, lors du parcours de l'existant, ainsi que de base de travail pour le reste de cette thèse, lors de la construction de nos propres représentations. Ils guideront notamment le choix des représentations des règles ainsi que le choix des méthodes et de leur implémentation. Nous commençons donc par évoquer les techniques permettant de construire des règles arborescentes.

1.3 Construction de règles arborescentes

1.3.1 Graphes d'induction

Ces méthodes produisent à partir d'un ensemble d'exemples classifiés, un arbre ou *graphe d'induction* permettant de relier une variable à prédire (comme la prévision d'un match) avec certains attributs explicatifs (le temps actuel, la forme des joueurs, la direction du vent, [Quinlan, 1986]). Leur utilisation dans des domaines comme l'aide au diagnostic médical, la gestion ou le marketing est due à leurs nombreuses qualités :

- ils permettent de créer un ensemble minimal de règles de décision,
- ils peuvent s'appliquer à des bases de données de grande taille,
- les résultats produits (graphes) sont facilement interprétables par un utilisateur ou un expert du domaine non spécialiste en informatique,
- ils sont capables de manier aisément des données hétérogènes de différents types : binaires, qualitatifs (c'est-à-dire pris dans un ensemble fini de modalités) ou quantitatifs (entiers ou continus que l'on doit discrétiser avec des méthodes statistiques).

Les travaux de [Quinlan, 1986] et [Quinlan, 1993] proposent des algorithmes relativement simples (respectivement ID3 et C4.5) : ils produisent des arbres dont chaque branche représente la valeur d'un attribut prédictif et chaque nœud correspond à une population d'individus qui satisfont toutes les valeurs des attributs placés entre ce nœud et la racine de l'arbre. Les valeurs des attributs peuvent être binaires, qualitatives ou continues. Pour traiter les valeurs continues, on utilise des algorithmes de discrétisation qui ont pour objectif d'obtenir un nombre de classes relativement faible et une bonne répartition statistique entre les classes. Voici comme illustration, l'algorithme C4.5, qui nécessite au préalable quelques définitions :

Soit $P = (p_1, \ldots, p_n)$ une distribution de probabilité des classes, alors l'entropie $\mathrm{I}(P)$ se calcule par :

$$\mathrm{I}(P) = -\sum_{i=1}^{n} p_i \log(p_i) \tag{1.1}$$

Si nous divisons une partition T sur la base de la valeur d'une variable exogène X en n ensembles T_1 à T_n alors l'information requise pour identifier la classe d'un élément de T est :

$$\mathrm{Info}(X,T) = \sum_{i=1}^{n} \frac{\Omega(T_i)\mathrm{I}(T_i)}{\Omega(T)} \tag{1.2}$$

où $\Omega(T)$ est le cardinal de l'ensemble T.

Le gain d'information dû à l'attribut X s'exprime donc par :

$$\mathrm{Gain}(X,T) = \mathrm{I}(T) - \mathrm{Info}(X,T) \tag{1.3}$$

Cette fonction a tendance à favoriser les attributs qui possèdent un grand nombre de valeurs. Pour compenser cet effet, Quinlan suggère d'utiliser à la place le GainRatio défini par :

$$\mathrm{GainRatio}(X,T) = \frac{\mathrm{Gain}(X,T)}{\mathrm{SplitInfo}(X,T)} \tag{1.4}$$

où SplitInfo(X,T) mesure la répartition des exemples selon l'attribut X. Une définition possible pour SplitInfo est :

$$\mathrm{SplitInfo}(X,T) = \mathrm{I}\left(\frac{\Omega(T_1)}{\Omega(T)}, \ldots, \frac{\Omega(T_m)}{\Omega(T)}\right) \tag{1.5}$$

où $\{T_1, \ldots, T_m\}$ est la partition de T par les valeurs de X.

C4.5 est présenté dans l'algorithme A1.

Cet algorithme renvoie un arbre dans lequel chaque nœud permet de partager l'ensemble d'exemples en fonction du critère qui semble le plus discriminant pour l'algorithme, d'après la fonction GainRatio. Puis une phase d'élagage emploie une fonction heuristique permettant d'estimer, même si en réalité elle n'est pas pertinente sur l'ensemble d'apprentissage, l'erreur d'un sous-arbre donné qui est alors remplacé par une feuille (simplification de l'arbre de départ). En exploitation, il ne reste plus qu'à parcourir cet arbre pour déterminer les règles de classification associées et les employer pour classer des exemples qui n'étaient pas dans la base initiale.

Enfin, citons la méthode SIPINA [Zighed et al., 1992] qui tente de répondre à certains inconvénients des méthodes arborescentes. Sans entrer dans le détail, il existe un certain nombre de difficultés spécifiques à ce type de méthode qui sont l'insensibilité à l'effectif (les classes sous-représentées ont le même poids que les autres) ou la préférence à la complexité (les méthodes construisent souvent des arbres contenant de nombreux nœuds). La méthode SIPINA généralise la notion d'arbre de décision en construisant plutôt un graphe, dit *graphe d'induction* dont la particularité est d'avoir un nombre de nœuds relativement restreint. Certaines méthodes de SIPINA ainsi que des méthodes de classification génériques sont intégrées dans un logiciel nommé TANAGRA [Rakotomalala, 2005], que nous avons utilisé dans nos expérimentations pour classer des données de télédétection par des méthodes inductives.

ALGORITHME A1
CRÉATION D'UN ARBRE DE DÉCISION PAR C4.5

$\rightsquigarrow$ PARAMÈTRES - R est l'ensemble des variables exogènes, C la variable endogène (à prédire) de valeurs $c_1, \ldots, c_n$, S l'ensemble des exemples d'apprentissage
$\rightsquigarrow$ RÉSULTAT - T, l'arbre de décision créé
$\rightsquigarrow$ FAIRE_NŒUD(E,V) crée un nœud renvoyant la valeur V si la condition E est vérifiée. Si V est lui-même un arbre, on renvoie la valeur de l'évaluation de cet arbre
$\rightsquigarrow$ COMPTE(S,C,v) renvoie le nombre d'enregistrements de S dont $C = v$
$\rightsquigarrow$ ENRACINE(T,N) enracine l'arbre N à la racine de T
$\rightsquigarrow$ ÉLAGUER(T) utilise une heuristique pour remplacer des sous-arbres par des feuilles afin de simplifier l'arbre T

si $S = \{\}$ ***alors***
  Renvoyer ERREUR
fin si
si COMPTE(S,C,v) $= \Omega(S)$ ***alors***
  $\rightsquigarrow$ Si C a la même valeur v dans S
  $T :=$ FAIRE_NŒUD(TRUE,v)
  Renvoyer T
fin si
si $R = \{\}$ ***alors***
  Trouver v tel que COMPTE(S,C,v) $\geq$ COMPTE(S,C,v') *pour toute valeur v' possible pour* C
  $T :=$ FAIRE_NŒUD(TRUE,v)
  Renvoyer T
fin si
Trouver D *tel que* GAINRATIO(D,S) $\geq$ GAINRATIO(D',S) *pour toute variable {D, D'} de* R
soit $\{d_i | i = 1, \ldots, n\}$ *l'ensemble des valeurs de la variable* D *et* n *leur nombre*
Trouver $\{S_i | i = 1, \ldots, n\}$ *l'ensemble des enregistrements de* S *dont la variable* D *vaut* d_i
$T := \{\}$
pour i *de 1 à* n ***faire***
  $N :=$ FAIRE_NŒUD($D = d_i$, C45($R - \{D\}$, C, S_i))
  $T :=$ ENRACINE(T,N)
fin pour
Renvoyer ÉLAGUER(T)

Algorithme 1: *Fonction* C45*(R,C,S)*

1.3.2 Algorithmes inductifs

De nombreux domaines d'étude créent fréquemment des masses d'informations gigantesques qui sont ardues à disséquer, à cause de la lourdeur des traitements. L'analyse de ces grandes quantités de données passe par la description de *concepts* sous-jacents permettant de résumer plusieurs enregistrements par une connaissance nouvelle, plus compacte, appelée *objet symbolique*. Certaines méthodes parviennent à cette abstraction des données de base en appliquant un processus de généralisation dirigé par deux critères : minimiser le volume des données et minimiser la perte d'informations induite [Bock et Diday, 2000] comme l'algorithme AQR ne créant qu'une seule règle de classification par classe à décrire. Cette méthode est célèbre depuis qu'elle a été testée avec succès sur des plants de soja. Nous pouvons aussi citer CN2 combinant les avantages d'AQR et d'ID3 [Clark et Niblett, 1989].

Les algorithmes correspondants sont beaucoup utilisés pour extraire des connaissances à partir d'une base de données. Pour satisfaire les critères évoqués dans le paragraphe précédent, ils utilisent des mesures de recouvrement (la généralisation d'une classe donnée doit recouvrir le maximum des individus de cette classe), d'homogénéité, de discrimination (minimiser l'erreur de classement des contre-exemples) et de densité (les individus de la classe doivent être regroupés dans le volume le plus faible possible). On y adjoint une méthode de spécialisation permettant de s'assurer que les individus singuliers sont éliminés : pour atteindre cet objectif, on cherche à trouver un bon compromis entre l'augmentation de la densité et la perte des individus extrêmes. Enfin, une méthode de décomposition comme la méthode DIV de [Chavent, 1998] finalise le traitement en divisant les classes en fonction d'un critère binaire, ce qui permet de construire des disjonctions de contraintes.

Ces méthodes sont souvent considérées comme étant *monothétiques*, car le choix du meilleur attribut dans la construction de ces arbres se fait toujours en le considérant seul, face à l'ensemble des classes à discriminer. D'autres méthodes dites *polythétiques* [Palma et al., 1997] ou poly-attributs prennent en compte la dépendance de certains attributs entre eux et donc les sélectionnent en blocs pour la construction des règles, ce qui permet de mettre en évidence la capacité de prédiction de plusieurs variables agissant simultanément. Citons, par exemple, des algorithmes symboliques d'exploration sélective de relations binaires comme le système CHARADE [Ganascia, 1987], ou des méthodes disjonctives parcourant l'espace des généralisations comme la stratégie de l'espace des versions [Mitchell , 1982] ou l'algorithme de l'étoile [Michalski, 1983]. Enfin, signalons que la qualité des techniques de généralisation est fortement dépendante du bruit présent dans les données, ainsi que des informations incomplètes ou incohérentes. Des algorithmes d'estimations des données manquantes se révèlent donc vite être nécessaires, par exemple l'algorithme MVI (ang., *Missing Value Imputation*, [Cao, 2001]).

1.4 Extraction de règles depuis les réseaux neuronaux

Dans cette section, nous allons nous intéresser à l'extraction de règles, dites *règles de production*, à partir de réseaux neuronaux à connaissance de type distribuée, et comparer les règles obtenues avec celles des systèmes d'apprentissage symbolique comme les arbres de décision ou les systèmes experts. Nous verrons notamment les algorithmes NNRE [Setiono et Liu, 1996], VIA [Thrun, 1995] et RULEX [Andrews et Geva, 1995] car ils sont de bons exemples des différentes approches existantes (décompositionnelle, pédagogique, ...). Notre propos n'est pas d'étudier ici la totalité des algorithmes disponibles pour extraire de telles règles depuis un réseau de neurones. De nombreux autres algorithmes existent et le lecteur intéressé pourra se référer à KBANN [Sordo, 1997], SUBSET [Towell et Shavlik, 1993] ou M-of-N [Setiono, 2000], pour les principaux utilisés.

La littérature a souvent comparé les avantages et les inconvénients de chacune de ces approches. [Osorio, 1998] a résumé dans un tableau les correspondances structurelles entre les réseaux de neurones et la base de connaissance d'un système d'apprentissage symbolique (voir tableau 1.1).

Les différences les plus courantes entre ces deux approches peuvent être résumées de la manière suivante :

- généralement, on considère un réseau de neurones artificiels comme une boîte noire (ang., *black box*). En effet, après l'apprentissage de deux problèmes diamétralement opposés, on peut obtenir

Modèle connexionniste	**Système symbolique**
Unités d'entrée	Faits (prémisses de base)
Unités de la couche cachée	Conclusions intermédiaires
Unités de sortie	Conclusion finales
Poids des connexions	Dépendances

TAB. 1.1 – Correspondances structurelles entre systèmes d'apprentissage symbolique et réseaux neuronaux.

deux réseaux de neurones dont les différences ne se situent qu'aux niveaux des poids de connexion, très difficiles à expliquer, surtout quand le nombre de neurones augmente (la topologie restant quand à elle générique : une couche d'entrée, une couche de sortie et une ou plusieurs couches cachées). D'autre part, les méthodes inductives à base d'arbres de décision sont connues pour produire un ensemble de règles plus facilement interprétables que celles produites à partir d'une matrice de poids de connexion [Shavlik et al., 1991],
- l'apprentissage classique des RN par rétro-propagation requiert un temps de calcul plus important [Dietterich et al., 1990],
- enfin, les deux approches ont une précision de prédiction relativement semblable, cependant la rétro-propagation donne parfois des résultats légèrement meilleurs [Quinlan, 1994].

De manière générale, les études effectuées sur les réseaux neuronaux artificiels (RN) ont exploré deux types de représentation de connaissances [Osorio, 1998] :

- Les représentations *localistes* : les concepts (c'est-à-dire les objets à apprendre) correspondent à des neurones spécifiques. Les réseaux à *zone d'influence* comme les RBF (ang., *Radial Basis Function*) sont des modèles localistes. L'intégration de concepts symboliques sera plus simple car chaque concept correspondra à un neurone. Les propriétés relationnelles entre les différentes unités pourront être mieux étudiées, notamment les relations contextuelles entre les concepts, les unités fortement liées, ... De plus, il est facile d'extraire une distribution probabiliste pour chaque concept. Ce type d'approche peut être intéressant en télédétection. Malheureusement, le codage du réseau est peu adapté à de gros ensembles de connaissances à cause de la prolifération des connexions entre unités.
- Les représentations *distribuées* : un ensemble de neurones est associé à chaque concept, et chacun de ces neurones participent à la représentation de ce concept. La connaissance est répartie au sein des unités et supporte le bruit : si la valeur individuelle d'un neurone change, la représentation globale du concept n'est pas déformée. Les principaux algorithmes d'apprentissage et d'extraction de règles présentés dans la littérature sont plutôt concernés par cette approche. Les réseaux de type PMC ont une approche distribuée. Ce type de réseau est capable de généraliser plus facilement (la valeur d'un exemple particulier ne domine pas l'apprentissage d'un concept) et le nombre d'unités nécessaires pour le stockage de l'information est moins important (avec n unités, on représente n concepts *localistes* et 2^n concepts *distribués* dans le cas binaire). Malheureusement, la représentation de relations arbitraires entre concepts est plus délicate.

De nombreux travaux ont été entrepris pour tenter d'obtenir à partir des réseaux de neurones des règles aussi simples et interprétables que ceux des arbres de décision. Ces travaux ont donné lieu à des algorithmes de recherche explorant les connexions et les poids des neurones. Comme le temps de calcul suit rapidement une courbe exponentielle, des méthodes de simplification interviennent pour déduire à partir de la valeur des connexions une réduction de l'espace à explorer. Par exemple, dans la méthode M-of-N, les valeurs d'entrées sont restreintes à -1 ou 1, et la fonction de transition utilisée est une tangente hyperbolique, permettant de limiter l'activation des neurones à 1 ou 0 [Setiono, 2000]. Malheureusement, les capacités d'apprentissage et de modélisation du réseau risquent d'être limitées par des problèmes nécessitant une définition uniforme de la fonction de transition.

1.4.1 L'algorithme NNRE

Obtenir des règles humainement interprétables constitue dans nos travaux un objectif très important : ces règles représentent une forme de connaissance et celle-ci se doit d'être facilement vérifiable par des experts de telle manière à pouvoir être étendue, retransformée voire critiquée et rejetée, ou comprise pour être réutilisée dans d'autres domaines. Parmi toutes les méthodes d'extraction de règles à partir de réseaux neuronaux, il en existe une qui met particulièrement l'accent sur la production de règles symboliques précises. L'algorithme NNRE (ang., *Neural Network Rule Extraction*) fonctionne globalement selon le modèle suivant [Setiono et Liu, 1996] :

1. Pour obtenir le bon nombre de couches cachées du réseau, deux approches sont possibles, consistant respectivement à augmenter le nombre de couches d'un réseau minimaliste ou à diminuer le nombre de couches d'un réseau surchargé. NNRE, dont l'objectif est de créer des règles simples, utilise la seconde approche puisqu'elle permet d'éliminer les neurones inutiles.
2. Le réseau est ensuite élagué. Cette fois-ci, c'est les connexions inutiles qui sont visées, en fonction d'un critère basé sur leurs poids maximum. Après chaque élagage, le réseau est ré-entraîné sur la base d'exemples et le processus d'élagage est stoppé lorsque le taux de classifications correctes descend en dessous d'un certain seuil. On peut noter que plus le seuil sera élevé, plus les règles contiendront un nombre élevé de parties conditions lorsqu'elles seront extraites.
3. Les valeurs d'activation des couches cachées, si elles sont continues, sont ensuite discrétisées avec un algorithme de clustering quelconque de telle sorte à pouvoir produire des règles de type *If ... Then ... Else* simples munies des opérateurs de comparaison classiques ($<$, $>$, ...).
4. Enfin, les règles proprement dites sont extraites en utilisant les attributs les plus fréquents caractérisant chaque classe (un ensemble d'exemples étiquetés de la même façon) pour construire les conditions des règles.

Produisant des règles symboliques, on peut se questionner sur ce qu'apporte cet algorithme par rapport à C4.5, une méthode connue pour créer des arbres de décision élagués et des règles ayant une bonne capacité de généralisation [Quinlan, 1993]. Les résultats seraient intéressants à analyser notamment pour les critères de lisibilité et de performance. La littérature nous documente sur une telle comparaison dans [Setiono et Liu, 1996] qui s'est faite entre autres sur des données médicales dans lesquelles on doit prédire le caractère malin d'une molécule en fonction d'un certain nombre de critères C_n. Les tableaux 1.2 et 1.3 permettent à la fois de se faire une idée sur le formalisme des règles générées par les deux approches et de comparer leur efficacité.

R_1	**If** $C_1 < 7$ **And** $C_2 < 8$ **And** $C_3 < 3$ **And** $C_8 < 9$ **Then** *bénin*
R_2	**If** $C_1 < 7$ **And** $C_2 < 8$ **And** $C_6 < 3$ **And** $C_8 < 9$ **Then** *bénin*
R_3	**If** $C_2 < 8$ **And** $C_3 < 3$ **And** $C_6 < 3$ **And** $C_8 < 9$ **Then** *bénin*
Règle par défaut	*malin*

TAB. 1.2 – Extraction de règles pour la détection de molécules cancéreuses à partir d'un réseau de neurones (NNRE).

Les résultats sur l'ensemble de test sont très similaires pour les deux approches (95% d'exemples correctement classifiés). D'après Setiono [Setiono et Liu, 1996], les règles produites par NNRE possèdent généralement plus de tests qu'avec C4.5, mais sont moins nombreuses. L'utilisation d'un réseau neuronal dans NNRE et d'une couche d'entrée mêlant tous les attributs en même temps permet d'explorer les différentes combinaisons d'attributs de manière plus systématique qu'avec C4.5. Ainsi, les règles produites par NNRE sont plus longues mais aussi plus spécifiques.

R_1	**If** $C_1 < 7$ **And** $C_2 < 3$ **Then** *bénin*
R_2	**If** $C_1 < 7$ **And** $C_4 < 2$ **Then** *bénin*
R_3	**If** $C_2 \geq 5$ **Then** *malin*
R_4	**If** $C_6 \geq 9$ **Then** *malin*
R_5	**If** $C_1 \geq 7$ **Then** *malin*
R_6	**If** $C_4 \geq 4$ **Then** *malin*
Règle par défaut	*bénin*

TAB. 1.3 – Extraction de règles pour la détection de molécules cancéreuses à l'aide de C4.5.

1.4.2 L'algorithme RULEX

L'algorithme RULEX [Andrews et Geva, 1995] est un algorithme d'extraction de règles à partir des réseaux neuronaux du type de ceux qui sont appris à l'aide de l'algorithme contraint de rétro-propagation du gradient (ang., *Constrained Error Back Propagation* ou CEBP), dont certains poids sont fixés durant l'apprentissage. Selon la classification de Craven [Craven et Shavlik, 1994] ou d'Andrews [Andrews et al., 1995], l'algorithme RULEX correspond à une *approche décompositionnelle*. Cette approche considère lors de l'extraction des règles la topologie interne du réseau, c'est-à-dire les valeurs caractérisant l'activation des neurones cachés ou non, ainsi que les valeurs des connexions entre les neurones. Des règles sont créées séparément pour chaque neurone des couches cachées et un ensemble complet de règles est ensuite dérivé lors d'une phase finale.

L'algorithme fonctionne de la façon suivante :

1. Chaque neurone dans le réseau est une unité de réponse locale (ang., *Local Response Unit* ou LRU) semblable à celle des réseaux dits RBF (ang., *Radial Basis Function*), à la seule différence que les fonctions d'activation sont des sigmoïdes plutôt que des gaussiennes.
2. Le réseau est appris en ajustant les centres, les largeurs et les distances des *pics* (correspondant à l'intersection des fonctions sigmoïdales) afin de réduire l'erreur de la couche de sortie.
3. Ensuite, les règles sont extraites en encodant directement la réponse de chaque unité cachée. Les règles sont de la forme suivante :

$$\textit{If}\ \text{Pic}_1\ \text{actif AND ... AND Pic}_n\ \text{actif}\ \ \textit{Then}\ \ \text{Class}_c$$

4. Enfin, une phase de raffinement est utilisée pour augmenter la compréhensibilité de la base de règles obtenue. Trois opérations de simplification sont employées. Si toutes les valeurs d'attribut sauf une sont représentées dans une règle, une règle formée de la *négation* de la valeur absente est utilisée à la place. Si un pic reste actif malgré toutes les valeurs possibles pour un attribut, cet attribut est *éliminé* car il ne contribue pas à la discrimination des exemples. La troisième opération, l'*absorption*, concerne la combinaison de plusieurs règles redondantes.

Cet algorithme peut s'appliquer sur des données discrètes, continues ou mixtes. Un autre avantage est le caractère local des LRUs : chaque neurone peut être encodé dans une règle indépendante des autres neurones. Cependant, cet algorithme possède aussi un certain nombre d'inconvénients. Tout d'abord, les LRUs ne doivent pas se chevaucher dans leur espace d'activation. Sinon, les règles extraites ne pourront pas traiter les exemples qui prennent leurs valeurs dans cet espace de chevauchement [Andrews et Geva, 1996]. Andrews suggère d'utiliser à la place un autre formalisme de règles, à base de logique floue. De plus, le caractère local de ces unités empêche le traitement de certains problèmes qui peuvent être décrits avec des règles globales. Un autre inconvénient est le fait que RULEX nécessite une architecture de réseau et un apprentissage bien particulier, ce qui le restreint aux problèmes que ces réseaux peuvent effectivement apprendre. Malgré tout, cet algorithme est connu pour produire des règles fiables, performantes et compréhensibles. Sa rapidité présente un intérêt dans le traitement de larges bases de données, notamment en télédétection. Par exemple, RULEX a été comparé à l'algorithme *CasCor* lors d'un problème de classification d'eau et de forêt [Nayak et al., 1997]. RULEX a montré une performance supérieure

pour un réseau plus petit (architecture 3 : 5 : 1 contre une architecture 15 : 2 : 1). Nous pouvons aussi citer le système hybride GYAN [Nayak, 2000], incorporant divers algorithmes d'extraction de règles, dont RULEX, dans l'étude de problèmes du monde réel (télédétection, cancérologie, cardiologie).

1.4.3 L'algorithme VIA

L'algorithme VIA (ang., *Validity Interval Analysis*) [Thrun, 1993; Thrun, 1995] est un algorithme d'extraction de règles qui correspond à une *approche pédagogique* (ang. *Black-box approaches*) [Andrews et al., 1995]. Ce type d'approche concerne les algorithmes d'extraction qui ne tiennent pas compte des unités de la couche cachée mais uniquement des valeurs obtenues par les neurones de sortie en fonction des valeurs des neurones d'entrée. En ce sens, les méthodes pédagogiques n'imposent pas de pré-requis sur la topologie du réseau et sur l'algorithme d'apprentissage, et peuvent être considérées comme des méthodes génériques.

L'algorithme fonctionne de la manière suivante :

1. Chaque neurone x_k du réseau (ou un sous-ensemble de neurones, par exemple, uniquement les neurones d'entrée et de sortie) est associé à un intervalle de validité $[a_k; b_k]$. Cet intervalle représente une restriction sur le domaine de valeurs pour lesquelles le neurone en question peut être activé. Ce domaine de valeurs peut être défini par l'utilisateur ou par l'algorithme ([0 ;1] par défaut).

2. Lors d'une phase dite *en avant* (ang., *forward phase*), ces intervalles sont calculés pour tous les neurones en tenant compte du comportement du réseau lors de la présentation des exemples d'apprentissage. La fonction d'activation est utilisée pour obtenir les bornes minimales et maximales des valeurs d'activation des différents neurones. Notamment, les intervalles des neurones qui ne sont jamais activés sont supprimés.

3. Lors d'une phase dite *en arrière* (ang., *backward phase*), ces intervalles sont raffinés. Pour chaque neurone x_k, l'intervalle de validité correspondant $[a_k; b_k]$ est réduit si le minimum (respectivement le maximum) de la valeurs des neurones de la couche précédente connectée à x_k, est supérieure (respectivement inférieure) à la borne a_k (respectivement b_k).

4. Les deux phases précédentes sont répétées jusqu'à obtenir la convergence de l'algorithme. Ensuite, les intervalles sont analysés et utilisés pour dériver un ensemble de règles. Ces règles sont de la forme *If ... Then*. Chaque condition est formée d'une conjonction de contraintes booléennes sur les valeurs des neurones d'entrée. Le résultat de la conditionnelle est une expression donnant l'intervalle de validité d'un neurone de sortie.

L'un des avantages concerne la correction des règles extraites : l'algorithme peut prouver que les règles sont correctes en les inversant (R est remplacé par $\overline{R}$) et en cherchant les contradiction logiques pour toute valeur possible des neurones d'entrée (notamment $\overline{R}$ ne doit jamais être vraie). C'est ce processus qui permet à l'algorithme d'affiner des règles générales (« *tout est de la classe C* »), jusqu'à les rendre suffisamment spécifiques pour qu'elles soient correctes. L'un des principaux inconvénients reprochés à cette méthode découle justement de la remarque précédente : le nombre de règles est souvent élevé et la recherche de règles correctes peut les rendre trop spécifiques. Par exemple, dans un problème de bras de robot, seulement 84% de l'espace des valeurs d'entrée ont pu être couvertes par 10000 règles [Thrun, 1995]. À notre connaissance[1], nous n'avons pas trouvé d'applications de cet algorithme en télédétection. Malgré le fait que les règles produites soient simples, ce qui est utile pour l'explication de connaissance dans ce domaine, il semblerait, selon Thrun lui-même[2], que le traitement de larges bases de données est très difficile.

[1] Selon la bibliographie publiée par S. Thrun.

[2] « *The process of testing each input pattern [...] is usually computationally intractable for larger domains.* » [Thrun, 1993]

1.5 Découverte génétique de règles

Connus sous les noms d'algorithmes évolutifs ou algorithmes évolutionnaires, ils sont sensés imiter les mécanismes darwiniens de l'évolution naturelle. Deux notions sont au moins communes à tous les algorithmes évolutifs, les différenciant ainsi des méthodes non évolutionnaires :

- basés sur la notion de *génération*, ce sont des algorithmes itératifs. Possédant de nombreux paramètres contrôlant l'évolution de l'apprentissage ou le comportement des opérateurs génétiques, ils se classent parmi les algorithmes les plus riches en paramètres. Une génération, ou *cycle*, crée une multitude d'*individus* (appelés selon le contexte *chromosome*, *classifieurs* ou *animat*) représentant chacun une solution au problème à résoudre. L'ensemble de ces individus, appelé *population*, est manipulé de telle sorte que l'on puisse finir par trouver, parmi une partie de ces individus, une réponse quasi optimale au problème posé,
- l'évolution de l'algorithme est matérialisée par le fait que la population des individus remplace à chaque génération un sous-ensemble de la population précédente. Chaque population est destinée à contenir un ensemble conséquent de solutions potentielles indépendantes, ce qui conduit ces algorithmes à être hautement parallélisables et leur permet d'exploiter l'efficacité du calcul distribué.

1.5.1 Apprentissage par algorithme génétique

Les algorithmes génétiques (AG), proposés par Holland [Holland, 1975] puis développés par Goldberg [Goldberg, 1989], sont des méthodes d'optimisation stochastiques. Ils peuvent résoudre des problèmes d'optimisation de fonctions, de régression ou de classification et sont souvent meilleurs en temps de calcul que les algorithmes déterministes pour les problèmes NP-complets. Le but est de faire évoluer une population d'individus (chromosomes) représentant les solutions au problème à résoudre, le tout guidé par une fonction *fitness*. Au départ initialisée de manière aléatoire, la population est brassée génétiquement par trois types d'opérateurs. Le premier est nommé croisement (ang., *crossing-over*), et exploite la population courante en échangeant, entre deux individus parents, une partie de leur patrimoine génétique. Les croisements peuvent être à un point, à n points ou uniforme. Le second est nommé mutation, et explore l'espace de recherche en substituant l'un des gènes d'un chromosome par un autre. Le troisième opérateur est la sélection qui intervient à plusieurs niveaux de l'algorithme : il fournit le matériel génétique à croiser ou à muter aux opérateurs correspondant (opérateur de sélection pour les parents) et intervient dans le recyclage générationnel (opérateur de remplacement pour le recyclage). Concernant les représentations des chromosomes, on peut citer les encodages par des séquences de valeurs binaires (la plus classique), entières ou continues, ou les encodages arborescents.

De très nombreux travaux ont été menés en algorithmique génétique concernant le traitement d'images ou la télédétection. Nous ne pouvons pas tous les citer, mais nous pouvons donner un aperçu des plus intéressants, notamment concernant la gestion du bruit dans les images, soit au niveau du capteur [Daida et al., 1995], soit au niveau des objets [Ozcan et Mohan, 1997]. Les deux approches sont décrites ci-dessous.

En analyse d'images topographique, [Daida et al., 1995] a utilisé une hybridation de deux techniques, utilisant pour l'une un algorithme génétique, pour l'autre une intégration par transformée de Fourier, chacune prédisant une partie des coefficients d'une série de Fourier pour la reconnaissance de la microtopographie de la surface de l'eau. Cette méthode a donné, d'après les auteurs, des résultats intéressants malgré la présence de pics de dépassement de capacité du signal, de bruits aléatoires, d'artefacts-bulles, donnant aux données un caractère non-linéaire.

Souvent la détection d'objets dans les images satellitaires est perturbée par le fait que ces objets sont partiellement occultés par d'autres objets présents dans l'image ou par le bruit atmosphérique. Un algorithme génétique, mis au point par Ozcan [Ozcan et Mohan, 1997], a été appliqué à la reconnaissance de ce genre d'objets (ang., *shape matching*). Le génome représente une chaîne dérivée d'une grammaire conçue pour être flexible et indépendante à la fois de la taille et de la rotation des formes. La grammaire, ressemblante au langage LOGO [Logo, 2005], permet de déterminer le contour d'un polygone en utilisant

des informations relatives aux segments déjà évalués. Des tests ont été effectués jusqu'à 25% de niveau de bruit et l'algorithme prédisait les contours exacts dans un cas sur deux.

Dans le domaine de la télédétection pure, on peut citer les travaux publiés dans [Liu et al., 2004] concernant l'optimisation génétique d'individus représentant un réseau de neurones, utilisant la rétro-propagation comme pression de sélection. L'algorithme génétique GA-MLP (ang., *GA-based Multi-Layer Perceptron*) a été appliqué sur des images SPOT[3] et CBERS[4]. La performance moyenne de l'algorithme a été montrée comme étant supérieure à un algorithme de type PMC simple, tout en donnant le meilleur réseau en moins de 15 générations génétiques au lieu de 3000 époques. Malgré tout, l'algorithme ne peut proposer une explication intelligible de la connaissance découverte par les réseaux de neurones (vus comme des *boîtes noires*). À l'inverse, l'étude de [Brumby et al., 2000], présente un chromosome dont chaque gène est une référence à une opération primitive de traitement d'images (médian, combinaison linéaire, opérateur de texture de Laws, ...). Les primitives ont été choisies de telle sorte qu'elles puissent à la fois traiter de l'information spectrale comme spatiale, car les classes d'étude s'étendent sur une large portion d'image. La détection d'objets urbains (routes, bâtiments) est moins performante qu'attendue par rapport à des objets spectraux plus simples (comme l'eau), d'après les auteurs, à cause d'un jeu d'apprentissage de taille trop faible.

La recherche en algorithmique génétique porte sur l'optimisation de modèles existants simples, soit statistiques (réseaux de neurones, primitives de classification) soit mathématiques comme par exemple les modèles GSM[5] dans lesquels on optimise les termes dans les équations ou l'ordre du modèle [Samadzadegan et al., 2000].

Plus récemment, [Chemin et al., 2005] s'est intéressé à la découverte d'un modèle de prédiction de l'évapotranspiration de cultures en comparant deux images satellitaires prise dans un intervalle de temps relativement faible. Comme ce type de modèle est clairement non-linaire, à cause des nombreux paramètres environnementaux qui interviennent dans l'équation, un algorithme génétique a été étudié, ainsi qu'une fonction *fitness* adaptée, basée sur un modèle atmosphérique comme pression sélective. Enfin, signalons une étude originale sur la localisation de l'oeil des cyclones depuis l'imagerie satellitaire [Yip et Wong, 2004]. La rapidité de prédiction étant de rigueur pour ces cas, un algorithme génétique a été préféré face à la lourdeur des catégorisations habituelles, afin de donner une solution précise au problème en 12 secondes. Les individus génétiques modélisant des paramètres spécifiques de la spirale cyclonique ont permis l'amélioration de la précision par rapport aux algorithmes existants dans ce domaine.

1.5.2 Apprentissage par programmation génétique

En programmation génétique (PG), le principe est strictement le même qu'en algorithmique génétique, mais les individus sont ici des fonctions, par exemple encodées en LISP [Koza, 1992]. Les expressions LISP sont des expressions symboliques permettant de traduire les fonctions sous forme de listes imbriquées (aussi communément représentées sous formes d'arbre syntaxique), par exemple :

$$fonction\ (argument_1,\ argument_2,\ \ldots,\ argument_n) \tag{1.6}$$

Dans les premiers travaux, Koza définit un ensemble de *fonctions primitives* et de *terminaux*, constituants les seuls termes possibles de ces expressions, ainsi qu'un type de retour unique. Les *fonctions primitives* sont des fonctions booléennes, arithmétiques ou des structures de contrôle (conditionnelle, boucle, ...). Les *terminaux* sont des variables d'entrées, des constantes ou des fonctions sans argument mais avec effet de bord. Le mécanisme d'évolution par croisement consiste à échanger des sous-arbres, éventuellement associés à une pondération permettant l'échange plus rare de feuilles que de sous-arbres de taille plus importante. Les mutations correspondent à la modification de la valeur ou de la fonction d'un nœud, et la fonction d'évaluation assure une pression sélective sur les individus.

[3] *Satellite Pour l'Observation de la Terre*

[4] Ang., *China-Brazil Earth Resources Satellite*

[5] Les *Generic Sensor Models* sont des modèles mathématiques de capteurs ou de caméras, dont les mesures sont interprétées comme des polygones, sous-ensembles du plan d'étude, à modéliser.

La programmation génétique (PG) est surtout utilisée pour la détection de formes. On peut notamment citer les travaux de Harris [Harris et Buxton, 1996] ou de Daida [Daida et al., 1996] dans lesquels un individu est un programme s'exécutant sur une fenêtre 3x3 ou 5x5 de l'image et devient un détecteur de la différence de contraste entre deux pixels de cette fenêtre. D'autres résultats ont été obtenus en détection de contour, en utilisant le langage de spécification EASEA [Collet et al., 2000; Bolis et al., 2001] pour créer des animats-détecteurs.

Cette technique a déjà été utilisée dans la classification d'images et le traitement d'images satellitaires. On peut citer [Robilliard et Fonlupt, 2001], [Valigiani et al., 2004] ou [Fonlupt et Robilliard, 2000] qui ont utilisé cette approche pour la résolution de ce que l'on nomme le *problème inverse* PAR (ang., *Photosynthesis Available Radiation*). Ce type de problème cherche à modéliser une fonction représentant certaines caractéristiques du milieu étudié, ici le nombre de photons disponibles pour la photosynthèse en milieu marin, grâce à l'étude du signal perçu par les capteurs. Les auteurs présentent la première application de la PG au problème PAR. Avec une population de 5000 individus, les opérateurs $(+,-,*,/)$ et une profondeur d'arbre maximale de 10, les auteurs obtiennent un résultat corrélé à 81% avec ceux d'un algorithme employé par la NASA [Robilliard et Fonlupt, 2001]. Comme montré dans l'article, sans un algorithme adapté, cette technique est très sensible au sur-apprentissage. De plus, l'application du modèle à des données bruitées, comme c'est souvent le cas en télédétection, reste relativement difficile.

Ross [Ross et al., 2002] se donne pour objectif d'identifier la présence de trois minéraux distincts à l'aide du vecteur de réflectance d'un pixel donné sur des images hyperspectrales. Le langage utilisé est plus puissant que dans le cas précédent : en plus des opérations de base ($+$, $-$, $*$, $/$, $<$, $\leq$, $>$ et $\geq$), des opérateurs *min*, *max*, *If ... Then ... Else* ainsi que d'autres calculant la moyenne et la déviation standard sont employés. Les auteurs ont pu obtenir des classifieurs capables de fonctionner sur des zones dans lesquels les minéraux sont mélangés. Plutôt que de *référencer* chaque signature spectrale, la PG a permis de *caractériser* la signature d'un minéral dans le contexte des autres signatures. Néanmoins, aucun résultat n'est présenté sur la capacité de généralisation de tels classifieurs, par exemple avec des tests sur d'autres zones ou d'autres types de région.

1.6 Découverte de classifieurs par renforcement

Un système par renforcement est un système d'apprentissage supervisé qui fonctionne en collaboration avec un environnement externe, représentant le problème à résoudre [Booker et al., 1989]. Le système communique avec l'environnement par l'échange de *messages* et s'améliore en percevant des pénalités ou des récompenses déterminées par son comportement au sein de cet environnement. Les messages permettent au système de *lire* l'environnement par ses *détecteurs* ou *capteurs* et d'*agir* sur cet environnement par ses *effecteurs*. Le fait que le système n'a besoin que d'une réponse positive ou négative de l'environnement constitue la part de *renforcement* de l'apprentissage (*apprentissage par renforcement*).

1.6.1 Le cas des réseaux de neurones

Dans le cadre des réseaux de neurones, on peut signaler un algorithme important qui est celui de l'apprentissage par pénalité-récompense (ARP, Associative Reward Penality) de Barto, Anderson et Anandan [Barto et al., 1985]. Les deux blocs de base de cet algorithme sont :

- le bloc de recherche associative (ASE, Associative Search Element), qui utilise une méthode stochastique pour déterminer les relations qu'il peut y avoir entre les entrées et les sorties du réseau,
- le bloc d'évaluation adaptatif (ACE, Adaptative Critic Element), qui apprend à donner une prédiction correcte de la future récompense ou punition.

La réponse externe est un signal binaire, valant 0 si le système est dans le domaine de validité, -1 sinon. Appliqué à la télédétection, le domaine de validité est, par exemple, une image classifiée par un expert. Concernant la réponse, le système reçoit la valeur 0 lorsque la classe trouvée correspond à celle de l'expert et -1 sinon. Le principe général se découpe en trois étapes :

- effectuer une propagation dans le réseau pour obtenir une décision de classification ainsi que la prédiction de la récompense correspondante,

– calculer l'erreur entre la prédiction et la récompense finalement obtenue,
– réajuster les poids des connexions en conséquence.

Barto et Anandan prouvent dans [Barto et Anandan, 1985] la convergence dans le cas de sorties binaires représentant un ensemble de motifs linéairement indépendants. Cependant, il semblerait qu'il s'agisse d'un système difficile à mettre en place, à cause des nombreux paramétrages à effectuer.

1.6.2 Le cas des systèmes de classifieurs

Les systèmes de classifieurs (SC, [Holland, 1975; Goldberg, 1989]) comptent parmi les méthodes évolutives les plus anciennes puisqu'on en trouve trace dans [Holland et Reitman, 1978] et même dans [Holland, 1971] si l'on en croît l'historique de [Wilson et Goldberg, 1989]. Ils permettent de résoudre des problèmes combinatoires difficiles (NP-complets) au même titre que les algorithmes évolutionnaires, les animats[6] ou les colonies de fourmis. Comme eux, ils sont évolutifs et constituent une technique d'apprentissage automatique.

Les systèmes de classifieurs manipulent des *classifieurs*, c'est-à-dire des règles lisibles par un être humain, indiquant pour une situation donnée le comportement que doit suivre le système. Un exemple simple de classifieur est le suivant : « **Si** <case Nord> contient <nourriture> **Alors** <avancer Nord> ». La partie « <case Nord> contient <nourriture> » est nommée <*condition*> et la partie « <avancer Nord> » est appelée <*action*>. Le système complet manipule un ensemble de classifieurs (appelé *population* dans la terminologie génétique ou *base de classifieurs* dans la terminologie des SC). Un SC évolue dans un *environnement* externe au système (le problème à résoudre) : dans l'exemple donné, l'environnement est un labyrinthe. Chaque classifieur de la base est supposé correspondre à un certain nombre d'événements provenant de l'environnement et y réagir. Lorsqu'un classifieur est sélectionné car sa condition correspond à l'événement courant de l'environnement, on parle d'*activation* (ang., *matching*) de ce classifieur. Celui-ci est alors dit *actif* ou *activé*. L'opérateur qui calcule cette activation est nommé *opérateur d'activation*.

Il n'y actuellement en littérature aucun article sur les SC concernant directement la télédétection. On peut toutefois citer quelques études approfondies en classification de problèmes réels (ang., *real-world problem*). Llorà [Llorà et Goldberg, 2003] étudie certaines hypothèses de généralisation par rapport au sur-apprentissage en présence de données bruitées. Bernadó-Mansilla [Bernadó-Mansilla et Garrell-Guiu, 2003] présente un système nommé UCS qui construit directement une liste de classifieurs activés de manière plus efficace. En moyenne, la performance du système présenté est comparable à XCS [Wilson, 1995], un SC très connu dans ce domaine que nous allons décrire dans le chapitre suivant, tout en donnant des bases de connaissance comprenant moins de règles (on observe une réduction de l'ordre de 40 à 60%). Néanmoins le système ne peut manipuler que des chaînes binaires. Des efforts ont été produits pour améliorer la qualité de classification pour des problèmes réels. Le système *EpiCS* [Holmes et al., 2000] propose une nouvelle méthode de sélection des exemples d'apprentissage, adaptée à partir de la technique du *bootstrapping*, visant à inclure des exemples aléatoires pour augmenter la capacité de généralisation de l'algorithme. Une autre étude [Bagnall et Cawley, 2003] a été réalisée avec l'une des bases de l'UCI KDD[7] concernant la couverture forestière (10 variables continues, 2 attributs qualitatifs à découvrir comprenant respectivement 4 et 40 valeurs distinctes, 44 attributs prédictifs binaires, 581000 exemples). L'algorithme XCS a été comparé à huit autres classifieurs (deux à vecteur support, trois connexionnistes et trois inductifs). Il a été montré qu'XCS pouvait battre les deux classifieurs à vecteur support (comprenant respectivement un noyau linéaire et un noyau gaussien) à condition qu'on y apporte quelques perfectionnements. Ici, pour chaque exemple du jeu de données, seuls 2 attributs valaient « 1 » sur les 44 attributs présents. Lors du croisement, il était alors possible de produire des classifieurs incorrects qui n'étaient activés par aucun exemple. Les auteurs ont alors modifié l'opérateur en conséquence pour ne produire que des *enfants* (ang., *offsprings*) valides. Le choix de la représentation est donc fortement dépendante du problème à résoudre. On peut souligner que dans notre cas, ce problème d'attributs *factices*

[6] Les animats (abréviation d'animal et robot) appartiennent au paradigme nommé *vie artificielle* et non au calcul évolutif. Cet axe de recherche possède ses propres conférences et ses propres notions comme par exemple la robotique évolutive (ang., *evolutionary robotics*).

[7] Knowledge Discovery in Databases Archive, University of California Irvine, http ://kdd.ics.uci.edu/

n'entre pas en ligne de compte car tous les attributs spectraux sont renseignés.

Nous citons enfin quelques-uns de nos travaux, consacrés à l'application des systèmes de classifieurs en télédétection, que nous détaillerons au cours de cette thèse [Korczak et Quirin, 2003a; Quirin et al., 2004; Quirin et Korczak, 2005a; Quirin et al., 2005].

1.7 Conclusion

Nous venons de retracer ici de nombreuses méthodes d'apprentissage supervisés, ainsi que le cadre de la découverte de règles de classification pour le traitement d'images de télédétection. Peu de travaux ont été consacrés à l'apprentissage par renforcement en télédétection, peut être à cause de leur lourdeur supposée (ils intègrent une composante AG à eux seuls) ou à cause du manque d'intérêt dans ce domaine. Cependant, ces systèmes nous intéressent profondément. Le chapitre suivant est, pour cette raison, consacré entièrement à la description en profondeur des systèmes de classifieurs. Nous pensons que pour faire face à la complexité des données, une méthode évolutive semble plus robuste que les autres. Concernant les critères de représentation de connaissance et de compréhensibilité, nous justifions dans cette thèse, à chaque fois que cela est nécessaire, les choix que nous avons retenus.

Chapitre 2

Fondements des systèmes de classifieurs

2.1 Introduction

Ce chapitre a pour double-rôle de présenter un état de l'art de ces systèmes, peu connus et peu usités même 30 ans après leur création en 1975, mais aussi d'installer les principes fondamentaux des systèmes de classifieurs qui sont utiles à la fois pour l'état de l'art et pour le reste de cette thèse, car de nombreuses notions sont nécessaires pour appréhender les améliorations apportées depuis cette date. Cette partie, plus théorique, n'est que peu ponctuée de références à leur application en télédétection à cause du calme plus prononcé de la recherche dans ce domaine. Pour preuve, aucune application en traitement d'images ou en télédétection n'a été présentée lors du principal cycle de conférences sur les systèmes de classifieurs IWLCS[1].

Le chapitre est structuré de la façon suivante. Nous posons dans un premier temps les fondements et le principe des systèmes de classifieurs. Nous détaillons ensuite l'interaction de l'algorithmique génétique au sein du système qui permet l'émergence de nouveaux classifieurs, ainsi que leur représentation classique. Les composants principaux de ces systèmes sont ensuite décrits, ainsi qu'un opérateur particulier, le *covering operator*. Enfin, nous donnerons quelques exemples de tels systèmes, notamment le système XCS, les systèmes flous, les systèmes à base de S-classifieurs et nous terminerons par présenter leurs principales améliorations.

2.2 Systèmes LCS

Il existe une classe particulière de SC qui possèdent la capacité d'*apprentissage symbolique*, qui nous intéressent dans le cadre de cette thèse et qui représentent l'un des points importants de la recherche dans ce domaine actuellement. Cette classe de SC sont nommés LCS (ang., *Learning Classifier Systems*).

Holland [Holland, 1986] résume ainsi les caractéristiques qui distinguent les LCS des SC :

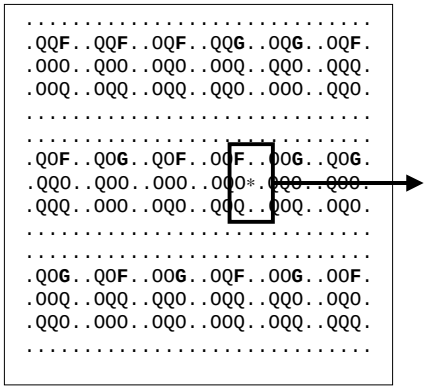
- L'activation de chaque classifieur dépend de paramètres qui sont modifiés dans le temps, pour mimer l'acquisition d'expérience. Cette expérience est apportée par l'environnement dont les classifieurs perçoivent une *récompense* (ang., *feedback*), c'est-à-dire une pénalisation ou une gratification en fonction de leur comportement dans l'environnement.
- Les classifieurs sont regroupés dans une base qui est modifiée au cours du temps, par exemple par génération de nouveaux classifieurs ou par suppression, ajout ou recombinaison des parties <*condition*> et <*action*> des classifieurs existants. Cette génération de nouveaux classifieurs est réalisée à l'aide d'un algorithme génétique.

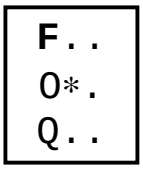
Les LCS se démarquent des autres techniques d'apprentissage automatique comme les réseaux de neurones ou la programmation génétique par leur aptitude à l'apprentissage collaboratif : un certain nombre de classifieurs vont servir à créer une chaîne complète dont l'ensemble constituera une solution

[1] International Workshop on Learning Classifier Systems. Les huit conférences de 1992 à 2005 ont été examinées

possible à un problème, se passant l'information d'un maillon à l'autre sous forme de messages à interpréter. Concernant le système XCS, ces notions sont précisées dans les travaux de Butz [Butz et Wilson, 2002]. Selon la taille de cette chaîne, l'apprentissage, lié au type d'environnement étudié, sera nommé *single-step* ou *multi-step*.

Dans un apprentissage de type *single-step*, le but est de modéliser une fonction en vue de faire de la classification ou de la régression. En exploitation, le problème est résolu en une seule étape grâce à l'application d'un seul classifieur qui contient en lui-même la décision de classification. En apprentissage, l'activation du classifieur est indépendante des états précédents du système et la récompense d'un classifieur est perçue à la fin de son application sur les exemples à apprendre. Les problèmes de classifications sont typiquement *single-step* lorsque ces exemples sont indépendants entre eux.





Vue locale

Encodage des éléments de l'environnement					
Nom	Nourriture		Obstacle		Vide
Élément	F	G	O	Q	.
Codage	110	111	010	011	000

Classifieur typique :

F	.	(...)	.	Q	O	: up-left
110	000	(...)	000	011	010	: 01

Classifieur générique :

F/G						: up-left
11#	###	(...)	###	###	###	: 01

FIG. 2.1 – Figure de gauche : représentation de l'environnement « Wood2 » correspondant à un labyrinthe. Figure de droite : en haut, le tableau d'encodage des symboles, au milieu, un classifieur typique et en bas, un classifieur générique résolvant le labyrinthe.

Dans un apprentissage de type *multi-step*, un message provenant de l'environnement (vision locale du système plongé dans l'état courant) passe d'un classifieur à l'autre en activant le classifieur le plus approprié à chaque fois. Le tout forme une *chaîne* d'activations. Dans des environnements complexes, cette chaîne peut être dynamique et dépendre de l'évolution de l'environnement lui-même au cours du temps. En apprentissage, la récompense peut être perçue à n'importe quelle étape et les messages d'activation dépendent des étapes précédentes. Ce type d'environnement s'applique, par exemple, aux problèmes de robots se déplaçant dans des labyrinthes. Dans le problème présenté sur la figure 2.1, un robot (symbole "*") est plongé dans un labyrinthe pour lequel il n'a qu'une vue locale. Son but est d'aller chercher de la nourriture (symboles "F" ou "G") en évitant les obstacles (symboles "O" ou "Q"). Les symboles "." représentent les cellules vides. Le robot possède un certain nombre de classifieurs qui indiquent la direction du prochain mouvement en fonction du contenu de son environnement local. Ces classifieurs sont encodés en parcourant les cellules dans le sens des aiguilles d'une montre, à partir de la cellule supérieure gauche, selon le tableau d'encodage présenté sur la figure 2.1. Un classifieur typique est présenté sous le tableau. Un nouveau symbole est introduit (#), représentant un « 0 » ou un « 1 ». Le symbole "#" est un caractère *joker* (ang., *wildcard*) représentant et pouvant activer tout autre autre symbole de $\mathcal{A}$. On l'appelle aussi le « *don't care symbol* » [Wilson, 1994]. Le but du système est de proposer les classifieurs les plus génériques possibles, par exemple en encodant la nourriture par « 11# » plutôt que par « 110 » ou « 111 ». Un tel classifieur est présenté sur la figure.

La modélisation du processus de classification des données présentes sur les images hyperspectrales fait plutôt appel à l'apprentissage de type *single-step*, bien qu'il soit possible de considérer les données hyperspectrales comme un environnement de type *multi-step*. Dans ce cas, par exemple, l'objectif serait de découvrir un enchaînement de classifieurs adéquats pour traiter les données temporelles. Comme de tels systèmes sont actuellement algorithmiquement hors de portée, nous nous concentrons dans nos travaux sur les modèles *single-step*.

Le but pour un classifieur, tout comme dans la plupart des modèles d'apprentissage par renforcement, est d'associer à chaque situation dans laquelle il est plongé la meilleure action possible. Chaque association possible entre un message de l'environnement (ou *stimulus*) et l'action correspondante entreprise par le classifieur (son comportement) peut être modélisée par un *état*. Le nombre d'états pour un problème complexe peut être relativement grand : si chaque état est décrit par n attributs discrets, il y a 2^n états possibles lorsque les attributs sont binaires. Avec une représentation classique du type *règle de décision*, pour pouvoir résoudre un problème comprenant un grand nombre d'états, il faudrait disposer d'un ensemble conséquent de règles, chacune associant un état avec une action donnée. La complexité algorithmique d'une telle situation exploserait vite. L'intérêt des classifieurs est qu'ils permettent d'associer directement l'action désirée à la valeur des attributs plutôt qu'à l'état dans lequel se trouve le système.

Avant les systèmes de classifieurs (LCS) de [Holland, 1971] et l'utilisation des algorithmes génétiques, les systèmes de classifieurs pouvaient être vus comme un système de production au sens de [Baum et Durdanovic, 2000] puisqu'ils définissent une architecture reposant sur une base de règles. Le schéma très simple d'un environnement qui produit un message activant les conditions d'une règle donnée, puis l'émission de l'action correspondante dans ce même environnement évoque fortement le principe général d'un moteur d'inférences. [Post, 1943] a montré que de tels systèmes, nommés *Post Production System*, étaient calculables en un temps polynomial. Ces règles, sous la forme *If ... Then ... Else* dans des langages de programmation de haut niveaux et utilisés par les experts dans les systèmes à base de connaissances [Marchand et Damper, 2000], présentent l'avantage d'être un modèle formel de programme universel.

2.3 Description des classifieurs

Selon Holland, un classifieur est une règle représentant une connaissance du système au niveau le plus bas. Il possède une ou plusieurs conditions (formant la partie <*condition*>) et une seule action (formant la partie <*action*>). La partie <*condition*> permet de déterminer par évaluation (ou *matching*) l'ensemble des messages qui activent la règle, et la partie <*action*> contient le message que le système va émettre lorsque la partie <*condition*> est satisfaite.

Plus formellement, on a la forme suivante :

$$<condition> : <action> \equiv c_1, c_2, \ldots, c_i, \ldots, c_n : a \tag{2.1}$$

où c_i ($i \leq n$ et $n \geq 1$) est l'une des valeurs binaires (en représentation binaire) formant la partie <*condition*> du classifieur et a la partie <*action*>.

Chacun des membres c_i de la partie <*condition*> est une chaîne de longueur k fixée définie sur un alphabet $\mathcal{A}$ donné. Dans la plupart des systèmes de classifieurs appliqués, on a $\mathcal{A} = \{0, 1, \#\}$.

La partie <*condition*> permet de coder des informations élémentaires, comme une couleur, un spectre, une forme ou tout autre constante. Elle est utilisée lors de l'activation du système par les messages d'entrée, et correspond aux conditions à remplir pour rendre le *classifieur actif*. Les valeurs c_i sont entendues comme des conjonctions de la partie <*condition*>. Il n'y a pas de disjonctions dans le modèle traditionnel des classifieurs.

La partie <*action*> est le message de sortie correspondant, codant l'action à effectuer lorsque le classifieur est activé. Il s'agit généralement d'une action basculant le système dans l'état suivant (en modifiant ou non l'environnement externe), par exemple, un changement de cap pour un robot ou la capture d'une nourriture. En télédétection, il s'agit le plus souvent de l'attribution d'une classe à un pixel.

Le but d'un système de classifieurs est de modéliser entièrement la fonction (ang., *mapping*) X x A, avec X représentant les messages d'entrée et A les actions correspondantes qui maximisent la récompense reçue de l'environnement.

On appelle *spécificité* d'un classifieur, le nombre de valeurs instanciées (différentes du symbole "#") d'une condition. S'il n'y a pas de *joker*, la *spécificité* est maximale et vaut la longueur de la condition et si tous les caractères sont des *jokers*, la *spécificité* est nulle. Lorsque, pour deux classifieurs, l'un a une *spécificité* plus faible que l'autre, et que les valeurs instanciées sont toutes identiques, on dit que le premier

englobe (ang., *subsume*) le deuxième. Dans l'exemple suivant, la base de règles BR1 est parfaitement équivalente, au sens de la spécificité, aux deux bases BR2 et BR3, mais pas à la base BR4.

BR1	001010 : 11100 101010 : 11100 001110 : 11100
BR2	#01010, 001110 : 11100
BR3	001#10, 101010 : 11100
BR4	#01#10 : 11100

La *spécificité* est importante au moment des manipulations des classifieurs par l'algorithme génétique, pour contrôler le niveau de généralisation que l'on rajoute à chaque fois : ceci peut notamment être contrôlé au niveau de l'opérateur de croisement génétique.

Le symbole "#" est présenté par [Schaeffer et Schuurmans, 1989] comme pouvant améliorer la performance des règles. Il permet l'économie de classifieurs dans le système et l'augmentation du nombre de solutions par classifieurs. De plus, la performance du système d'apprentissage est améliorée car les *jokers* autorisent les notions de *principes généraux* et d'*exceptions* [Goldberg, 1994]. Enfin, ils introduisent une notion de hiérarchisation, dont [Riolo, 1988] a montré l'influence lors de l'apprentissage (notion de *niches*).

À moins de connaître l'espace de solutions, les règles initiales pour l'apprentissage (dont l'ensemble est nommé *base de connaissance initiale* du système) ne sont introduites ni par le programmeur, ni par un expert du domaine. L'algorithme de sélection de ces règles joue donc un rôle capital. Pendant l'apprentissage la compétitivité des *hypothèses* introduites par la partie <*action*> des règles est totalement déterminée par leurs performances passées et leurs *spécificités* [Richards, 2001].

D'autres représentations ont été introduites ou testées depuis. En plus de la représentation classique (binaire, de type <*condition*>/<*action*>), on peut citer les représentations continues (chaque condition est une séquence de valeurs réelles ou d'intervalles de réels), les expressions LISP [Lanzi, 1999b], celles dont les classifieurs ne sont activés qu'en fonction de la valeur d'un registre (de la forme <*condition*><*registre*> : <*action*><*registre*>), et enfin avec messages auto-réinjectés, correspondant au cadre formalisé par les travaux initiaux de Holland.

2.4 Approches pour la base de connaissances

La base de connaissances, manipulée par le système sous la forme d'une *population de classifieurs*, n'est pas une simple liste de classifieurs. Principalement deux approches ont été étudiées.

- L'approche de Pittsburgh [DeJong, 1988] considère que la population est un ensemble de bases de règles (ang., *rule-sets*). Chaque individu est donc un système de production à part et la population évolue en hybridant et en sélectionnant la meilleure base de règles.
- L'approche de Michigan [Holland, 1986] est l'approche historique. Elle considère, quant à elle, que chaque règle est un individu isolé. Les règles sont évaluées de manière individuelle par l'algorithme génétique et la population évolue en croisant ses règles entre elles. Cette approche est non seulement la plus classique, la plus documentée dans la littérature, mais aussi celle qui donne en général les meilleurs résultats [Holland, 1986; Booker et al., 1989].

L'approche retenue doit être choisie précautionneusement en fonction du problème à résoudre. Il existe cependant des difficultés dans les deux cas [Carse et Pipe, 2001] : dans les systèmes de type Michigan, l'équilibrage entre coopération et compétition des individus est délicat, mais la représentation de la solution par des individus indépendants est efficace pour beaucoup de problèmes. Dans les systèmes de type Pittsburgh, les individus sont des entités plus complexes et sont plus difficiles à manipuler par les opérateurs génétiques.

Une troisième approche, moins courante, est une approche d'apprentissage de règles itératives [Venturini, 1993]. Les chromosomes codent des règles individuelles et à chaque cycle de l'algorithme génétique une nouvelle règle est adaptée puis ajoutée à la base de règles.

2.5 Composants d'un système de classifieurs

Au sein d'un système de classifieurs (LCS), un AG ne représente qu'une simple composante : le système utilise le paradigme évolutif pour créer des individus et adapter leur comportement à un environnement qui peut être modifié au cours de l'apprentissage. La figure 2.2 présente une illustration schématique des différents composants du système de classifieurs étudié par [Wilson, 1994]. Son fonctionnement est détaillé dans la section suivante. Il se compose des parties suivantes :

- *L'environnement.* Souvent présenté comme une région de l'espace de recherche dans laquelle évolue les classifieurs, il formalise le problème à résoudre. Il évolue et se modifie de manière autonome mais peut interagir avec le LCS à l'aide d'un système de messages.
- *Une interface d'entrée composée de détecteurs.* Ils permettent de traduire en messages d'entrée la situation courante et les événements que l'environnement a créé dans l'état courant.
- *Une interface de sortie composée d'effecteurs.* Ils permettent au LCS d'interagir avec l'environnement en produisant certaines actions, pilotées par les messages de sortie du système.
- *Plusieurs listes de messages* dont l'intérêt et l'utilisation dépendent des particularités qu'apportent leurs auteurs aux LCS considérés. On peut toutefois considérer que la présence des trois composants suivant est invariante :
  - une base de classifieurs $[P]_t$ (ang., *Population Set*). Appelée *pool de classifieurs*, elle contient la population de classifieurs au départ de l'algorithme et représente les connaissances du système à l'instant t. Plus formellement, si l'environnement est markovien, la transition de $[P]_t$ à $[P]_{t+1}$ est homogène[2],
  - une liste de messages $[M]_t$ (ang., *Match Set*), contenant tous les messages *actifs*, c'est-à-dire tous les messages qui coïncident avec le message d'entrée courant du système,
  - une liste d'actions $[A]_t$ (ang., *Action Set*), contenant les parties $<action>$ (voir la section 2.3) des messages actifs et qui seront utilisées par le LCS pour déterminer le message à envoyer aux effecteurs.

2.5.1 Système de répartition de crédits

La sélection et la manipulation des classifieurs dans les différentes listes fait appel à deux algorithmes :

- un algorithme de répartition de crédits distribuant des récompenses aux classifieurs efficaces et pénalisant les autres,
- un algorithme génétique permettant l'évolution de la base de classifieurs.

Le système de répartition de crédit (ang., *bucket brigade algorithm*) a été décrit par Holland dans [Holland, 1985] et utilise le principe de la vente aux enchères : les classifieurs efficaces sont récompensés tandis que les autres sont éliminés. Dans cet algorithme, il y a deux éléments principaux : la vente aux enchères et la chambre de compensations. Lorsqu'un message provenant des détecteurs déclenche des classifieurs, ces derniers s'inscrivent pour une vente aux enchères. Chaque classifieur y propose une enchère, généralement proportionnelle à sa *force* (ang., *strength*). La *force* S d'un classifieur est essentiellement calculée à partir de la fonction d'évaluation. Celui qui emporte la vente va produire un message correspondant à l'action à effectuer, action qui va être émise par le système. L'action et donc le message peut être bénéfique ou non à l'environnement : selon le cas on augmentera ou diminuera la *force* du classifieur correspondant. Puis, il passera par la chambre de compensation, en payant de manière équitable les classifieurs actifs qui n'ont pas gagné (ceux qui concordent avec le message des détecteurs, mais qui n'ont pas été sélectionnés) et en perdant une partie de sa *force*. Cette récompense qui remonte le long de la chaîne des classifieurs perdants a donné le nom à l'algorithme (dit des *porteurs d'eau*).

2.5.2 Interactions entre composants

L'ensemble des classifieurs qui sont mis en compétition pour contrôler le système et optimiser la réponse de l'environnement sont rassemblés dans une liste ($[M]_t$ dans la figure 2.2). Cette réponse est

[2]Le pool au temps $t+1$ est entièrement déterminé de manière probabiliste par le pool au temps t.

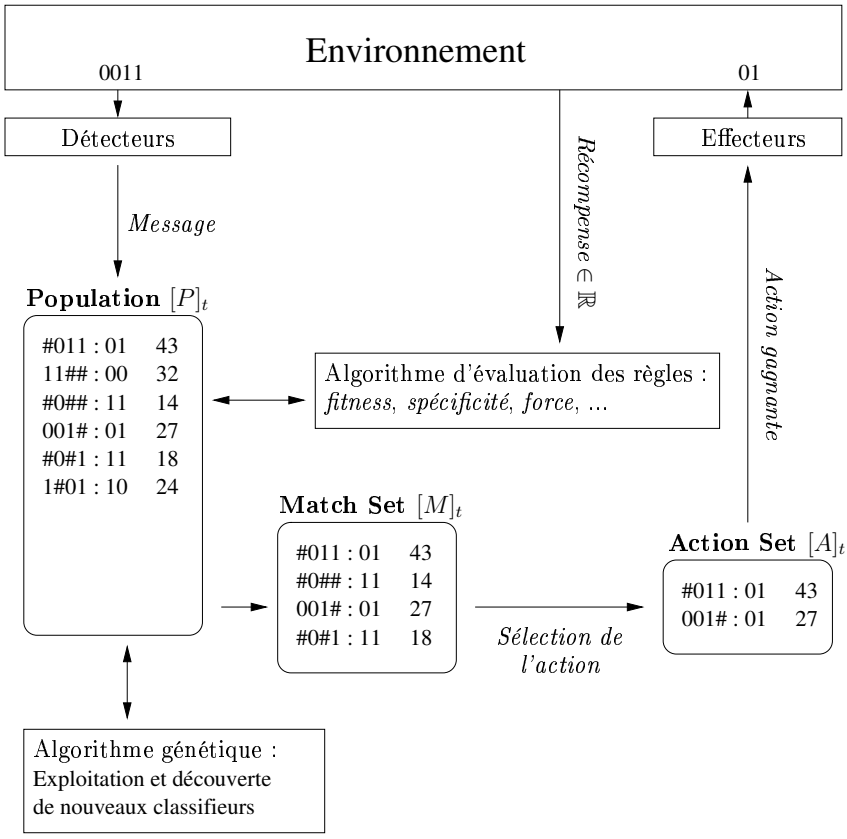


FIG. 2.2 – Modèle simplifié d'un système de classifieurs [Wilson, 1994].

donnée sous la forme d'une récompense (ang., *reward*) ou d'une pénalisation basée sur la performance ou l'échec de l'action courante du système vis à vis de l'environnement. Du point de vue du système, cette récompense se cumule au crédit (ou *force*) des classifieurs ayant généré l'action et détermine, au fil de l'apprentissage, l'influence de chaque classifieur dans la compétition. *A priori*, les individus sont créés à l'initialisation de l'algorithme et seule l'évolution de la *force* de chacun d'eux permet de les distinguer par la suite. L'apprentissage ne pourrait donc renvoyer de résultats satisfaisants sans qu'un processus évolutif renouvelle la population. Ce rôle est tenu par un algorithme génétique, couplé à la base [P], qui permettra de créer de nouveaux classifieurs par croisement ou mutation à partir des meilleurs, et d'oublier les plus mauvais.

Pour pouvoir évaluer la connaissance assimilée par le système et guider la génération des nouveaux classifieurs, le LCS utilise une fonction d'évaluation qui est déterminée par le problème à résoudre. Dans la plupart des cas, elle correspond directement à la réponse donnée par l'environnement. La performance des classifieurs est jugée en regard de leur degré de contribution à la résolution du problème.

[Richards, 2001] distingue deux modes dans lesquels peuvent se trouver un LCS à un instant t donné : le *mode apprentissage* (ang., *Learning Mode*) et le *mode application* (ang., *Application Mode*). Lorsque le système n'est pas en train d'apprendre, il reçoit de l'information de l'environnement via ses détecteurs, détermine le classifieur et l'action adaptée puis exécute cette action dans l'environnement. Ce processus est nommé *Application Mode*. L'algorithme correspondant est présenté dans la figure 2.3.

La base de classifieurs peut être vue comme une population d'hypothèses à tester. Une hypothèse, ici un classifieur, est sélectionnée lorsqu'elle est appropriée à la situation courante (matérialisée par les

1. Perception par les détecteurs d'un message de l'environnement
2. Conversion dans l'alphabet du système, assemblage en *messages environnementaux*
3. Comparaison avec les parties <*condition*> du pool $[P]_t$
4. Transfert des classifieurs *actifs* du pool $[P]_t$ vers le pool $[M]_t$
5. Création du pool $[A]_t$ à partir d'une stratégie de sélection (roulette, ...)
6. Création d'un message M_a à partir de l'action gagnante
7. Emission du message M_a sur les effecteurs qui modifient l'environnement
8. Envoi d'un message au système de répartition des crédits afin de payer les gains ou prélever les charges des classifieurs

FIG. 2.3 – *Application Mode* : interactions avec l'environnement.

messages perçus par le système). Cette pertinence ou *compétitivité du classifieur* est déterminée par la valeur de la *force* du classifieur dans les cycles antérieurs de l'algorithme et par sa *spécificité* par rapport au message courant. Au cours du temps, l'algorithme génétique provoque l'échange d'information acquise par les classifieurs. Ils sont sélectionnés à l'aide d'une fonction d'évaluation et d'un opérateur spécial, appelé le *Triggered Genetic Algorithm Operator* (TGAO). Son rôle est de lancer les opérateurs de croisement et de mutation génétique sur les classifieurs à intervalles réguliers.

La figure 2.4 présente le fonctionnement détaillé des interactions entre les composants principaux et la circulation des messages dans le système. Les messages sont émis par l'environnement (en haut de la figure) vers le système, qui les traite et les stocke dans une base de classifieurs ($[P]_t$). L'opérateur TGAO, que nous venons de décrire, s'active alors sur cette base : il représente la source de création des nouveaux classifieurs et il est basé sur un algorithme génétique (voir la figure 2.5 qui présente le *mode apprentissage*). Le système doit alors choisir une action à émettre dans l'environnement. Pour cela, un mécanisme de répartition des crédits permet d'élire l'action gagnante, qui *paie* une enchère aux classifieurs non activés de la façon décrite dans la section précédente. Dans le cas où aucun classifieur ne pourrait répondre au stimulus envoyé par l'environnement, ou lorsque la base de classifieurs ne répond pas à certains critères déterminés par un opérateur spécial nommé TCDO (ang., *Triggered Cover Detector Operator*), un nouveau classifieur est créé de manière aléatoire. Une action est dans tous les cas émise vers l'environnement qui gratifie ou pénalise les classifieurs activés.

L'évolution naturelle du système de classifieurs conduit à remplacer les classifieurs par défaut les plus généraux par des classifieurs plus spécifiques correspondant aux situations particulières. Le développement de règles générales et spécifiques, permet au système d'apprendre avec habilité et flexibilité, traitant les cas inédits grâce aux règles les plus générales et les exceptions par les règles les plus spécifiques. Ce principe de classifieurs hiérarchisés est connu dans la littérature sous le terme de *default hierarchies* [Holland et al., 1986]. Lorsque le nombre de réponses (ang., *feedback*) positives provenant de l'environnement augmente, on considère que le nombre d'hypothèses validées augmente également et que les classifieurs, simples hypothèses au départ, sont devenus des lois éprouvées.

2.6 Le *Covering Operator*

Si la liste [M] est vide, c'est-à-dire lorsqu'aucun classifieur de [P] n'est activé par le message $m = \{mess_{env}\ 0, \ldots, mess_{env}\ n\}$ des détecteurs, le reste de l'algorithme ne peut s'appliquer, car cela empêche le choix d'une action et son émission dans l'environnement. Dans ce cas, l'environnement risque de se stabiliser : le LCS serait immobilisé dans un minimum local et, au mieux, la solution finale ne serait pas satisfaisante. [Wilson, 1994] propose plusieurs stratégies de détection de telles situations et propose

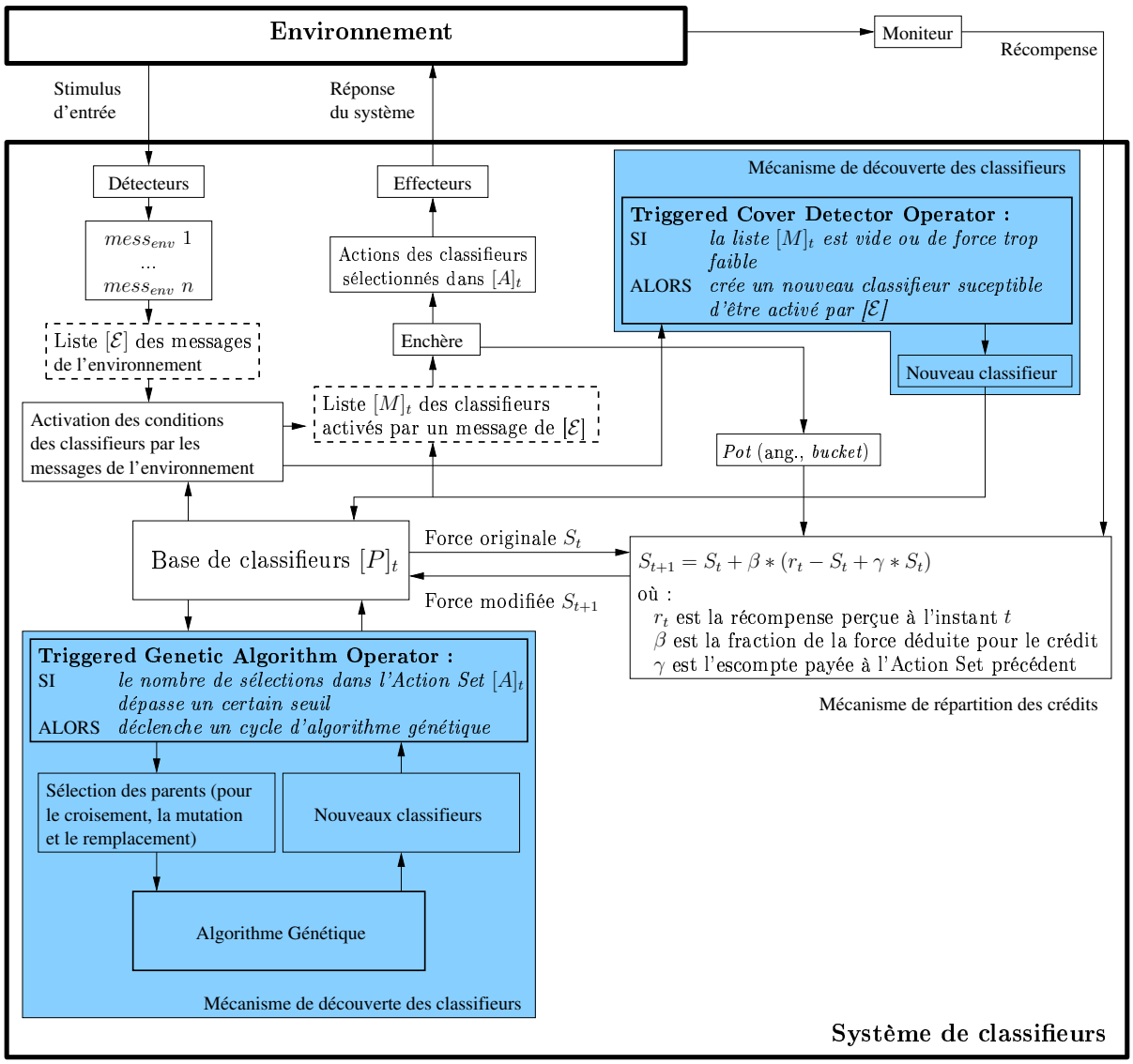


FIG. 2.4 – Interactions détaillées des messages au sein d'un LCS. Les mécanismes qui sont dédiés à la découverte des classifieurs sont placés dans des cadres sombres.

des méthodes de recouvrement d'une liste [M] valide, qu'il formalise dans un opérateur nommé *Covering Operator*. Celui-ci intervient lorsque la liste est vide ou lorsque la *force totale* $S_{[M]_t}$ de $[M]_t$ (somme des *forces* S de tous les classifieurs de [M] à l'instant t) est inférieure à une fraction ϕ de la *force* moyenne des classifieurs du pool [P]. Le rôle du *Covering Operator* est de créer un nouveau classifieur C de la façon suivante :

- la partie <*condition*> est construite à partir du message m en y ajoutant un taux $\theta_{\#}$ fixé de caractères *jokers* (#),
- la partie <*action*> est choisie de manière aléatoire,
- la *force* de C est équivalente à la *force* moyenne du pool [P].

C est ensuite introduit dans [P] après avoir supprimé un classifieur en utilisant une procédure de sélection eugénique lancée par l'algorithme génétique. Un nouveau pool $[M]_{t+1}$ est créé et l'algorithme se poursuit de façon habituelle. À titre d'illustration, $\phi = 0.5$ et $\theta_{\#} = 1/3$ semblent être d'après Wilson

1. Initialisation du système de classifieurs (t=0)
2. Génération de la population initiale ($[P]_0$)
3. Exécution d'un cycle de la boucle principale de l'algorithme (voir la figure 2.3)
4. Évaluation de la population $[P]_t$ (*fitness, TGAO*)
5. Sélection des classifieurs pour la population $[P]_{t+1}$
6. Croisement et ajout des enfants dans $[P]_{t+1}$
7. Mutation de la population $[P]_{t+1}$
8. Remplacement des plus mauvais classifieurs de la population $[P]_t$ par les meilleurs de la population $[P]_{t+1}$ puis t = t+1
9. Si Terminaison(t) = *faux*, alors revenir en (3)

FIG. 2.5 – *Learning Mode* : algorithme général.

de bons paramètres pour résoudre un problème de labyrinthe simple. Cet opérateur est un peu brutal puisqu'il risque d'orienter le LCS vers un apprentissage par cœur et n'exploite pas les connaissances déjà apprises comme le fait l'algorithme génétique. Néanmoins, l'apparition de messages provoquant la génération de pools de taille nulle est inévitable dans la plupart des problèmes, et le *Covering Operator* est vu comme un dernier recours. De plus, il permet de tester de nouvelles hypothèses (principe de l'exploration) basées sur des messages authentiques, plutôt que d'agir de manière totalement aléatoire.

2.7 Le classifieur de Wilson (XCS)

Généralement, le paramètre de *force* est utilisé à la fois comme une prédiction de la future récompense perçue et comme paramètre principal de la fonction d'évaluation. Wilson a remarqué que dans certains cas, cette valeur de prédiction n'est pas adéquate lorsqu'elle est utilisée dans le calcul de la fonction d'évaluation. Par exemple, l'environnement peut se trouver dans deux états distincts activant chacun le même classifieur et l'action appropriée dans chacun des cas peut être différente de celle prévue par le classifieur. Ce classifieur va alors acquérir une *force* correspondant à la moyenne entre une récompense maximale et une pénalisation maximale. Ce phénomène vient du fait que le classifieur n'est pas assez spécifique pour distinguer les deux cas, mais il survit cependant car la fonction d'évaluation est basée sur la *prédiction de la récompense* et non sur la *précision de cette prédiction.*

Pour résoudre ce problème, l'algorithme XCS [Wilson, 1995] étend la famille des systèmes de classifieurs en associant plusieurs paramètres supplémentaires à chaque règle.

Le premier paramètre, p, est la prédiction de la récompense que XCS recevra s'il applique cette règle. Le second paramètre, e, est l'erreur associée à cette prédiction, et le dernier paramètre F représente la *fitness*, une mesure de la précision de la prédiction utilisée dans l'algorithme génétique. Ce dernier est implanté non plus sur la population de règles [P] (comme présenté dans les figures 2.2 et 2.4), mais sur le *Match Set* [M], ce qui permet de découper le problème en niches (ici un ensemble d'états de l'environnement activé par le même ensemble de classifieurs) partageant les récompenses ce qui évite qu'une niche dominante ne prenne le contrôle des décisions émises par le système, par rapport à une autre niche. Une autre extension de Wilson concerne les macro-classifieurs. Cette notion ne fait pas allusion à une représentation différente des classifieurs : il s'agit d'une technique algorithmique pour réduire la complexité de calcul de l'activation des classifieurs par le ou les messages entrants. Dans des problèmes simples (multiplexeur, labyrinthe), les classifieurs ayant les mêmes conditions et actions sont légions. Wilson propose de ne garder qu'une copie par classifieur identique et d'incrémenter un autre paramètre du classifieur, nommé *numerosity*, en fonction du nombre d'occurrences réelles du classifieur

concerné. La différence de qualité est nulle par rapport à un LCS dépourvu de cette amélioration et cela n'affecte pas les opérations génétique sur les classifieurs sous-jacents. On appelle *population réelle* une base de classifieurs uniques et *population virtuelle* l'ensemble des classifieurs dupliqués en tenant compte du paramètre de *numerosity*.

Signification	**Valeur idéale**
Nombre de classifieurs dans la population	50
Taux d'apprentissage pour l'actualisation de la *fitness*, de l'erreur, de la prédiction et de l'estimation de la taille de l'*Action Set*	0.2
Fraction de la *fitness* moyenne en dessous de laquelle un classifieur peut être supprimé	0.1
Triggered Genetic Algorithm Operator (TGAO) : Nombre moyen de sélections dans l'*Action Set* que doivent avoir les classifieurs avant de déclencher l'AG	5
Covering Detector Operator (CDO) : Fraction de la *force* moyenne du *Match Set* pour déclencher le *Covering Operator*	0.1
Seuil d'erreur en dessous duquel la précision du classifieur est fixée à 1	10
Durée de vie minimale d'un classifieur	20
Probabilité de croisement	0.8
Probabilité de mutation	0.15
Réduction de l'erreur lorsqu'un nouveau classifieur est créé par l'AG	0.25
Reduction de la *fitness* lorsqu'un nouveau classifieur est créé par l'AG	0.1
Prédiction initiale d'un classifieur	10.0
Erreur initiale d'un classifieur	0.0
Fitness initiale d'un classifieur	0.01

TAB. 2.1 – Significations et valeurs des principaux paramètres utilisés dans un classifieur de type XCS [Wilson, 1994].

Le tableau 2.1 présente quelques paramètres intéressants associés à XCS et leurs valeurs idéales, selon Wilson.

XCS a été testé dans des environnements *single-step* et *multi-step*. Dans les problèmes de type *multiplexeur* (*single-step*), Wilson a montré que la complexité du problème en nombre de générations n'est pas proportionnelle à la taille de l'espace de recherche mais au nombre de classifieurs génériques (nombre de caractères *jokers* non nul) nécessaires pour résoudre la tâche. Ainsi, l'obtention d'une performance maximale pour un problème inscrit dans un espace de recherche deux cents fois plus grand a été pour XCS seulement trois fois plus difficile en terme de nombre de générations et de taille de la population finale [Wilson, 1995].

Dans [Wilson, 1998], Wilson introduit deux nouvelles optimisations qui permettent d'augmenter sensiblement la qualité de la généralisation et les performances du système :

1. L'algorithme génétique croise et mute les classifieurs stockés dans la liste *Action Set* et non plus ceux du *Match Set*. Le *Match Set* contient des classifieurs jeunes, dont ceux créés par le *Covering Operator*, et qui risquent fortement de se faire éliminer à la génération suivante par l'algorithme génétique. D'autre part, les conditions de ces classifieurs se ressemblent, ils y sont justement parce qu'ils ont été activés par le même message d'entrée. L'opérateur de croisement est donc inefficace, seul l'opérateur de mutation pourrait être utile. Des études ont montré que si le *Match Set* contient

des classifieurs performants et précis pour toutes les actions possibles, les conditions de ces classifieurs sont différentes, ce qui produit fréquemment des *enfants* inadaptés après l'application de l'opérateur de croisement. En revanche, appliquer l'algorithme génétique dans l'*Action Set*, dans lequel les parties <*action*> sont toutes identiques, produit de meilleurs résultats.

2. La seconde amélioration concerne les opérateurs génétiques eux-mêmes. Après leur application, si un classifieur fils possède une condition englobée (ang., *subsumed*) par un classifieur déjà existant et donc possédant une *spécificité* plus grande, ce fils est remplacé par une copie du classifieur englobant. Par exemple, supposons que la population contienne deux classifieurs « C1 :11## » et « C2 :#### » et que ces deux classifieurs ont la même performance (l'erreur de prédiction est la même). C1, en étant plus spécifique que C2, va s'activer moins souvent que C2. Puisqu'il sera sélectionné moins souvent, les occasions de réception d'une récompense lui seront plus rares et il sera finalement éliminé par l'algorithme génétique (ce qui est souhaité). L'amélioration proposée par Wilson raccourcit simplement le *temps d'attente* avant sa suppression. À cause de son effet sur la base de classifieurs, cette optimisation peut être considérée comme une *mutation dirigée* dans le sens où l'algorithme génétique est contraint à ne produire que des *enfants* plus généraux (avec une *spécificité* plus faible) que les parents.

2.8 Classifieurs et logique floue

Parmi les possibilités offertes par les systèmes de classifieurs, une piste consistant à mêler classifieurs et logique floue a été explorée. Appelés LFCS (ang., *Learning Fuzzy Classifier System*, [Rendon, 1997]), ces systèmes sont capables de traiter des classifieurs requérant des variables continues et dont une interprétation linguistique pourrait être proposée. Cette classe d'algorithmes est connue dans la littérature sous le nom générique de GFS (ang., *Genetic Fuzzy System*), parmi lesquels les systèmes les plus sophistiqués sont sans doute les GFRBSs (ang., *Genetic Fuzzy Rule-Based Systems*) [Cordon et al., 2001; Valenzuela-Rendón, 1991; Parodi et Bonelli, 1993]. L'intégration de la logique floue permet d'utiliser des facteurs de certitude en *langage naturel* comme chaud, froid, sûr, certain, peu sûr, etc. Concernant la représentation de la connaissance exprimée par les classifieurs, elle peut être considérée comme étant divisée en :

- une base de règles (BR) qui regroupe l'ensemble des règles floues. L'utilisation de l'algorithmique génétique pour apprendre de nouvelles règles nécessite d'avoir préalablement intégré dans la base de données des fonctions d'appartenance floue. La littérature a considéré les trois types d'approches possibles (voir la section 2.4) : l'approche de Michigan [Ishibuchi et al., 1999], l'approche de Pittsburgh [Hoffmann et Pfister, 1997] et l'approche itérative [Gonzalez et Perez, 1999]. Dans l'approche de Pittsburgh, la représentation la plus couramment utilisée pour les groupes de classifieurs sont les matrices relationnelles et les tables de décision. Dans le cas de l'approche de Michigan, les classifieurs sont encodés sous la forme de simples listes de règles. Pour coder les règles individuelles, on emploie des chaînes binaires de longueur fixe [Gonzalez et Perez, 1999] ou on utilise le *messy-coding* (voir la section 2.9).
- une base de données (BD) qui contient la définition des facteurs d'échelle et des fonctions d'appartenance de l'ensemble flou associé avec les termes linguistiques utilisés. Contrairement à la BR, l'apprentissage automatique d'une BD par algorithme génétique est plus délicat [Velasco, 1998]. En effet, la représentation des composants d'une BD et donc de l'espace de recherche est hétérogène. On préférera donc plutôt utiliser des techniques d'optimisation (ang., *tuning*), consistant plus à optimiser une BD existante qu'à en apprendre une nouvelle, c'est-à-dire à trouver les paramètres optimaux pour les fonctions d'appartenance ou les facteurs d'échelle. Un *tuning* génétique est possible. Les facteurs d'échelle sont paramétrés par un simple scalaire ou deux limites inférieure et supérieure. Les fonctions d'appartenance, habituellement triangulaires [Carse et Pipe, 2001], trapézoïdales ou gaussiennes sont encodées dans le chromosome en utilisant de 1 à 4 paramètres réels [Cordon et al., 2001].

L'algorithme général de fonctionnement d'un LFCS diffère peu de celui des LCS classiques. On y ajoute simplement une fonction d'activation floue des messages entrant avec les classifieurs de la base, ainsi qu'une conversion inverse (ang., *defuzzification*) au niveau des effecteurs pour les messages de sortie [Valenzuela-Rendón, 1998] :

1. Les détecteurs perçoivent les messages d'entrée de l'environnement. Ils sont encodés en messages flous (les valeurs continues sont converties en valeurs floues prises dans un ensemble borné de valeurs possibles) et ces derniers sont ajoutés à la liste de messages [$\mathcal{E}$].
2. La base des classifieurs [P] est parcourue afin de trouver tous les classifieurs dont les conditions sont satisfaites, complètement ou partiellement, par les messages de la liste [$\mathcal{E}$]. Ces messages sont placés dans la liste [M].
3. La liste [$\mathcal{E}$] est vidée.
4. Les actions des classifieurs de [M] sont placés dans la liste [A].
5. Les effecteurs convertissent par *defuzzification* les messages flous de la liste [A] en valeurs de sortie acceptables par l'environnement et y propagent ces valeurs.
6. La récompense de l'environnement est transmise aux classifieurs choisis dans la liste [A].
7. Un nouveau cycle débute.

Ce type de système permet d'utiliser des fonctions d'appartenance floue, autorisant l'approximation des variables d'entrée du système, plutôt que d'obliger une activation exacte comme dans le cas des LCS. Malgré la puissance apparente de ces classifieurs, il n'est pas évident que les systèmes de type LFCS soient capables de construire des chaînes de règles efficaces d'une certaine taille, pour résoudre des problèmes *multi-step*, par exemple. Il semblerait, selon [Furuhashi et al., 1993], que lorsque le système cherche de nouveaux classifieurs, l'imprécision transférée d'un classifieur à un autre par les variables floues explose et que le système ne puisse plus rien déduire.

2.9 Les S-classifieurs

Les représentations binaires des classifieurs présentent deux principales limitations : tout d'abord, l'encodage binaire des messages des détecteurs peut induire une perte d'information à propos de la structure réelle de l'environnement. Ensuite, la correspondance entre la position des bits dans la partie <*condition*> d'un classifieur et celle des bits produits par les détecteurs est fixe, ce qui interdit les messages de taille variable. Si le premier point risque de provoquer des limitations uniquement avec les environnements dans lesquels l'aspect représentatif joue un rôle important (comme dans le problème de classification d'images), le second point est indépendant du type d'environnement et de détecteurs et se rencontre donc plus fréquemment.

Jusqu'à présent, les applications dévolues aux systèmes de classifieurs (par exemple, les labyrinthes simplistes du type de la figure 2.1, page 22) se suffisaient amplement d'une représentation binaire, car la capacité d'apprentissage du système était considérée comme plus importante que sa capacité à généraliser. Par exemple, le système XCS de [Wilson, 1995] aborde une représentation plus évoluée. Lanzi s'est rendu compte que le problème de la représentation allait devenir une question d'intérêt croissant. Il proposa dans deux articles distincts ([Lanzi, 1999a] et [Lanzi, 1999b]) deux nouvelles approches pour le formalisme des classifieurs en introduisant les *S-classifieurs*.

La première, nommée *messy-coding*, a pour but d'éliminer la correspondance fixe entre position des bits dans la partie <*condition*> d'un classifieur et la position des bits des détecteurs. Cette technique permet aux conditions d'acquérir une taille variable en découpant les gènes en plusieurs blocs indépendants. Pour aligner les blocs des classifieurs avec les détecteurs, chaque bloc est associé à un *tag* qui fait référence à un détecteur. Il est possible d'avoir un classifieur qui possède deux gènes de même *tag* (classifieur sur-spécifié) ou des gènes manquants (classifieur sous-spécifié). La propriété de généralisation des symboles *joker* s'exprime donc naturellement par des classifieurs sous-spécifiés. Dans le *messy-coding*, un classifieur est représenté par un chromosome correspondant à une liste de taille variable de couples

<variable, valeur>, dont l'ordre n'a pas d'importance. Une classe d'algorithmes génétiques spéciaux, les MGA (ang., *Messy Genetic Algorithms* [Goldberg et al., 1989]) a donc été créée pour évaluer la *fitness* de tels chromosomes. Ces travaux ont donné lieu à l'implémentation de XCSm [Lanzi, 1999a], une version de XCS adaptée pour le *messy-coding* dans laquelle les opérateurs de *covering*, d'*activation* et les opérateurs génétiques ont été réécrits. XCSm a été surtout testé dans des problèmes de labyrinthe du type Wood (voir la figure 2.1, page 22). Huit capteurs possibles (N, S, E, O, NE, ...) déterminent la nature du terrain autour de l'agent et son prochain mouvement. Un exemple de *messy-gene* (gène de XCSm) situé dans la partie <*condition*> d'un classifieur est le suivant :

(N , 1#)

Cette condition est activée si l'objet situé au Nord de la position courante du système est l'un des deux objets correspondant à la description « 1# », c'est-à-dire l'objet « 10 » ou « 11 ». L'un des principaux intérêts des *messy-gene* est qu'ils permettent la réutilisation de la connaissance : la spécificité d'un allèle de la partie <*condition*> est définie en fonction du capteur convenable (ici le capteur N pour Nord) et non en fonction de la position du bit codant sa description dans le message entrant, si bien qu'il soit possible de réutiliser les gènes appris dans un problème donné avec un certain type de capteur pour résoudre d'autres problèmes avec des capteurs différents. Dans [Lanzi, 1999a], Lanzi obtient de manière expérimentale une base de classifieurs entraînés avec un système comprenant des capteurs 4-voisins (N, S, E, O) et démontre la portabilité et la correction optimale de cette base plongée dans un système comprenant des capteurs 8-voisins (N, ..., O, NE, SE, NO, SO). La correction de la base s'est faite en utilisant une phase d'entraînement supplémentaire n'ayant employé aucun autre opérateur génétique que celui de la mutation. L'algorithme sait tirer parti de la *pré-évolution* d'une population intégrée dans un système comprenant de nouveaux capteurs : le nombre de générations nécessaires à la convergence de l'algorithme se réduit considérablement.

La seconde approche proposée par Lanzi poursuit davantage l'amélioration de la représentation des classifieurs. Le système XCSL (ang., *eXtended Classifier System in LISP*, [Lanzi, 1999b]) est une extension basée sur les S-expressions du langage LISP. Bien que ces expressions puissent être écrites de manière linéaire, elles sont traitées par les opérateurs génétiques comme étant des arbres, dont la rapidité de l'évaluation est l'un de leurs avantages. Lanzi n'a considéré que des applications simples pour son système XCSL, dont les solutions s'expriment par des fonctions booléennes. Pour cela, la partie <*condition*> de ses classifieurs représente une composition des opérateurs logiques *AND*, *OR*, *NOT*.

```
<cond> :=   "(" NOT <cond> ")"
          | "(" AND <cond> <cond> ")"
          | "(" OR  <cond> <cond> ")"
          | <var>

<var> := "X0" | "X1" | "X2"
```

FIG. 2.6 – La grammaire BNF qui permet de générer toutes les conditions possibles des classifieurs de XCSL.

La figure 2.6 présente la grammaire BNF [Naur, 1960] qui permet de générer toutes les conditions possibles des classifieurs de XCSL qui s'inscrivent dans un sous-ensemble des fonctions booléennes à trois variables.

Sur la figure, les terminaux sont représentés entre guillemets et les symboles fonctionnels entre chevrons. Cette grammaire a été utilisée sur le problème d'apprentissage de fonctions booléennes[3]. Par exemple, voici l'expression d'une condition complète : « (OR S1 S2) » où S1 et S2 sont des terminaux dépendant de l'environnement : une simple variable booléenne dans le cas du multiplexeur ou un prédicat

[3]Fonction f de n variables $(x_0, \ldots, x_n)$ définie par $f : \{0,1\}^n \rightarrow \{0,1\}$.

de test des capteurs d'orientation dans le cas d'un labyrinthe. Les différences par rapport à XCS sont les mêmes que pour la première approche. Certains opérateurs ont été modifiés de manière adéquate :

- l'opérateur d'*activation* ne pose pas de difficulté particulière, l'expression est simplement évaluée en respect de la syntaxe classique du système de parenthésage de LISP,
- l'opérateur de *covering*, dont le rôle est de créer un nouveau classifieur avec une condition aléatoire lorsqu'aucun classifieur n'est activé par le message d'entrée, nécessite quelques précautions. Lanzi s'est rendu compte que pour éviter un problème de sur-apprentissage induit par une sous-spécialisation des gènes du classifieur, il valait mieux construire, pour la condition, une conjonction (OR) de trois expressions, chacune étant activée par le message d'entrée. Chacune de ces sous-expressions est une disjonction (AND) d'un nombre arbitraire de bits du message d'entrée, sauf la première qui reprend le message exact. Par exemple, si le message est « 010 », on construit le classifieur « 010 OR 01# OR ##0 » (l'opérateur AND n'est ici pas représenté). Cette représentation garantit d'avoir un classifieur spécifique au message d'entrée, tout en introduisant de la généralisation par les deux dernières sous-expressions. Malheureusement l'auteur ne précise pas dans quelle mesure le choix du nombre trois permet d'obtenir expérimentalement des résultats optimaux.
- *les opérateurs génétiques* : Dans XCSL, l'algorithme génétique est appliqué sur l'*Action Set* [Wilson, 1998] (voir la section 2.7). Deux classifieurs sont choisis avec des probabilités proportionnelles à leur *fitness*, puis sont copiés, croisés avec une probabilité χ, mutés avec une probabilité μ puis réinsérés dans l'*Action Set*. Le croisement et la mutation respectent les principes connus en programmation génétique [Koza, 1992] : le croisement inverse deux sous-arbres dans l'arbre représentant la partie condition et la mutation remplace un sous-arbre par un autre créé aléatoirement.

Concernant la représentation des classifieurs, Lanzi note un problème subtil apparaissant avec l'arrivée de l'opérateur OR, qui n'apparaît pas avec les autres opérateurs booléens, ni dans le cas des classifieurs binaires typiques de XCS. Il démontre qu'un degré de représentation plus élevé introduit un biais qui conduit l'algorithme génétique à créer des classifieurs corrompus et provoque l'instabilité du système (la performance globale se mettant à osciller[4]). La solution proposée consiste alors à modifier aléatoirement l'ordre d'évaluation des conditions liées par l'opérateur OR lors d'une étape spécifique au cours des générations génétiques.

Ce système a aussi été testé dans les environnement *multi-step* comme les labyrinthes, dans lequel le biais de la fonction OR s'est révélé être plus modéré. Cela se justifie par le fait que dans un problème de multiplexeur les conditions sur-générales sont plus courantes car le parcours de l'espace de recherche y est plus approfondi. Dans un labyrinthe, une condition n'est jamais active sur une longue période de temps car le message d'entrée change fréquemment, du fait que l'agent se déplace au sein de son environnement.

Les systèmes XCS et XCSL ont été testés dans des environnements quasi-similaires en *single-step* et en *multi-step*, ce qui permet de comparer expérimentalement leurs performances (des performances optimales ont été obtenues dans les deux cas sur tous les types de problèmes avec un nombre restreint d'itérations d'apprentissage). Cependant, de nombreux problèmes émergent avec l'apparition des classifieurs en LISP. À l'heure actuelle, ces derniers ont sans doute été encore trop peu explorés dans la littérature. Parmi ces

[4] Pour illustrer ce phénomène, prenons les deux classifieurs suivants :
(a) **if** C1 **then** action A
(b) **if** C2 **then** action A
où C1 et C2 sont des conditions formées par le symbole `<cond>` de la grammaire BNF présentée dans la figure 2.6. Ces classifieurs sont *a priori* en tout point similaires au classifieur (c) suivant :
(c) **if** C1 **or** C2 **then** l'action A
En effet, ce classifieur s'applique dans les mêmes situations que les classifieurs (a) et (b) et renvoie la même action A. Supposons que pendant une longue période de temps, le message d'entrée active plus fréquemment la condition C1 que C2 si, par exemple, C2 est légèrement plus spécifique. Si le système utilise les classifieurs (a) et (b), la performance du classifieur (b) décroîtra et il finira par être éliminé. Par contre, s'il utilise à la place le classifieur (c), la condition C2 n'influencera pas la croissance de la performance de ce classifieur et ce, même si la condition C2 est complètement inadaptée, voire fausse. Cela peut conduire l'algorithme à créer des classifieurs inefficaces, sans qu'il en soit informé. Lanzi nomme une telle condition (C2), une condition *cachée* par l'opérateur OR. Il en conclue qu'il est possible d'altérer une partie des conditions d'un classifieur sans influencer négativement sa performance (et donc sans qu'il soit possible de détecter et d'éliminer ces conditions ou classifieurs) tant qu'on a pas vérifié la totalité des éléments de la conjonction.

problèmes, il y a le calcul du degré de généralisation (ou à l'inverse, de la *spécificité*) d'un classifieur : en binaire, ce problème est trivial et revient à compter le nombre de caractères *jokers*. Avec les S-expressions, ce problème devient beaucoup plus complexe (NP-complet si l'on doit comparer deux à deux les nœuds de chacun des arbres), ce qui restreint fortement l'utilisation de certaines améliorations proposées par [Wilson, 1998], notamment la suppression des classifieurs englobés, à moins de disposer d'heuristiques particulièrement efficaces. Un autre problème est l'augmentation de la complexité des conditions qui requièrent du temps processeur pour être calculées. Là encore, des heuristiques de simplification et/ou d'évaluation d'expressions doivent être mises au point.

2.10 Principales améliorations apportées aux systèmes classiques

Même si l'on peut imaginer d'autres méthodes de gratification ou de pénalisation, [Riolo, 1988] a montré par des simulations que la capacité d'apprentissage du système de classifieurs ne dépend pas de son mécanisme exact (modification de la *force* des classifieurs pendant que le système apprend, taux d'apprentissage, ...), simplement l'apprentissage peut être plus efficace dans un cas ou dans l'autre [Richards, 2001].

[Richards, 2001], par exemple, ajoute un bruit aléatoire à l'enchère du classifieur vainqueur pour promouvoir l'exploration de l'espace des classifieurs. De plus, grâce à l'introduction de deux types de taxations supplémentaires, les classifieurs inutiles mais avec une *force* élevée sont supprimés. Une *taxe à vie* (ang., *life tax*) est définie, appliquée à chaque classifieur et à chaque itération, ainsi qu'une *taxe à l'enchère* (ang., *bid tax*) appliquée à chaque classifieur vainqueur, ce qui pénalise les classifieurs qui enchérissent à chaque fois. Ainsi, les règles qui sont inactives car occultées par des règles plus performantes ont une probabilité plus élevée d'être sélectionnées.

Il reste le cas où aucun classifieur ne satisfait le message envoyé par les détecteurs. [Robertson et Riolo, 1988] a introduit le TCDO (*Triggered Cover Detector Operator*), un opérateur qui s'applique dans certains cas, par exemple, lorsqu'aucune autre règle n'est applicable. Dans ce cas, un classifieur est créé aléatoirement de façon a être *compatible* avec le message et on lui attribue une valeur pour son paramètre de *force*. Cette *force* est assez complexe à déterminer car elle ne doit être ni trop faible, ni trop forte.

Enfin, il semble logique d'éliminer en priorité les classifieurs de *force* faible. Cependant, [DeJong, 1975] montre que le simple remplacement des plus mauvais individus est un peu expéditif et doit être amélioré. Il propose de créer une sous-population d'individus peu performants en les croisant plutôt qu'en les supprimant.

De nombreuses autres extensions ont été proposées, parmi lesquelles nous citons :

2.10.1 Les niches implicites

Pour maintenir une certaine diversité dans la population de classifieurs, [Horn et al., 1994] a exploité un système de compétition dans lequel les ressources (c'est-à-dire la quantité de récompenses à récolter) sont limitées. L'algorithme est dit *implicite* puisque l'on va forcer les règles des classifieurs à partager les ressources existantes. L'approche génétique classique consiste en un problème d'optimisation de fonction : les classifieurs gagnants sont la simple copie des meilleurs individus de leur génération. Cependant, lorsque les classifieurs s'influencent les uns les autres, on ne peut plus adopter cette solution. On parle de *co-évolution*, *co-adaptation* ou de fonctions d'optimisation dépendantes du contexte. Parce qu'il y a une compétition (tout le monde veut s'approprier les mêmes ressources), la seule manière d'y arriver est de privilégier les règles qui encouragent le partage. Tout le problème est donc de découvrir des règles individuelles et diversifiées qui permettent un partage collectif de l'environnement. [Horn et al., 1994] montre que sans cette méthode, l'algorithme génétique seul associé aux systèmes de classifieurs ne pouvait pas maintenir cet ensemble coopératif de règles.

2.10.2 La méthode COGIN

La méthode COGIN (ang., *COverage-based Genetic INduction*, [Greene et Smith, 1994]) est basée sur la notion d'*espèces* et de *niches*, déjà introduite par [Goldberg et al., 1989]. Une *espèce* est une sous-population de classifieurs qui restent stables au fur et à mesure de leur évolution. Une *niche* est une portion de l'environnement dans laquelle une espèce peut espérer survivre. Autrement dit, les individus perçoivent une gratification lorsqu'ils sont présents dans cette niche. Les individus occupant une niche consomment ses ressources (la proportion de gratification de cette niche baisse), et si une espèce souhaite survivre jusqu'à la prochaine génération, elle doit être meilleure que les autres individus de cette niche, ou bien se trouver dans une niche sans aucun autre adversaire : ce type d'apprentissage est dit *compétitif*. La méthode COGIN est utilisée lorsque l'on doit maintenir la diversité dans une population afin de prévenir l'immobilisation dans un minimum local. Elle a été utilisée dans [Fong et Yuen, 2000] pour faire de la détection d'arêtes, la diversité de la population ayant permis de traiter des images avec un taux de bruit assez significatif (10%).

2.10.3 Le modèle IMGA

Le modèle IMGA (ang., *Island Model Genetic Algorithm*, [Whitley, 1993]) est un modèle distribué. Il considère les problèmes mettant en jeu des sous-populations d'individus indépendantes entre elles, qui peuvent chacune évoluer séparément comme si elles étaient isolées les unes des autres (à l'image d'îlots peuplés), avec éventuellement des possibilités de migration d'individus d'une île à l'autre, ce qui permettrait, par ces échanges, de maintenir une certaine diversité dans l'environnement. Le principal intérêt de ce modèle est qu'il peut être parallélisé très facilement : on installe une île sur chaque processeur et l'ensemble peut traiter de manière simultanée la population complète. Ensuite, au bout de n itérations, on envoie $x\%$ de la population sur le processeur voisin, qui remplace une partie de la sienne.

2.10.4 Le modèle prédictif

On peut aussi citer le système de classifieurs prédictif de Sutton [Sutton, 1991], l'un des fondateurs de l'apprentissage par renforcement et son algorithme DynaQ de la famille Dyna. Dans un tel système, les règles de la forme :

Si $<m_t$ vérifie $c_t>$ **alors** $<a_t>$

sont remplacées par des règles de la forme

Si $<m_t$ vérifie $c_t>$ **et si** <l'action est $a_t>$ **alors** $<m_{t+1}$ vérifiera $c_{t+1}>$

où m est un message d'entrée, c une ou plusieurs conditions et a une action.

Au lieu de spécifier simplement une action, le classifieur prédit le contenu de son environnement local dans l'état de l'instant suivant. Un tel apprentissage est qualifié de *latent* par la littérature de la psychologie [Sigaud, 2002]. S. Wilson [Wilson, 1995] dit s'être d'ailleurs inspiré de ces travaux dans son système XCS, dans lequel le caractère prédictif des classifieurs joue un rôle important.

2.11 Conclusion

L'extraction de la connaissance sous la forme de règles de classification représente l'un des objectifs de nos travaux. Les systèmes de classifieurs présentent une approche intéressante concernant la manipulation et la découverte de la connaissance à travers leurs classifieurs. Dans ces systèmes, les classifieurs font partie intégrante du processus d'apprentissage et ne reproduisent pas le biais d'apprentissage provoqué par les méthodes inductives, par exemple, lorsqu'elles extraient des règles à partir d'une représentation figée par la méthode elle-même, comme c'est le cas pour les arbres de décision ou même les réseaux de neurones. Cependant, la représentation classique par une séquence de valeurs binaires pour les classifieurs n'est pas adaptée au problème de classification des données de télédétection, qui nécessite la manipulation de données continues. Une nouvelle amélioration de XCS sera présentée dans cette thèse, capable de manipuler des données réelles, ainsi que divers autres systèmes répondant aux mêmes objectifs. Notamment,

une technique confiant le choix de la représentation à l'expert (géographe, connaisseur du domaine, ...) à base de programmation génétique sera présentée. Ces techniques sont détaillées après avoir abordé le problème de la complexité des images de télédétection, ce que nous feront dans le chapitre suivant.

Chapitre 3

Des sources brutes et expertes aux pré-traitements des données

3.1 Énoncé de la problématique

Les satellites employés à l'heure actuelle autorisent de nombreuses utilisations dans un nombre tout aussi important de domaines, allant des télécommunications aux prévisions météorologiques, en passant par les exploitations militaires, la surveillance d'un déversement d'hydrocarbure ou l'étendue d'une inondation. L'une d'entre elles, la télédétection consiste à photographier une zone terrestre d'intérêt, avec une résolution spatiale pouvant atteindre l'ordre du mètre avec des satellites comme SPOT-5 ou CASI, voire de la dizaine de centimètres avec QuickBird (en mode panchromatique). Les images obtenues sont ensuite analysées dans le but de produire diverses cartographies, ou utilisées pour différentes études de classifications (présence de routes, d'immeubles, ...), de détections indirectes (détection de villes en fonction de la densité des routes) ou de quantifications des objets situés sur le terrain (en pourcentage par espèce de végétation par exemple).

La classification d'images est un vaste domaine qui recouvre parfois des sous-domaines à problématique plus spécifiques, comme la découverte d'une fonction par régression (découverte d'indices de végétation [Ricotta et al., 1999], problème inverse PAR [Robilliard et Fonlupt, 2001], ...). Nous n'allons pas détailler ici toutes les applications des recherches dans ce domaine, mais il est intéressant de noter qu'encore aujourd'hui, la production de ces cartes ou de ces fonctions est loin d'être toujours automatisée : elle nécessite le travail manuel, long et fastidieux d'un expert, obligatoire lorsque la précision est de rigueur (par exemple, les phases interactives lors de la production de cartes IGN), et dans le cas contraire elle s'appuie encore dans la plupart des cas sur des algorithmes statistiques qui ne sont pas forcément fiables, à cause de leur déterminisme ou de leur manque de robustesse.

Notre problématique principale est la classification supervisée d'images, c'est-à-dire l'extraction d'objets thématiques intéressants pour l'expert (immeubles, routes, ...) et leur localisation sur des images brutes, produites notamment par des satellites. Nous verrons (dans la section 3.4.1) que dans l'ensemble, ces données sont de taille importante et sont plutôt complexes. Notre processus de classification se compose globalement de trois phases : l'apprentissage à partir d'exemples, le partitionnement de l'image cible et l'affectation des partitions à chacune des classes d'intérêt. Elle se distingue de la segmentation, qui n'est qu'un partitionnement de l'image en composantes connexes uniformes selon un critère déterminé et de la catégorisation, en non-supervisé. Notre problématique sous-jacente est la découverte de *règles de classification* permettant d'une part, de décrire des concepts et d'autre part, d'identifier les instances de ces concepts sur des images de télédétection.

Les deux sections suivantes exposent le matériel dont nous avons disposé pour nos classifications. Les données brutes sont présentées dans la section 3.2 et les données expertes dans la section 3.3. Une discussion détaillée sur la complexité et le pré-traitement des données est présentée dans la section 3.4.

Enfin, deux études préliminaires concernant l'étude de la robustesse des algorithmes de classification face au bruit et l'intérêt d'utiliser une analyse par réduction de données pour réduire ce bruit sont présentées dans la section 3.5.

3.2 Données brutes

Les données brutes représentent l'ensemble du matériel de travail, c'est-à-dire celui qui va être concerné à la fois par l'apprentissage et l'exploitation des règles. Alors que les données expertes ne sont là que pour guider l'élaboration d'une classification, la qualité finale des règles dépend en grande partie de la qualité des données sources dont on dispose. Les données brutes utilisées dans notre étude se présenteront toujours sous la forme d'images de même résolution (spatiale) que les données expertes utilisées conjointement par notre algorithme. Nous allons voir dans cette section essentiellement des données dites *spectrales*, car pour chaque pixel, elles fournissent une information quantitative des valeurs radiométriques de ce pixel pour une certaine plage de longueurs d'onde.

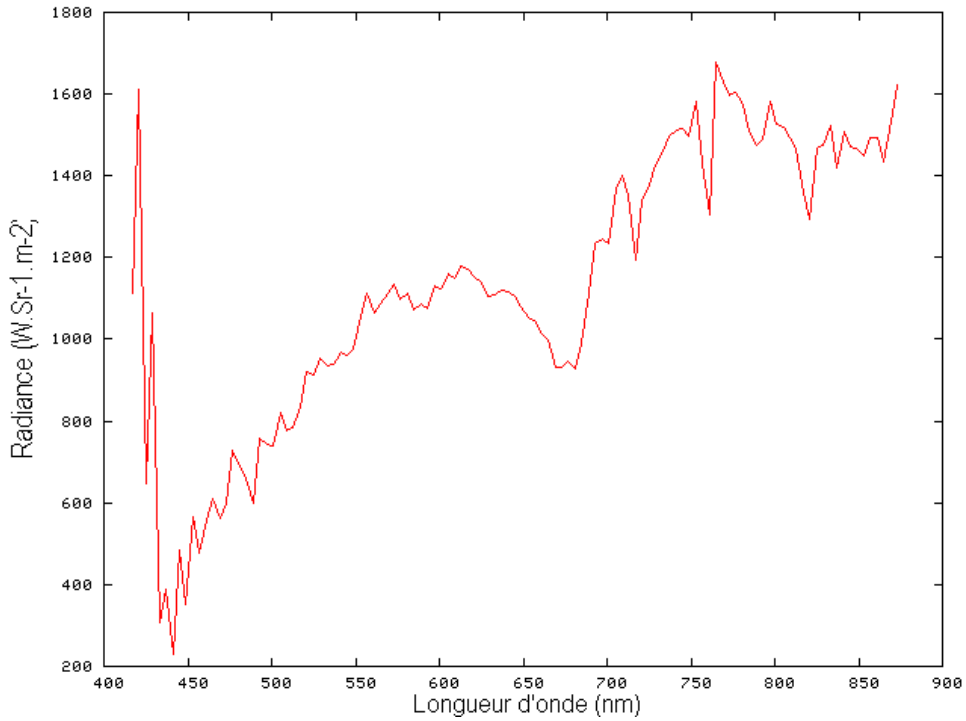


FIG. 3.1 – Spectre de radiance observé pour un pixel de végétation de l'instrument hyperspectral MIVIS.

Formellement, une image est une matrice à trois dimensions $I_{X,Y,S}$ où (X, Y) est respectivement la largeur et la hauteur de l'image et S le nombre de canaux spectraux (ou *bandes spectrales*). Une valeur $I(x, y, s)$ dans cet ensemble est la radiance observée pour le pixel situé à la localisation (x, y) et pour la longueur d'onde λ_s correspondant au canal spectral s. Une valeur de radiance correspond à l'intensité de la réponse radiométrique obtenue depuis le sol. L'espace d'entrée d'un problème de classification peut être vu comme un vecteur ordonné de nombre réels. Pour chaque pixel, la signature spectrale de ce pixel a été utilisée. La figure 3.1 montre le spectre de radiance d'un pixel observé depuis un capteur hyperspectral. Chaque canal spectral de MIVIS offre une résolution spectrale d'environ 10 nm.

Mis à part les différents types des supports (une matrice CCD n'a pas les mêmes caractéristiques qu'une pellicule photographique) et des outils disponibles pour les prises de vues (satellite, avion, ballon), il faut aussi parler des nombreux problèmes techniques que l'on peut rencontrer dans ces images : la

résolution spatiale, par exemple, n'est pas toujours adaptée aux objets au sol. Pour Strasbourg nous avions à notre disposition des résolutions plutôt faibles (30 mètres pour un satellite Landsat, 10 à 20 mètres pour un satellite SPOT), alors qu'un paysage urbain est une situation extrêmement complexe dans laquelle le pixel du capteur de télédétection peut renfermer quantité d'objets. La réponse spectrale obtenue sera une composition non-linéaire de spectres purs. On parlera alors de *pixels mixtes*. Lorsque le contraste entre un objet et son environnement est suffisamment élevé, cet objet pourra tout de même être détecté, même si sa taille est inférieure à celle du pixel. Il y a donc une relation entre d'une part la capacité théorique de détection d'un objet et d'autre part, sa taille et son contraste. Cependant, le capteur a ses limites, certains objets même très brillants sont parfois vraiment trop petits pour pouvoir être présents dans l'image.

Nous allons donner un bref aperçu des différents outils d'acquisition de données brutes exploitables. Le projet TIDE, de part sa nature, a été plus prolifique en données validées que nous n'ayons pu en obtenir pour Strasbourg, nous nous attachons cependant beaucoup à ces données car c'est une région que nous connaissons bien, ce qui facilite largement les validations.

3.2.1 Images SPOT

Les satellites SPOT (Satellite Pour l'Observation de la Terre) sont un ensemble d'instruments de surveillance de la Terre (SPOT5 a été lancé le 4 mai 2002) dont les images servent dans des domaines aussi divers que l'agriculture, la fabrication de cadastres, la planification urbaine, les télécommunications, l'ingénierie géologique et civile ou l'exploration de nappes de pétrole ou de gaz. Ce type de satellite est muni de 6000 micro-capteurs [SPOT, 2005a] permettant de prendre d'un seul coup une bande au sol de 60 km de longueur. La résolution spatiale réalisée est de l'ordre de 10 mètres de largeur par pixel, et de 20 mètres de largeur lorsque l'on couple ensemble deux détecteurs, ce qui permet d'avoir des images multispectrales [SPOT, 2005b]. SPOT5 autorise une résolution plus importante, de l'ordre de 2.5 mètres.

Les valeurs obtenues sont définies sur 8 bits, ce qui autorise 256 valeurs différentes par pixel et par bande. L'instrument VEGETATION est un ensemble de capteurs comprenant les mêmes caractéristiques sur SPOT4 ou SPOT5, et qui permet de conduire des études spécifiques sur l'environnement (cultures, biosphère, ...). Le tableau 3.1 présente les fréquences pour les 4 bandes spectrales d'analyse, adoptées pour la spécificité de la réponse spectrale de certains objets comme le sol, la végétation, le désert ou la neige.

Canaux	**Longueurs d'onde**
Voie visible verte V (XS1)	0.50 à 0.59 μm
Voie visible rouge R (XS2)	0.61 à 0.68 μm
Voie visible proche infrarouge PIR (XS3)	0.78 à 0.89 μm
Voie moyen infrarouge MIR	1.58 à 1.75 μm

TAB. 3.1 – Longueurs d'onde de SPOT-4 ou SPOT-5 [SPOT, 2005c].

On peut noter qu'un miroir plan présent dans le système de détection du satellite permet d'orienter la direction de visée, permettant de réaliser des couples d'images stéréoscopiques : cela permet notamment de restituer le relief du site observé, la variation et l'orientation de la pente, la hauteur des immeubles (influençant la taille de leur ombre), etc. Malheureusement ce type d'information, pourtant très importante mais aussi très coûteuse, n'a pas été disponible pour réaliser nos travaux.

3.2.2 Images haute résolution

Ces images d'une très grande résolution spatiale (un pixel représente au sol une taille de 1.30 m x 1.30 m) ont été fournies par le CNES en 1999, lors d'un vol de simulation depuis un avion volant à haute altitude, au cours d'un projet visant à tester les capteurs embarqués sur SPOT5 (les images contiennent donc le même nombre de bandes spectrales que le SPOT officiel). Des détails comme des arbres ou des

voitures sont tout à fait reconnaissables sur ce type d'image, ce qui paradoxalement peut compliquer leur classification, car ces petits objets peuvent être assimilés à du bruit (coloration différentes des voitures, tailles d'arbres variées, ...).

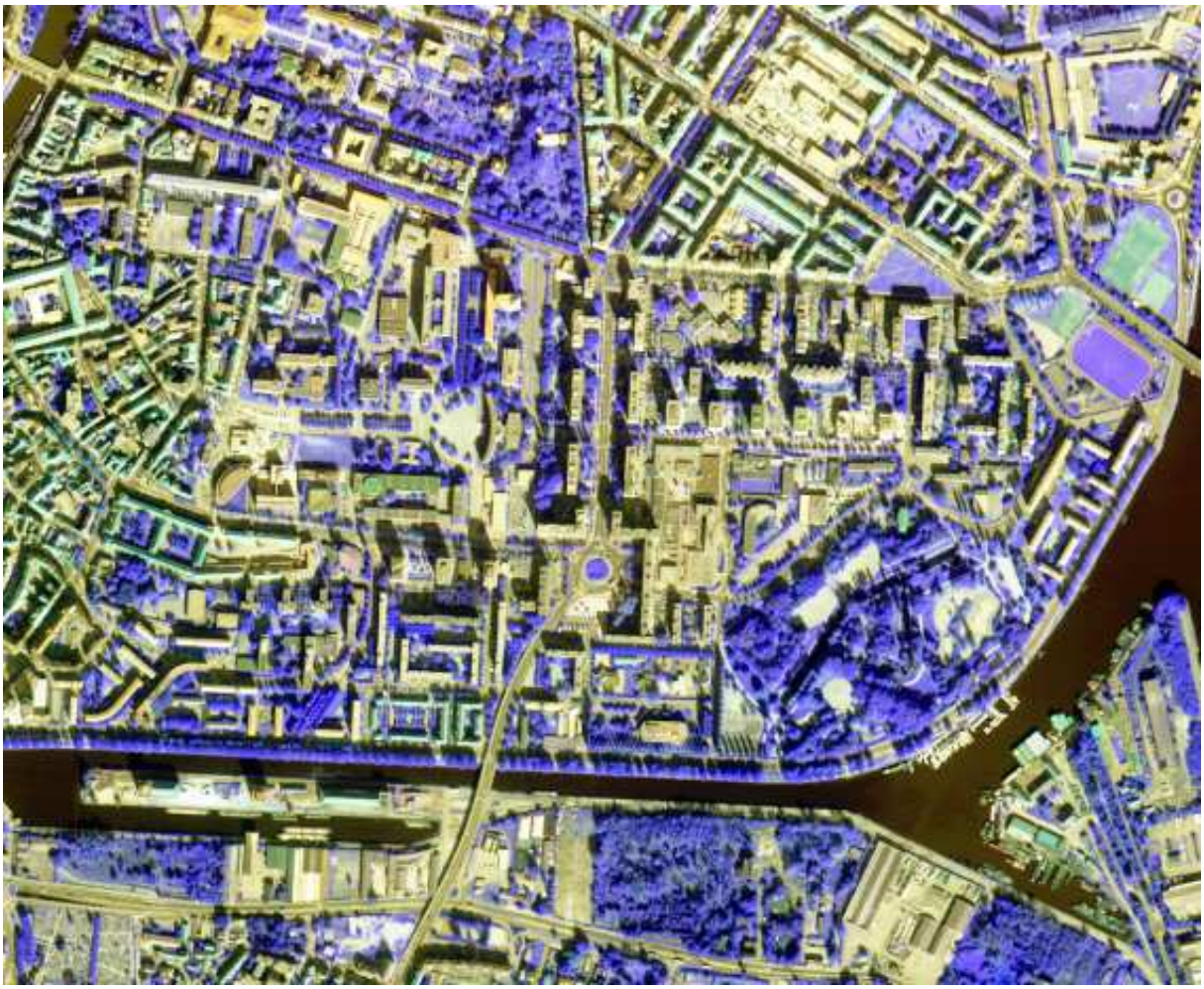


FIG. 3.2 – Image haute résolution de Strasbourg (simulations CNES).

Voici une image de Strasbourg (figure 3.2), représentant une partie à l'est du centre de la ville, d'une taille de 1100 x 900 pixels, dans laquelle on peut observer les quartiers Est de Strasbourg. L'un des intérêts de cette image, outre sa très grande résolution spatiale, est l'existence d'une image experte parfaitement synchronisée de la même zone (les pixels se superposent exactement), comme nous allons le voir dans la partie 3.3. La photo a été traitée par une égalisation d'histogramme afin d'en voir clairement les contours. On peut facilement imaginer les problèmes posés par la présence de zones ombragées, surtout dans un contexte urbain dans lequel les différents objets de la scène peuvent avoir des hauteurs non négligeables. Ces zones rendent difficile la reconnaissance des parties couvertes par l'ombre, car leur radiance est fortement amoindrie et altérée. Lors de la phase de reconnaissance, on préférera donc placer l'ombre dans une classe à part, plutôt que de déterminer celle des pixels sous-jacents. L'image de la figure 3.2 a été prise à 9 heures du matin, heure locale (présence d'ombres rasantes), dans des conditions favorables. Malgré son apparence, il ne s'agit pas d'une simple photo, mais bien des 3 bandes du capteur de SPOT, reproduites directement en fausse couleur. Le volume de cette image représente 3 Mo de mémoire.

3.2.3 Images hyperspectrales

À la différence des images SPOT, les capteurs de ce type de satellite sont capables de traiter un nombre très important de longueurs d'onde, de l'ordre de la centaine. Un pixel ne sera plus représenté par un triplet de valeurs comme dans le cas de SPOT, mais par un vecteur hyperspectral représentant l'ensemble des radiances pour toutes les longueurs d'onde du spectre étudié. Des systèmes aéroportés comme DAIS permettent une résolution spectrale de 79 canaux, une résolution spatiale de l'ordre de 3 mètres par pixel, et définissent les valeurs de radiance sur 16 bits, ce qui fait 65536 valeurs possibles [DAISEX, 2001]. D'autres satellites permettent une grande précision d'échantillonnage du spectre des pixels, comme CASI qui possède une étendue spectrale de 288 canaux avec une résolution spatiale de 1.3 m au sol [CASI, 2005].

Ces images ne pourront jamais être affichées telles quelles à l'utilisateur. La prévisualisation de ces images étant tout de même importante dans un souci de validation, des algorithmes d'affichage ont été développés, notamment au sein de notre équipe pour notre plate-forme d'expérimentation **ICU**. Ces algorithmes tiennent compte de l'information du spectre visible par l'homme pour déterminer une couleur réaliste des pixels, tout comme s'ils étaient visualisés directement par nos yeux depuis le ciel.

L'image suivante (figure 3.3) représente une photo prise par DAIS, un instrument aéroporté hyperspectral, le 17 juillet 1999 vers 18h50 (heure locale) après 6 semaines de pluie. Elle représente une zone couvrant Strasbourg depuis le lac de Reichstett au Nord (représenté à droite) jusqu'au terrain d'aviation du Polygone au Sud (représenté à gauche). Sa taille est de 512 x 2885 pixels, pour une résolution de 3m x 3m par pixel. Un algorithme spécifique pour la visualisation des images hyperspectrales a été utilisé pour produire un rendu réaliste à partir des 45 bandes spectrales de l'image (on a retiré 34 bandes des données qui en comptaient initialement 79 à cause de leur aspect extrêmement bruité). Les longueurs d'onde de ces bandes varient de 0.498 à 12.668 μm. Le volume de l'image de la figure 3.3 dépasse les 220 Mo.

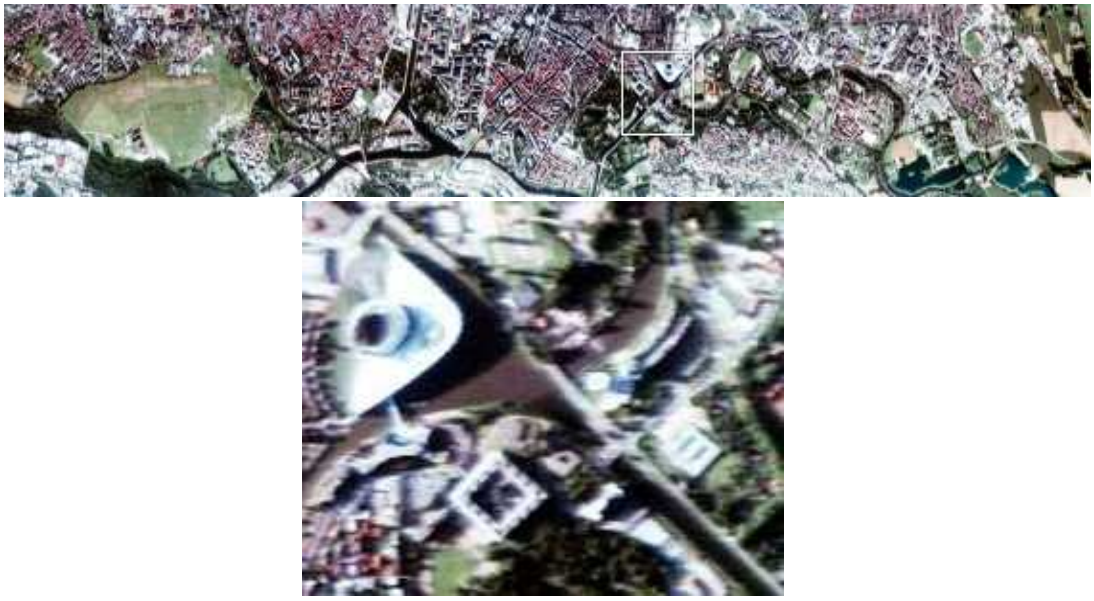


FIG. 3.3 – Image hyperspectrale de Strasbourg (DAIS) et un extrait réorienté représentant le Parlement Européen, le Palais des Droits de l'Homme et le Palais Européen.

Le dernier exemple représente San Felice (région de Venise, voir la figure 3.4). D'une taille de 932 x 368 pixels, elle a été prise par ROSIS, un autre instrument aéroporté hyperspectral, le 30 novembre 2001, avec une résolution d'environ 1m x 1m par pixel. L'image produite par le satellite avait 115 bandes mais, là encore, les 19 premières bandes devaient être enlevées avant tout traitement pour les mêmes raisons de bruit qu'auparavant. Les longueurs d'onde de ces bandes varient de 0,418 à 0,874 μm. Le volume en

mémoire de l'image est proche de 80 Mo.

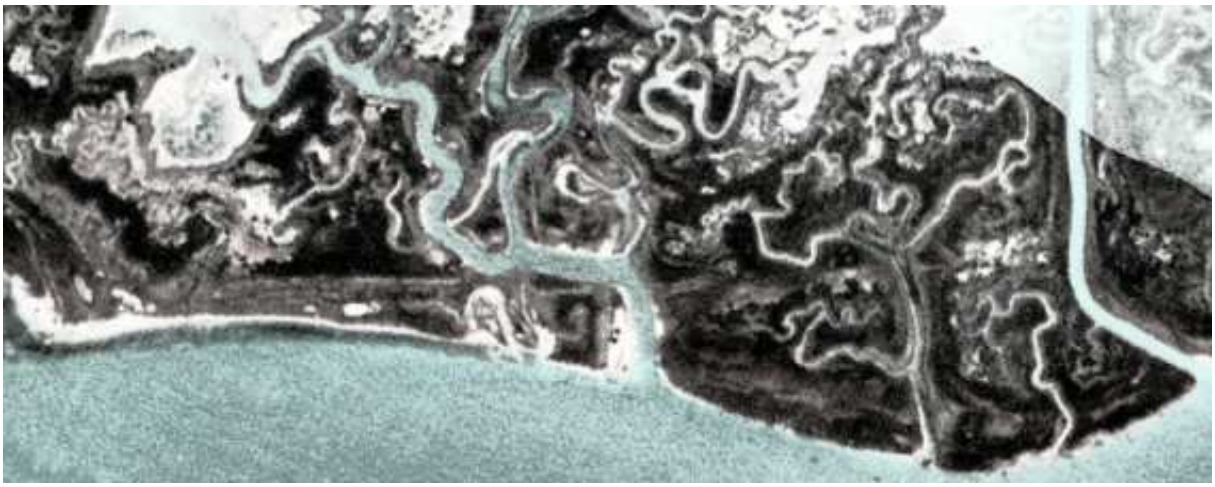


FIG. 3.4 – Image hyperspectrale de la lagune de San Felice (ROSIS).

Nous avons travaillé sur bien d'autres images multispectrales ou hyperspectrales, chacune possédant leurs intérêts propres, mais nous ne les avons pas tous détaillées ici, car ce n'est pas l'objet de cette thèse. Le tableau 3.2 présente un résumé des principales caractéristiques des autres images utilisées dans le cadre du projet TIDE.

Type	Instrument imageur				
	Nom		Caractéristiques		
			Canaux	Plage (λ)	Sensibilité
Satellite	QuickBird 2002	Panchrom.	1	0.45 - 0.90 μm	Panchrom.
		Multispectral	4	0.45 - 0.90 μm	Visible, NIR
Aéroporté	CASI 2003	Hyperspectral	288	0.4329 - 0.8741 μm	Visible, NIR
	MIVIS 2002	Port 1	20	0.433 - 0.833 μm	Visible, NIR
		Port 2	8	1.15 - 1.55 μm	Middle IR
		Port 3	65	2.000 - 2.492 μm	Middle IR
		Port 4	9	8.20 - 12.70 μm	Thermal IR
	LIDAR 2003		2	1^{ere} ou dern. impulsion	Elevation

TAB. 3.2 – Base de données des images du projet TIDE.

3.3 Données expertes

La phase d'apprentissage supervisée a besoin au préalable, pour suivre un déroulement correct, d'un certain nombre d'informations supplémentaires sur les classes à trouver, dites informations expertes. Ces informations peuvent être produites de manière indépendante et de plusieurs façons :

- L'expertise humaine, très coûteuse, permet néanmoins une analyse précise de la zone observée. Un expert ne tombera pas dans les pièges faciles d'une analyse automatique, lorsqu'il s'agira de classer des cas rares comme la présence d'une péniche remplie de bois sur de l'eau, ou la présence d'une piscine sur un immeuble. Son expertise se révèle être toujours convenable, grâce à une éventuelle validation directement sur le terrain pour confirmer les décisions lors des situations extrêmes.

- L'analyse statistique ou non supervisée produit un ensemble de classifications de manière systématique et rapide, mais pas toujours fiable. Ces analyses résultent d'algorithmes qui ne sont jamais parfaits : ils ont leurs domaines de prédilection donc, à l'inverse, leurs domaines de vulnérabilité. Bien entendu, ces classifications peuvent être validées par un expert humain, mais cela est irréaliste du point de vue de la taille des images.
- L'exploitation d'images de télédétection pose souvent des problèmes non triviaux, que les méthodes statistiques ne peuvent pas résoudre. Pour aider davantage le processus d'extraction de connaissances, l'introduction de données sémantiques, structurelles ou contextuelles supplémentaires, comme par exemple la fréquence de voisinage de l'eau avec des bâtiments bétonnés (notion de texture), la forme régulière desdits bâtiments ou leurs relations dans l'environnement global de l'image (notion de contexte) peuvent bien entendu être utilisées sous forme de règles de composition, mais nous n'entrerons pas dans ce domaine qui oscille encore entre la spéculation et la manipulation d'ontologies complexes.

Les informations expertes que nous possédons se répartissent en plusieurs catégories :
- les données observées directement sur le terrain (ang., *ground-truthing*) [Marani et al., 2005],
- celles sélectionnées et validées par l'expert en laboratoire qui éventuellement choisi les ensembles de test et d'apprentissage, nommés ROI (ang., *Region of Interest*) [Silvestri et al., 2002],
- et enfin dans le pire des cas, des classifications automatiques (supervisées ou non).

3.3.1 Données expertes obtenues de manière manuelle

Ces données rassemblent un certain nombre de points (souvent peu nombreux) caractérisés par un expert humain. Ces points, nommés *ground-truthing* mettent en relation une position réelle (obtenue par un GPS) avec un certain nombre de classes. Plus formellement, à partir d'un ensemble de points d'une image I et d'un ensemble de classes C, on défini l'expertise $E : I \rightarrow C$ comme suit :

$$\omega(x, y) \rightarrow \{c_i \in \{c_1, c_2, \ldots, c_n\}\} \tag{3.1}$$

où $\omega(x, y)$ est l'expertise du pixel (x, y) et c_i la classe de ce pixel.

Il s'agit souvent d'une expertise de qualité (désignée par le terme *gold standard*) mais fastidieuse à obtenir, d'où sa rareté. Actuellement, l'expertise tend de plus en plus à fournir une identification plus précise des objets observés, par exemple en affectant pour chaque pixel le pourcentage estimé de sa composition pour toutes les classes d'étude.

La figure 3.5 montre un exemple de données expertes. Chaque point est affecté à une classe et représente la relation entre une donnée GPS et une donnée radiométrique (obtenue avec un spectromètre au sol). Souvent, pour des raisons de temps d'acquisition, uniquement les zones frontalières de deux espèces de végétations sont marquées (formation de polygones thématiques).

Ces points sont ensuite sélectionnés en laboratoire pour former les régions d'intérêt (ang., ROI ou *Region of Interest*). Les points sont ajustés, les polygones sont fermés et sont labellisés par la description de la classe. Après ce traitement, les zones ne se recouvrent plus. La figure 3.6 montre une telle structure.

3.3.2 Classifications obtenues de manière automatique

Lorsque l'information experte de terrain n'est pas disponible, nous avons utilisé des classifications obtenues à l'aide de segmentations ou de méthodes statistiques. Ces classifications ont toujours été affinées et validées à la main par un expert connaissant la région.

La carte présentée sur la figure 3.7 a été extraite directement à partir de l'image haute résolution présentée sur la figure 3.2. L'un des avantages de la technique automatique est le fait que la zone représentée est synchronisée et la recouvre entièrement. Cependant, nous avons dû la corriger pour deux raisons : premièrement la présence de pixels uniques et isolés, constituant des classes à part entière, risquait de perturber fortement notre algorithme de classification. Ces pixels ont donc tous été rattachés à des classes existantes et quantitativement mieux représentées dans l'image, en fonction des classes voisines et du bon sens. Deuxièmement, compte tenu de la réponse spectrale pratiquement identique pour l'eau

FIG. 3.5 – Données expertes (*ground truthing*) pour San Felice (MIVIS).

FIG. 3.6 – Régions ROI (ang., *Region of Interest*) comprenant des ensembles de points sélectionnés par l'expert pour San Felice (MIVIS).

et pour l'ombre, certaines rues du centre ville étaient considérées comme inondées. Strasbourg n'étant pas Venise, il a donc fallu manuellement remplacer toutes les régions noyées par des zones ombragées. Cependant, ce traitement se révèle être très laborieux dans certaines parties de l'image, par exemple au niveau du pont situé au milieu du tiers inférieur de l'image : la présence de pixels d'eau et d'ombre (des spectres très proches vis à vis de la résolution spectrale de SPOT) sont ici mêlées, et cette partie de l'image n'a pas été traitée en apprentissage.

Quand on analyse la réponse spectrale de nombreux échantillons provenant du milieu urbain ou autre, on observe au moins trois grandes catégories distinctes discriminant ces échantillons :

– ceux se rapportant au végétal, à la matière vivante : la courbe spectrale dans le domaine du proche

FIG. 3.7 – Image experte de Strasbourg (segmentation).

infrarouge (canal NIR) présente un pic au niveau des longueurs d'onde de la plage 0.530 à 0.560 μm (couleur verte), dû à la présence de chlorophylle dans la végétation,
- ceux se rapportant au minéral, à la matière inerte : la courbe spectrale est croissante et quasi-linaire dans le domaine du visible et se prolonge même au-delà dans l'infrarouge,
- et l'eau qui se distingue profondément des catégories précitées : sa réponse spectrale se caractérise globalement par une courbe décroissante dans le domaine du visible (en considérant des spectres représentés avec les longueurs d'onde croissantes en abscisse et les valeurs de radiance croissantes en ordonnée).

Concernant le projet TIDE, l'image de la figure 3.8 présente l'une des classifications supervisées que nous avons utilisées lorsque les régions d'intérêt expertes n'ont pas été disponibles, ou simplement pour valider les résultats de nos algorithmes évolutifs. Elle a été obtenue par un algorithme nommé SAM (ang., *Spectral Angle Mapper*), classique, connu dans la littérature en télédétection et couramment implanté dans les systèmes commerciaux [Yuhas et al., 1992].

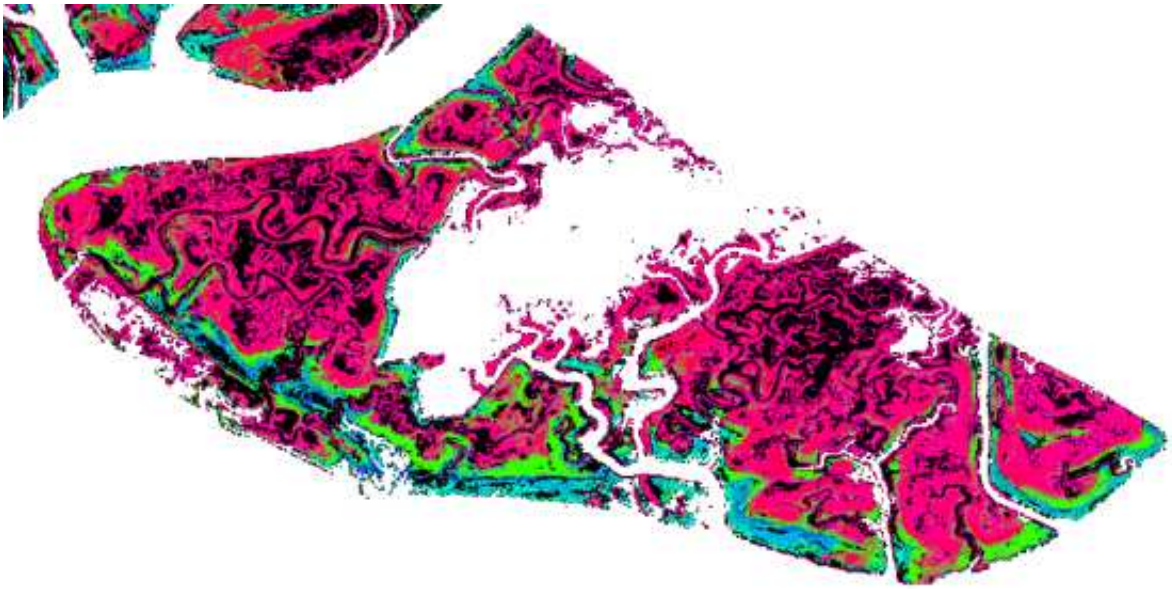


FIG. 3.8 – Image utilisée en tant qu'expert, obtenue à partir d'une classification supervisée pour San Felice.

3.3.3 Propriétés attendues pour les données expertes

Les données expertes représentent une source d'information importante pour les algorithmes d'apprentissage supervisé. Voici une liste des propriétés attendues concernant les données expertes.

1. Les données dans le jeu d'apprentissage sont représentatives du nombre de classes, de leurs fréquences et de leurs variations.

2. Les données de référence (données brutes et expertes) sont parfaitement synchronisées par géoréférencement.

3. Il n'y a pas d'erreur dans les données de référence, c'est-à-dire une identification incorrecte des objets par l'expert, une évolution de la végétation au sol ou de la canopée entre la date d'acquisition de l'image et sa validation terrain, des erreurs de positionnement GPS, ...

4. En classification *hard* ou *soft* (voir la section 4.2), il n'y a pas de biais trop important entre la composition spectrale des mixels et le choix des classes durant l'identification (conversion d'attributs réels en attributs nominaux).

Travailler sur des données expertes de qualité est très important pour la fiabilité des règles de classification extraites par les algorithmes d'apprentissage. Malheureusement, dans la réalité, plusieurs de ces propriétés apparaissent comme très optimistes [Verbyla et Hammond, 2002]. Les données que nous avons eues à manipuler présentaient une qualité parfois variable. Cependant, cette qualité peut être améliorée par l'application d'un certain nombre de pré-traitements, que nous présentons dans la section suivante.

3.4 Complexité et pré-traitements des images spectrales

3.4.1 Considérations sur la complexité des données

Les données brutes que nous avons vues jusqu'alors ne peuvent pas être utilisées directement dans les algorithmes d'apprentissage. En effet, les images de télédétection présentent des difficultés que nous nous proposons de rassembler ici. Heureusement, des pré-traitements sont disponibles permettant d'en contourner un grand nombre. Ils seront présentés dans la partie suivante. Ces difficultés sont :

Une taille souvent large. Les images sont de plus en plus larges, permettant de couvrir une plus grande zone à coût moindre, tout en évitant les sur-coûts de temps et d'imprécision à devoir gérer le « mosaicking[1] ». Souvent le recollement de bandes laisse apparaître des différences de contraste et d'intensité sur les sites frontaliers, qui sont absentes des images plus larges. Cependant, de larges images nécessitent des processus de traitement plus expansibles, adaptés à une grande masse de données, et peuvent faire apparaître des effets de distorsion géométrique[2]. En général, l'utilisation de petites images est préférable. La taille dépend de la résolution du capteur, de quelques dizaine à quelques centaines de kilomètres de fauchée et peut conduire à produire des images de plus de 12 000 pixels de côté. Du point de vue de la durée du traitement, il est préférable que la complexité des algorithmes soit linéaire en fonction du nombre de pixels.

Une résolution élevée. Il existe quatre types de résolution :

1. La résolution **spatiale** mesure la taille réelle d'un objet correspondant à un pixel de l'image. Pour les applications de télédétection, elle varie de 60 cm pour les instruments les plus performants (QuickBird) à environ 4 km.
2. La résolution **spectrale** concerne le nombre et la finesse des canaux (ou *bandes spectrales*) c'est-à-dire le nombre et la taille des *fenêtres* de longueur d'onde distinctes. On distingue les capteurs *panchromatiques* ne capturant qu'un seul canal (en niveau de gris), les capteurs *multispectraux* (2-20 bandes), *hyperspectraux* (100-300 bandes) et *ultraspectraux* (quelques milliers de bandes). À l'heure actuelle, ces derniers capteurs ne sont pas encore développés. La figure 3.1 (page 40) montre un exemple du spectre de radiance d'un pixel depuis un capteur hyperspectral.
3. La résolution **radiométrique** concerne le nombre de niveaux de radiance détectés par le capteur. Par exemple, une image numérique classique délivre 256 niveaux de gris par canal. En télédétection, la résolution radiométrique d'un capteur dépend de sa sensibilité et de sa précision : on peut avoir des valeurs sur 8, 16 ou 32 bits par canal.
4. La résolution **temporelle**, qui dépend du temps d'un cycle orbital[3] dans le cas d'un satellite. Avoir une bonne résolution temporelle n'est importante que pour les phénomènes de courte durée dont on veut observer la périodicité (inondation, déforestation, dégazage, ...), ou lorsque l'on souhaite analyser plusieurs images simultanément ce qui n'est pas le cas de notre problématique dans le projet TIDE.

Toutes ces données sont souvent nécessaires simultanément à un système d'apprentissage pour lui donner une chance correcte de discriminer selon les bons critères. La taille de ces données est donc directement liée au produit de l'information apportée par chaque résolution, et peut facilement atteindre le giga-octet.

Une complexité non linéaire. Les spectres obtenus, malgré une résolution spectrale élevée, ne seront jamais comparables à des spectres parfaits, de laboratoire. De nombreuses interactions environnementales interviennent et dégradent le signal perçu, ce qui peut le rendre dissemblable par rapport à un étalonnage ou aux validations terrain. On peut signaler l'influence des saisons, des conditions

[1] Ensemble de *fauchées* (ang., *stripes*) distinctes obtenues par passage successif du satellite ou de l'instrument imageur ayant des zones de recouvrement plus ou moins grandes. Leur recomposition s'appelle l'opération de *recalage*.

[2] Le sommet de l'objet (immeuble, ...) au centre sera vu correctement tandis que les objets éloignés laisseront apparaître, en plus, l'un de leurs côtés.

[3] Passage au dessus d'un même point au sol, ou *nadir*.

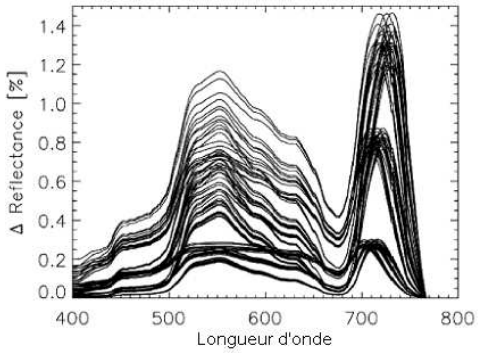


FIG. 3.9 – Effet de l'humidification sur la transformation progressive du spectre d'une espèce végétale : les spectres s'aplatissent en présence d'eau, [Leone et Menenti, 2000].

d'illumination terrestre, de la position des plantes ou des feuilles renvoyant des spectres différents pour la même espèce végétale comme sur la figure 3.9, montrant les altérations spectrales observées en fonction du taux d'humidification des feuilles. Enfin, signalons le problème des *mixels*, ces pixels qui renferment plusieurs types d'objets différents de résolution supérieure en mêlant leurs spectres respectifs de manière non-linéaire [Foody et al., 1997]. Cette complexité nécessite donc des modèles performants qui ne se contentent pas simplement d'associer des classes pures aux pixels étudiés mais de caractériser les modifications de ces spectres purs, loin d'être linéaires.

FIG. 3.10 – Canaux bruités typiques : respectivement Strasbourg (bande numéro 42 et 49 de DAIS) et la Lagune de Venise (bande numéro 45 de DAIS).

Des données bruitées. Le bruit des images intervient à plusieurs niveaux. Tout d'abord l'image brute est toujours entachée de nombreux défauts. Par exemple, quelque soit l'instrument imageur, qu'il s'agisse d'un satellite, d'un avion ou d'un ballon, les distorsions géométriques sont toujours présentes. Cela peut dépendre de l'optique du capteur, du relief en trois dimensions de la surface de la Terre ou de sa rotation qui peuvent causer de légers décalages à l'Ouest (*distorsion oblique*). D'autres types d'artefacts peuvent apparaître, par exemple l'absorption atmosphérique des valeurs de radiance dans les faibles longueurs d'onde, quand il ne s'agit pas des capteurs eux-mêmes (défectuosité, *memory effect* de Landsat, vignetage, ...). La figure 3.10 présente quelques canaux bruités typiques. Une image hyperspectrale classique peut en contenir jusqu'à 50%.

Les objets représentés peuvent être eux-mêmes source de perturbations. Par exemple, la structure en 3D du milieu urbain offre une complexité qui provoque certaines particularités au niveau du signal reçu par les capteurs : ombres ou réflexions des façades, camouflage de différents éléments

urbains, effets directionnels des matériaux urbains, diversité des réponses pour le même type d'objet en résolution élevée.

Ensuite, les données expertes comprennent aussi un certain nombre d'imprécisions notamment à cause du choix du site de validation, de l'échantillonnage des points et de la précision humaine.

Une expertise concise. Enfin, nous devons signaler que l'information experte, fondamentale à tout traitement supervisé, est souvent peu nombreuse, comparée à la taille des images à classifier, même si elle est d'une très bonne qualité. Ingrate, longue et coûteuse en terrain réel, l'expert choisira de valider de préférence des zones de terrain stables et pures, de façon à avoir une correspondance maximale avec l'image capturée. Ces zones stables et pures ne sont pas forcément représentatives de tous les mélanges au sol, ce qui réduit d'autant plus les chances de succès des algorithmes d'apprentissage, qui fonctionnent selon un principe hétéro-associatif, les modèles connexionnistes en particulier.

Enfin, un autre type de verrou à la classification d'images est le fait que les données entretiennent une certaine complexité relationnelle. Différentes sources de données (que ce soient des images brutes ou des données expertes) peuvent avoir une relation forte entre elles. Un exemple simple est le cas des images multi-échelles. La couverture du terrain ne se fait habituellement pas par une prise de vue unique, qu'elle soit satellitaire (SPOT, LANDSAT) ou aérienne (DAIS, ROSIS). Les raisons en sont essentiellement : (1) la faculté de pouvoir appliquer des indicateurs différents sur des sources différentes afin de promouvoir la complémentarité des mesures pour la classification et (2) la calibration de ces mesures pour la même région. Bien que l'on pourrait se contenter d'un seul couple images sources - images expertes par classification supervisée, les nouvelles thématiques de recherche traitées par les spécialistes en télédétection abordent des méthodes qui leur permettent d'intégrer plusieurs sources d'informations simultanément. La littérature a abondamment souligné l'existence de cette nouvelle piste [Bijaoui et al., 1994; Murtagh et Starck, 1999; Sharon et al., 2000], sans toutefois pouvoir proposer des solutions satisfaisantes pour l'emprunter, même si quelques résultats ont été obtenus, notamment dans la segmentation de textures de paysages naturels [Metzler et al., 2000]. Ce problème est traité activement dans l'ACI Fodomust [FoDoMuST, 2005] et ne sera pas développé ici.

Du point de vue de l'expertise, une même donnée peut réunir un ensemble d'informations contradictoires pour le même pixel. Ce problème intervient, par exemple, lorsque la résolution de l'image n'est pas assez fine et que la région en question contient plusieurs espèces végétales différentes. Le pixel (ou sa représentation sous forme de signature spectrale) est alors une composition, pas forcément linéaire, de plusieurs essences de végétaux, se traduisant par une *mixture* des spectres respectifs. Ce type d'information experte est appelée *information non consensuelle* sur *mixel*. À l'heure actuelle, il n'existe aucun modèle fiable permettant de produire les compositions spectrales de tels *mixels*. Les caractéristiques de ces informations devraient être prises en compte à la fois lors du traitement des données brutes (classification) mais aussi lors de l'apprentissage, au niveau de l'expertise. Les méthodes de traitement doivent intégrer le fait que l'expertise peut se composer de valeurs floues, ce qui ajoute encore à la complexité ambiante des données.

Toutes ces considérations mettent en avant la spécificité des données rencontrées dans un problème de classification d'images satellites, ainsi que les diverses difficultés l'accompagnant. La partie suivante présente la liste des pré-traitements qui ont pu être utilisés lors de l'adaptation de ces images brutes à notre algorithme de classification.

3.4.2 Pré-traitements

Certains pré-traitements des images sont effectués par les fournisseurs. Il peut s'agir d'*étalonnages*, de *corrections géométriques* (correction de la déformation par le relief de la Terre), de *corrections atmosphériques*, de *géoréférencement* par capteur GPS et de *recalages* (assemblage des différentes fauchées). D'autres pré-traitements ont été effectués par des librairies développées dans notre équipe. Concernant les données brutes, nous pouvons citer l'épuration des canaux (suppression des plus mauvais, amélioration des autres), la sélection des zones d'intérêt (ang., *cropping*) et certains filtrages statistiques (égalisation

d'histogramme, correction du contraste, ...). Concernant l'expertise, les deux pré-traitements principaux sont le calcul des enveloppes convexes et la *catégorisation* des classes : pour gagner du temps, les experts ne constituent souvent au GPS que des polygones thématiques (ang., *patches*), délimités par un certain nombre de points situés sur la frontière de la zone qu'ils souhaitent labelliser. Il s'agit alors d'appliquer ces polygones sur l'image, de calculer leurs enveloppes convexes (algorithme QuickHull [Graham, 1972; Preparata et Shamos, 1985]) et d'intégrer les pixels aux exemples de référence. Un certain nombre de points, situés sur la frontière, ne doivent pas être pris en compte, car ils peuvent appartenir à plusieurs espèces de végétation différentes mais concomitantes.

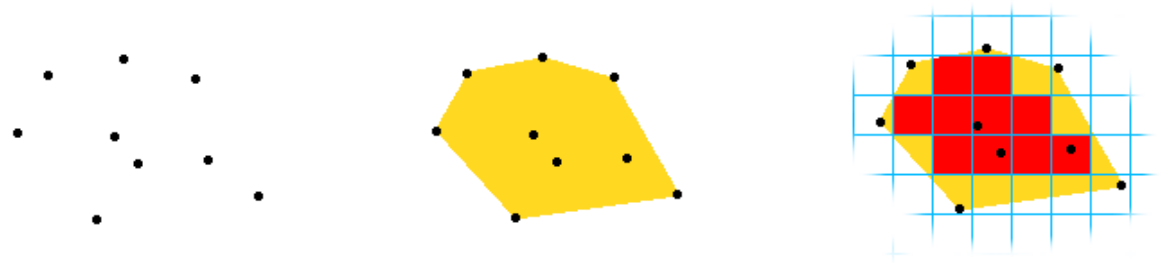


FIG. 3.11 – Principe du traitement des polygones thématiques, par calcul de l'enveloppe convexe.

La figure 3.11 illustre ce procédé automatique : les pixels rouges (carrés sombres) sont ceux effectivement pris en compte. La catégorisation concerne la labellisation des *mixels* par des classes pures ou des combinaisons virtuelles de classes. Par exemple, pour un *mixel* $M = \{c_1 : 85\%, c_2 : 10\%, c_3 : 5\%\}$ formé d'une composition spectrale de 3 classes $\{c_1, c_2, c_3\}$, on peut soit considérer que M appartient à la classe dominante c_1, soit créer une classe virtuelle $c_4 = \{c_1 : \frac{85}{85+10}\%, c_2 : \frac{10}{85+10}\%\} = \{c_1 : 89\%, c_2 : 11\%\}$ représentant ce type de composition, par exemple dans le cas où elle serait fréquente. Généralement, un *mixel* est dit pur si la proportion d'une espèce est d'au moins 90%. Selon le type et la robustesse des algorithmes, on peut être amené à supprimer – des données de référence – les *mixels* dont la classe dominante à une proportion autour de 50%.

3.5 Études approfondies

3.5.1 Réduction des données

Compte tenu de la forte taille et de la haute résolution spatiale et spectrale des données, il est intéressant de noter qu'il existe des techniques de réduction de données judicieuses dans notre cas. L'une d'entre elles est sans doute l'analyse par composantes principales (PCA ou ACP).

Il s'agit d'une technique permettant de transformer un système d'équations linéaires en un autre de telle sorte que les termes (appelés *composantes*) ayant les coefficients les plus forts soient situés en premier. Cette transformation linéaire consiste à projeter chaque point dans un autre système d'axes orthogonaux. Les dernières composantes ont souvent des coefficients faibles, ce qui permet de les éliminer et donc de réduire les données.

La figure 3.12 représente les 100 canaux les moins bruités d'un extrait de l'image de ROSIS présentée sur la figure 3.4, page 44. L'image hyperspectrale originale en comporte 115. On peut noter trois types de bandes : celles qui sont bruitées (type I), celles présentant la végétation sous un aspect sombre (type II) et celles la montrant sous un aspect clair (type III).

La figure 3.13 montre la fidélité de reconstruction en utilisant les N composantes obtenues en utilisant l'ACP. Q_p est la qualité pratique et indique la similarité normalisée entre l'image reconstruite et l'image originale et Q_t est la qualité théorique, obtenue en calculant la somme normalisée des valeurs propres (ang., *eigenvalues*). Les moyennes sont calculées sur un échantillon de 3000 points choisis aléatoirement

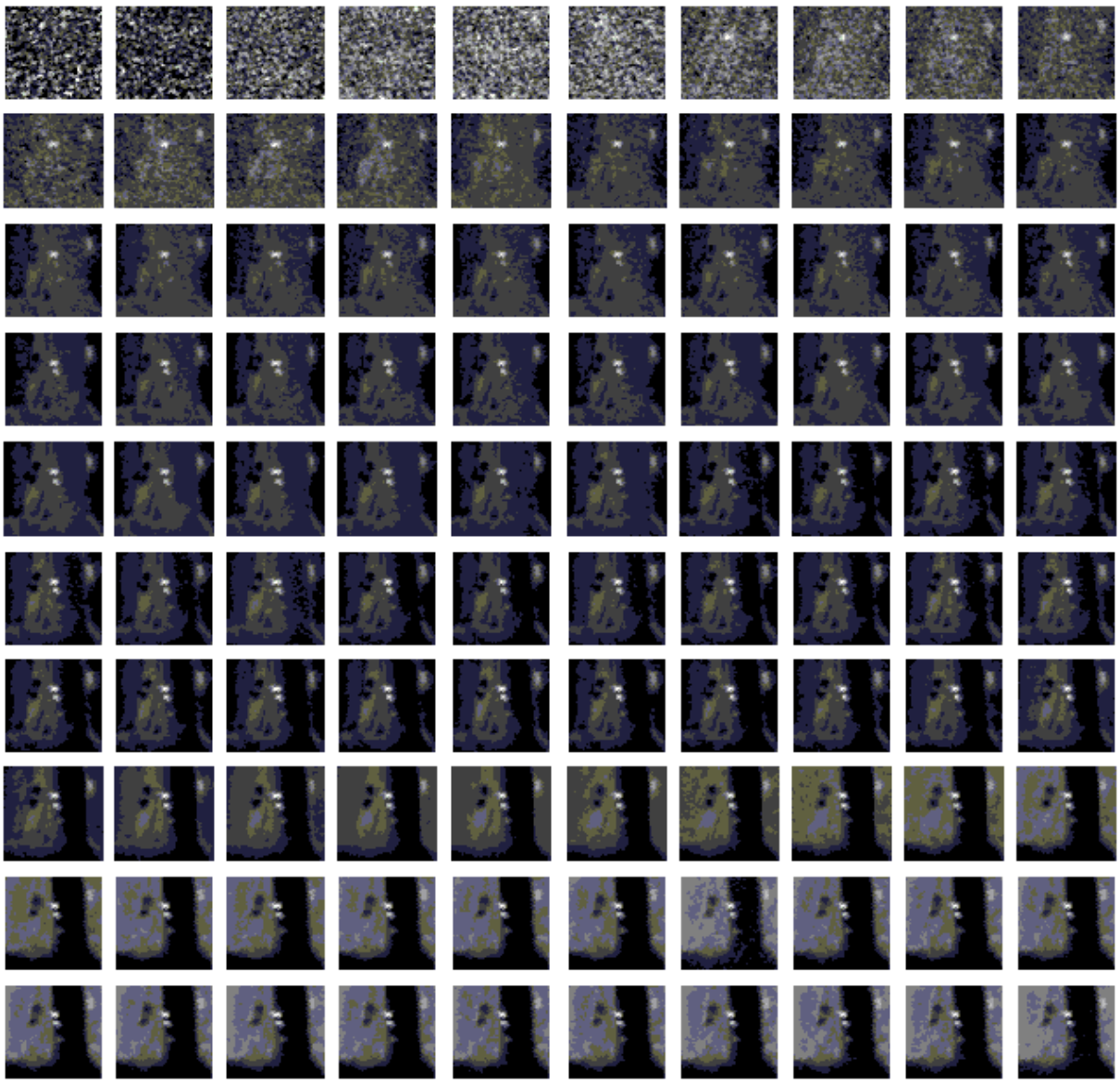


FIG. 3.12 – Les 100 canaux les moins bruités de ROSIS.

dans les données. On observe que 95% d'information se trouve dans les sept premières composantes. La figure 3.14 illustre le traitement de l'ACP pour ces premières composantes : chaque composante est représentée par un ruban coloré. Chaque ruban indique pour chacune des bandes (axe horizontal à gauche) les valeurs du vecteur propre correspondant (ang., *eigenvector*). La première composante intègre donc les canaux de 1 à 50 (du type II) et la seconde composante les canaux 51 à 115 (du type III).

La figure 3.15 présente les quatre premières composantes. L'observation des sept premières composantes, qui contiennent toute l'information, indique que seules les deux premières ne sont pas bruitées. Des tentatives successives de classification de ces données ont toujours échouée, la qualité décroissant significativement lorsqu'on manipulait plutôt l'image par composantes que l'image originale (par exemple, avec l'algorithme évolutif **ICU**, présenté par la suite, le taux d'exemples correctement classifiés passait de 0.82 à 0.39 soit une diminution de 52%). En fait, il semblerait que le bruit soit considéré comme une

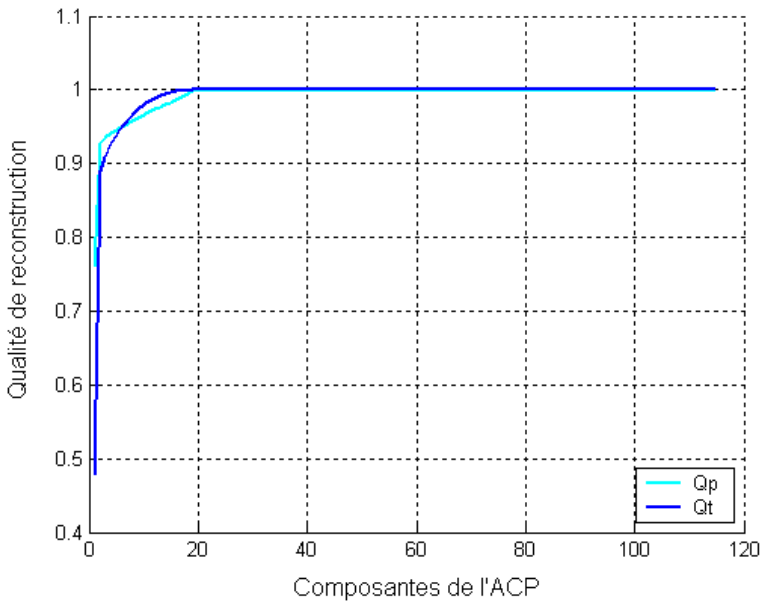


FIG. 3.13 – Fidélité de reconstruction en utilisant l'ACP sur l'image de ROSIS.

information beaucoup trop tôt, car dans ce type d'images, la composante bruitée du signal est souvent présente de manière corrélée sur plusieurs bandes, et se fait donc extraire en une composante majeure par l'ACP. Les bandes restantes ne contiennent pas suffisamment d'informations pour classifier l'image.

Il est cependant possible que l'ACP donne de meilleurs résultats avec d'autres images ou d'autres algorithmes, par exemple, en non supervisé. Cependant, l'ACP est aussi utile dans d'autres cas qui dépassent notre étude : le suivi temporel consistant à comparer deux images prises à des dates différentes (les informations intéressantes se trouveront dans les dernières composantes) ou la fusion d'images à des résolutions spectrales différentes. Cependant, un inconvénient subsiste : la transformation des données par l'ACP résulte en une perte de l'interprétabilité spectrale pour l'expert [Velez, 2003], ce que nous considérons comme une perte d'information non acceptable dans notre cas.

D'autres méthodes comme l'ACI (Analyse par Composante Indépendante) ou la PP (ang., *Projection Pursuit*) peuvent être utiles, nous n'avons cependant pas mené d'autres investigations à ce sujet. Par exemple, une étude a été conduite en utilisant l'ACI, sans obtenir toutefois des résultats déterminants pour le moment [Lennon et al., 2001]. Dans nos expérimentations, nous nous sommes contentés de réduire les données en utilisant de simples échantillonnages, mais basés sur les nombreuses recommandations des experts, notamment concernant la répartition statistique des classes et des valeurs extrêmes.

3.5.2 Analyse de la robustesse face au bruit

Comme l'étude précédente tend à prouver qu'il existe des données qui rendent difficile l'utilisation d'une ACP en pré-traitement, il restait à savoir si les algorithmes employés ont une chance d'être suffisamment robustes au bruit.

L'algorithme A2 présente un protocole de validation d'une méthode d'apprentissage face à des données bruitées. Il a été testé sur l'image de ROSIS en sélectionnant, pour le jeu de référence S, un échantillonnage de points validés correspondant à 10% de toute l'image (c'est-à-dire 34297 points). Cet algorithme appelle la méthode d'apprentissage sur des sous-ensembles d'exemples bruités (50% en apprentissage, 50% en validation) et calcule la performance moyenne sur un certain nombre n de tests. La méthode d'apprentissage doit être paramétrée par l'utilisateur. Nous avons choisi la méthode **ICU**

ALGORITHME A2
VALIDATION D'UN ALGORITHME FACE À DES DONNÉES BRUITÉES

$\rightsquigarrow$ PARAMÈTRES - S représente les données, n est le nombre de cycles de tests, ρ_d est le pourcentage de bruit appliqué sur les données brutes et ρ_e sur les données expertes
$\rightsquigarrow$ RÉSULTAT - Q, la qualité des règles apprises
$\rightsquigarrow$ RANDOM(S,τ) renvoie de manière aléatoire et en les supprimant de S (tirage sans remise), k_1 exemples de telle sorte que $k_1 = \tau\Omega(S)$ où $\Omega(S)$ est le cardinal de l'ensemble S.
$\rightsquigarrow$ NOISE(S,ρ) perturbe les données en modifiant k_2 exemples (permutation des attributs ou des pixels) de telle sorte que $k_2 = \rho\Omega(S)$
$\rightsquigarrow$ LEARNING(D,E) est une méthode d'apprentissage déjà paramétrée qui renvoie un jeu de règles permettant d'inférer E lorsque l'on a D
$\rightsquigarrow$ VALIDATION(D,E,R) est une méthode validant une base de règles R par rapport à un jeu de référence (D, E)

soit $Q_\mu := 0$
pour i *de 1 à* n ***faire***
  $\rightsquigarrow$ Altération des données d'apprentissage
  $(Data_l, Expert_l) :=$ RANDOM$(S, 0.5)$
  $\rightsquigarrow$ Altération des données de validation
  $(Data_v, Expert_v) :=$ RANDOM$(S, 1.0)$
  $Data_{Nl} :=$ NOISE$(Data_l, \rho_d)$
  $Expert_{Nl} :=$ NOISE$(Expert_l, \rho_e)$
  $Rules :=$ LEARNING$(Data_{Nl}, Expert_{Nl})$
  $Q :=$ VALIDATION$(Data_v, Expert_v, Rules)$
  $Q_\mu := Q_\mu + Q$
fin pour
$Q_\mu := \frac{Q_\mu}{n}$
Renvoyer Q_μ

Algorithme 2: *Fonction* BRUIT(S, n, ρ_d, ρ_e)

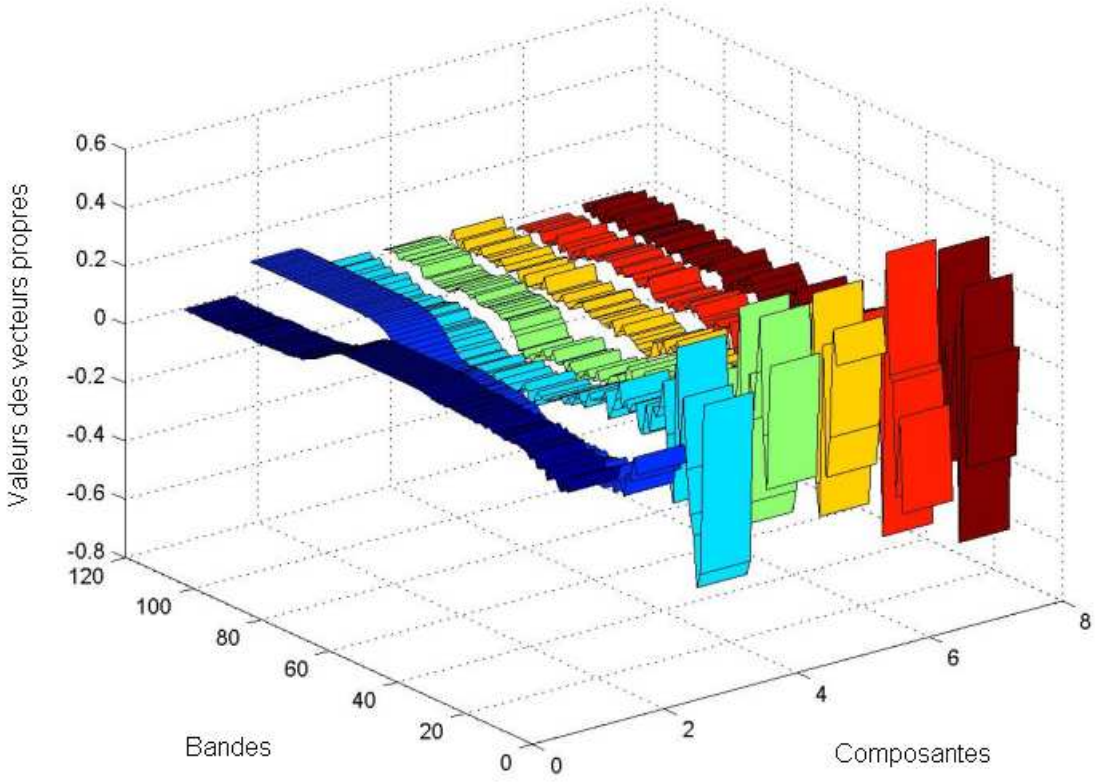


FIG. 3.14 – Représentation de la transformation par ACP pour les sept premières composantes.

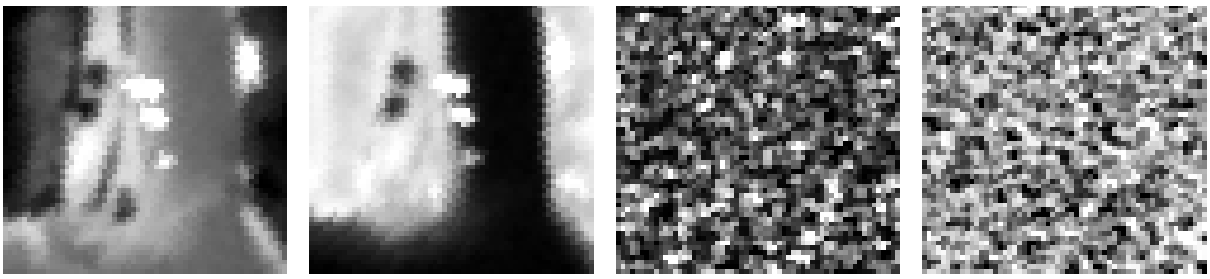


FIG. 3.15 – Les quatre premières composantes de l'ACP sur l'image de ROSIS.

avec 50 individus et 50 générations. La validation consiste à appliquer les règles sur une nouvelle base d'exemples et à calculer la corrélation entre les classes obtenues par les règles et celles attendues par l'expert.

Le graphique de la figure 3.16 présente le tracé des deux courbes suivantes :

1. $C_{data} = \{Q_{d_1}, Q_{d_2}, \ldots, Q_{d_m}\}$ tel que $Q_{d_i} = \text{BRUIT}(S, n, d_i, 0)$
2. $C_{expert} = \{Q_{e_1}, Q_{e_2}, \ldots, Q_{e_m}\}$ tel que $Q_{e_i} = \text{BRUIT}(S, n, 0, e_i)$

où le jeu de référence S a été fixé à 10% de l'ensemble de l'image, le nombre de tests n à 10 et d_m

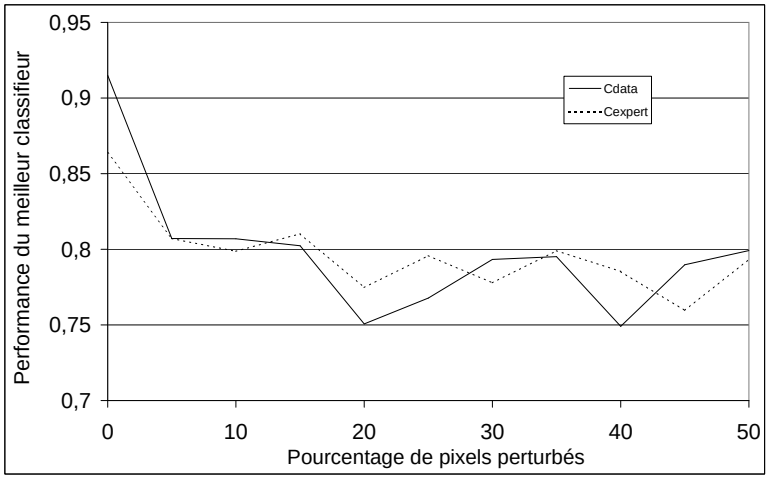


FIG. 3.16 – Décroissance de la qualité en fonction du bruit.

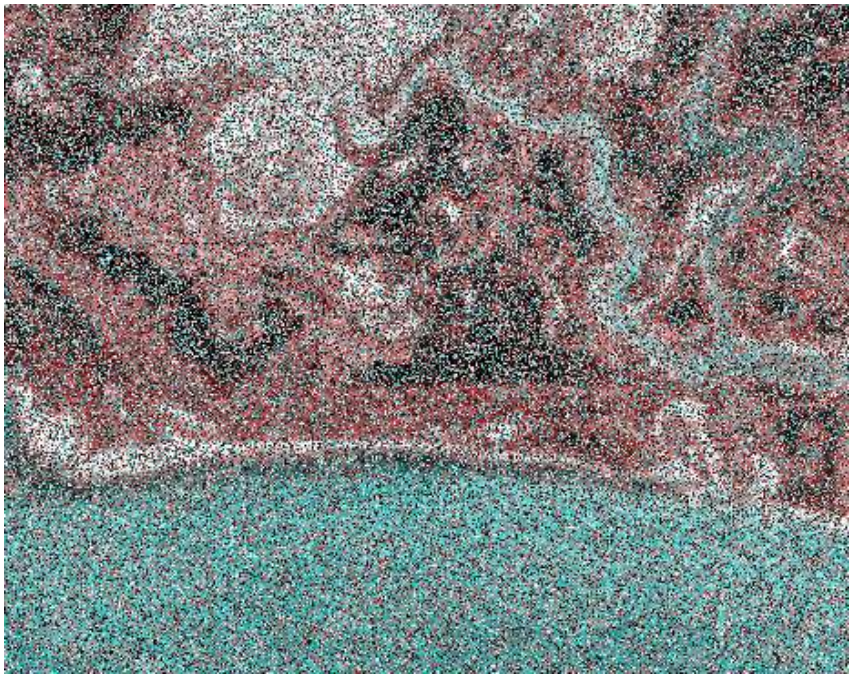


FIG. 3.17 – Extrait d'une image hyperspectrale artificiellement bruitée à 50%.

$= e_m = 50\%$.

On note que la qualité des règles apprises décroît de 0.9 à 0.8 entre 0 et 5% de bruit injecté dans les données brutes ou les données expertes. Ensuite, elle se stabilise entre 0.75 et 0.8. L'écart-type des performances observées est de $\sigma_{C_{data}} = 0.042$ et $\sigma_{C_{expert}} = 0.026$. À titre d'exemple, la figure 3.17 présente un extrait de ROSIS artificiellement bruité à 50% (par mélange de pixels pour ne pas modifier les valeurs

extrêmes et ne pas perturber la phase d'initialisation, qui n'est pas testée ici).

Nous avons testé les algorithmes évolutionnaires qui seront présentés dans le chapitre suivant par cette procédure (en remplaçant la fonction LEARNING dans l'algorithme A2 par ces algorithmes) et nous avons obtenu des performances similaires. On peut en conclure qu'une méthode évolutive est capable de résister à un coefficient relativement élevé de bruit, même si la qualité se dégrade déjà à partir de 5%. Ce type de constat n'est pas forcément surprenant : l'ajout de bruit, lorsqu'il n'atteint pas un seuil critique, permet à certaines méthodes de généraliser [Singh, 1998]. L'explication est la suivante : parmi les signaux présentés à l'algorithme, la plus grande partie est cohérente, tandis que l'autre apparaît complètement décorrélée. Quelle que soit la performance obtenue par les règles, la meilleure note sera toujours attribuée à la règle ayant capturé le maximum d'informations cohérentes, car nous avons rendu son expressivité, et donc la quantité d'informations qu'elle pouvait mémoriser, très restreinte.

À titre de synthèse, nous devons remarquer que le bruit présenté ici n'est pas un bruit naturel. Dans les situations réelles, le bruit est fortement corrélé au type d'objet soumis au capteur, ce qui peut représenter une difficulté supplémentaire. Notamment il est possible de concevoir des filtres qui élimineraient un bruit aléatoire linéaire ou gaussien comme celui présenté dans cette étude et qui seraient inefficaces avec un bruit réel. La modélisation d'un tel bruit étant relativement complexe, les tests les plus fiables se réalisent directement avec les données réelles. Pour nos expérimentations, la base de données d'images dont nous disposions possédait de nombreux défauts. Notamment, des différences de contraste ou de luminosité ont pu être observées après recalage, en plus des imperfections atmosphériques. Pour éviter de perturber nos algorithmes, nous avons travaillé au cas par cas, en collaboration avec les experts de terrain, pour éliminer, découper ou améliorer la qualité des bandes par des méthodes statistiques disponibles dans certains logiciels commerciaux.

3.6 Conclusion

Ce chapitre a présenté quelques échantillons de données brutes et de données expertes avec les principales caractéristiques qui leurs sont associées. Nous avons vu que ces données étaient très volumineuses et complexes, et nous avons donné la liste des difficultés qui peuvent perturber un algorithme de classification en conjonction avec les principaux pré-traitements disponibles pour les contenir. Notons que certains pré-traitements peuvent induire une perte d'information dans les données. Des techniques récentes, utilisant des données multi-sources et multi-expertes permettent parfois de contourner ces difficultés mais introduisent en même temps de nouvelles approximations. Dans certains cas, nous l'avons montré lors de l'expérimentation précédente ou des expérimentations effectuées sur du bruit réel (certaines seront détaillées dans le chapitre 7), un bruit aléatoire ne perturbe pas, voire améliore les capacités d'apprentissage d'un algorithme évolutif : leur principal point fort est leur tolérance au bruit (robustesse). En effet, nous avons présenté deux études, l'une concernant l'ACP, l'autre décrivant un protocole de validation permettant d'évaluer le degré de robustesse des algorithmes, dont nous parlerons dans ce mémoire, au bruit. Nous en avons conclu que, malgré les difficultés rencontrées avec l'ACP sur une image de ROSIS, il était possible d'envisager l'utilisation de méthodes génétiques sur des données bruitées. Le chapitre suivant détaille notre approche ainsi que les principes que nous avons développés pour la mise en place de notre architecture d'apprentissage.

Chapitre 4

Concepts de base des classifieurs

4.1 Approche proposée

Après de nombreuses expérimentations, nous avons considéré l'utilisation des méthodes évolutives comme étant acquise, tant pour leur capacité à résoudre les problèmes posés par la complexité des données que pour proposer un modèle robuste. Nous détaillons dans ce chapitre les principes de notre processus d'apprentissage. Nous verrons différentes représentations pour les règles et les façons de les appliquer sur les données brutes pour obtenir les classifications. En effet, de telles données nécessitent l'étude approfondie des représentations des règles afin de fournir à la fois une base de connaissances fiable mais aussi utile à l'expert, c'est-à-dire respectant les nombreux critères définis dans la section 1.2 (page 5), comme la compréhensibilité, la robustesse face aux données ou la cohérence, pour n'en citer que quelques-uns. Une fois la représentation choisie, et seulement à ce moment-là, des algorithmes adaptés pourront être construits.

Notre approche se distingue particulièrement des autres méthodes d'analyse mises en œuvre dans le domaine de la télédétection. Les méthodes traditionnelles de classification déterminent, à partir d'une image satellite brute, des associations entre l'ensemble des pixels de l'image et certaines classes thématiques grâce à une analyse supervisée ou non. Cependant, le processus doit être réitéré lorsque l'image est modifiée ne serait-ce qu'un peu, c'est-à-dire lorsque l'on change de zone géographique, de position (rotation, ...), voire de résolution (images multi-échelles).

À la différence de ces méthodes, nous voudrions disposer de ce que l'on nomme des *règles de classification*. Le système doit produire de lui-même une certaine forme d'information permettant d'induire, à partir des mêmes pixels que dans le cas des procédés classiques, les classes thématiques des zones couvertes. Cette forme d'information doit être, en plus, le plus possible *réutilisable* sur une partie non traitée de l'image ou sur toute autre donnée se rapprochant de l'image analysée, sans devoir recommencer à chaque fois le long processus d'apprentissage. Ces règles devront, après la phase d'apprentissage, *s'évaluer rapidement* (on appelle *évaluation* l'application d'une règle sur les pixels de l'image afin d'en déterminer les classes respectives) et tenir compte de toute l'information spectrale que peut contenir un pixel. Enfin, elles devront être *maximalement spécifiques*, c'est-à-dire qu'elles couvriront le maximum de pixels de la classe correspondante, en évitant au maximum les pixels d'une autre classe.

La phase d'apprentissage ne doit pas, d'une part, être liée au type de la zone étudiée (par exemple urbaine ou rurale), mais se conformer et s'adapter aux données soumises. D'autre part, elle doit produire des règles de classification dans un temps raisonnable, mais sans nuire d'aucune façon à leur qualité. Ces deux derniers critères peuvent sembler de prime abord contradictoires, mais nous verrons plus loin les nombreuses méthodes de notation et de validation qui ont été mises en place pour rendre compte de la performance des règles.

Enfin, les règles produites devraient être *simples*, c'est-à-dire que le nombre de contraintes qu'elles imposent afin de discriminer les pixels doit être très réduit. Les principales qualités des règles simples sont la vitesse d'évaluation en exploitation, leur compréhensibilité permettant la découverte ou la mise à

jour de la connaissance experte, et leur aptitude potentielle pour la généralisation. En effet, le principal intérêt de ces règles de classification doit être leur réutilisation dans un domaine ou la fréquence d'analyse des images est intensive et routinière (par exemple, pour la gestion des flux vidéo). Cependant, nul ne garantit que les structures de données étudiées ici (qui d'une certaine manière présagent de la qualité des résultats) permettront une réutilisation totale : cela peut en effet dépendre des conditions de prise de vue (date, saison, conditions d'illumination, ...), ou de divers autres facteurs.

Dès lors, il s'agit de répondre à deux questions essentielles :

1. Quelles sont les meilleures représentations pour les règles de classification ?
2. Quels sont les meilleurs algorithmes pour l'apprentissage, la manipulation et éventuellement le raffinement de ces règles de classification ?

La réponse à la première question est l'objet de ce chapitre, tandis que l'autre sera développée dans le chapitre suivant.

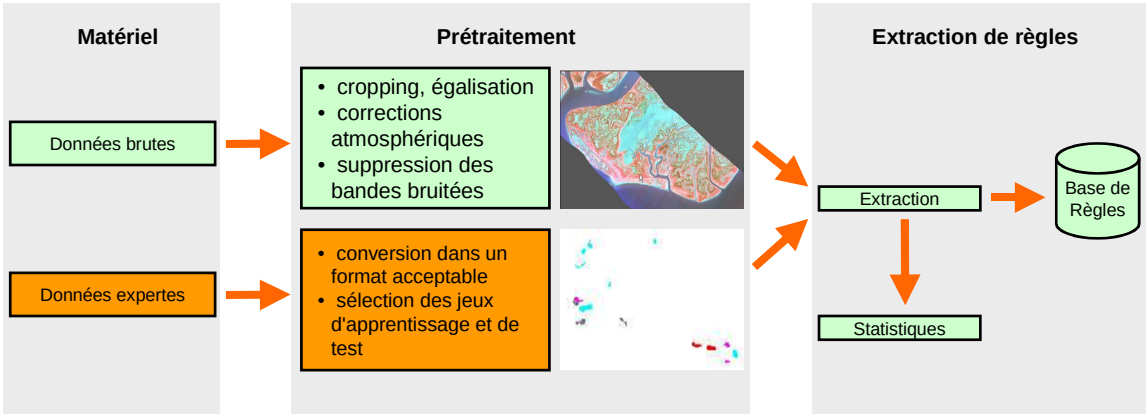


FIG. 4.1 – Schéma général du processus d'apprentissage.

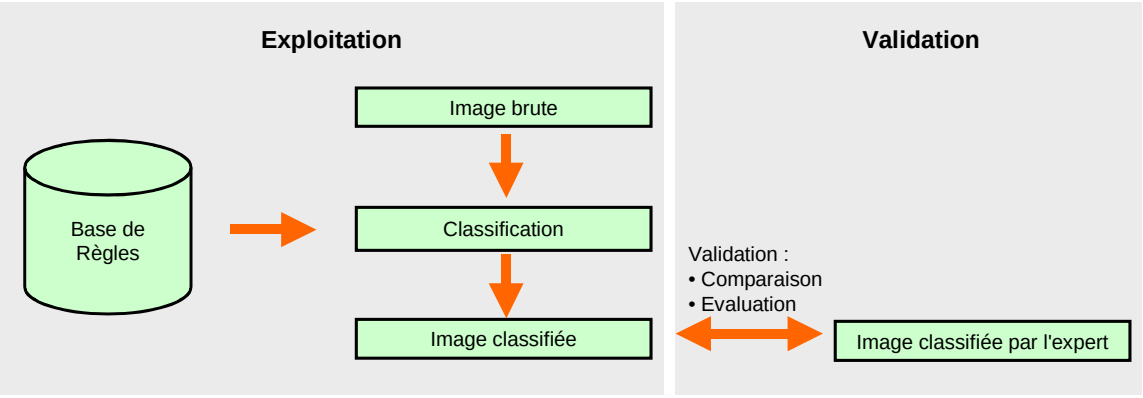


FIG. 4.2 – Schéma général du processus de validation.

La figure 4.1 présente le schéma général de notre processus d'apprentissage. Après un ensemble de pré-traitements, les données brutes et expertes sont analysées par un programme d'apprentissage. Son

rôle est d'extraire un certain nombre de règles modélisant une fonction qui permet, à l'aide d'un ensemble de descripteurs caractéristiques d'une donnée brute (valeurs spectrales des pixels, ...) de retrouver des informations sur la classe de cette donnée (qualitatives, quantitatives, statistiques). Ces règles sont stockées dans une base de règles et y attendent la phase d'exploitation. Cette phase, présentée dans la figure 4.2, se compose d'une partie classification et d'une partie validation, permettant de dégager un certain nombre de statistiques, notamment des mesures de qualité sur les algorithmes employés ou les résultats produits.

La section suivante présente quelques rappels sur les différents types de classification. La section 4.3 présente les termes que nous avons définis pour décrire le formalisme de nos règles, exposé dans la section 4.4. La section 4.5 présente l'algorithme général que nous allons employer, suivi d'une présentation des mesures de qualité utilisées avant ou après l'apprentissage (section 4.6).

4.2 Rappels sur les différents types de classification

Avant d'aborder les sections suivantes, nous devons prendre en compte le fait qu'il existe plusieurs types de problèmes de classification [Sousa et al., 2003; Huguenin et al., 1997]. L'existence de différents types dicte à la fois les principes du processus d'apprentissage (nous serons confrontés à plusieurs types d'expertises) et ceux de la représentation des règles (les règles prendront plusieurs types de décisions).

- *La classification dure*[1] (ang., *hard classification*). Les différentes classes forment une partition pure de l'image. Pour chaque pixel p, sa classification $f_{c_i}(p)$ (selon l'algorithme) pour la classe experte c_i est soit 0, soit 1, sans recouvrement possible :

$$f_{c_i}(p) = \{0,1\} \text{ pour } i = 0,\ldots,n \text{ avec } \sum f_{c_i}(p) = 1 \tag{4.1}$$

  où n est le nombre de classes.
  Elle est aussi appelée classification *one-by-one* car chaque pixel est affecté à une et une seule classe.

- *La classification douce*[2] (ang., *soft classification*). Les différentes classes forment des ensembles qui peuvent se recouvrir ou laisser des pixels non classés :

$$f_{c_i}(p) = \{0,1\} \text{ pour } i = 0,\ldots,n \tag{4.2}$$

- *La classification floue* (ang., *fuzzy*, *subpixel* ou *one-by-N classification*). Un vecteur de coefficients est affecté à chaque pixel indiquant, pour chacune des classes, la composition supposée de cette classe pour ce pixel :

$$f_{c_i}(p) \in \mathbb{R} \text{ pour } i = 0,\ldots,n \quad \text{avec } \sum f_{c_i}(p) = 1 \tag{4.3}$$

  Les méthodes pratiquant la *démixtion spectrale* (ang., *unmixing*) sont typiquement des classifieurs flous. Une autre interprétation existe pour la classification floue, appelée *classification probabiliste*. Dans ce cas, la valeur $f_{c_i}(p)$ représente la probabilité qu'un pixel soit de la classe c_i et peut permettre en apprentissage supervisé d'orienter la suite de la classification.

- *La classification à intervalles flous* (ang., *fuzzy-intervals classification*). Nous avons développé un quatrième type de classification, car il répond à des besoins spécifiques pour les experts. Un intervalle réel est associé à chaque pixel p indiquant l'estimation de la composition d'une classe donnée pour ce pixel :

$$f_{c_i}(p) \in [a_{c_i}, b_{c_i}] \text{ pour } i = 0,\ldots,n \quad \text{avec } a_{c_i}, b_{c_i} \in \mathbb{R} \tag{4.4}$$

[1] Nous utiliserons dans la suite le terme *classification hard*.
[2] Nous utiliserons dans la suite le terme *classification soft*.

Ce type de classification généralise les types précédents. En effet, elle correspond à une classification *soft* si $a_{c_i} = b_{c_i} = \{0,1\}$, et à une classification floue si $a_{c_i} = b_{c_i}$. Contrairement aux représentations précédentes, celle-ci permet de représenter de nouveaux types d'expertise, comme par exemple « les proportions respectives des classes A, B et C dans le pixel p ne sont pas connues, mais il n'y a pas la classe D ». Une classification *soft* ou floue doit préciser la proportion exacte pour chaque classe, alors que la classification par intervalles flous permet à l'expert d'indiquer une imprécision totale (intervalle [0 ;1]) ou partielle (intervalle [0 ;0.5], par exemple) pour une classe. Certaines expertises, qui doivent être rejetées dans le cadre des autres représentations, peuvent ici être utilisées.

Le tableau 4.1 présente quelques exemples de classifications pour un pixel donné et trois classes (A, B et C).

Classification	**Exemple**	**Description**
Hard	A = 100%, B = 0%, C = 0%	De la classe A
Soft	A = 100%, B = 0%, C = 100%	Disjonction d'expertises
Floue	A = 23%, B = 12%, C = 65%	Conjonction d'expertises
À intervalles	A ∈ [0 ;100], B ∈ [50 ;100], C ∈ [0 ;0]	La proportion de la classe A est inconnue, celle de la classe B est d'au moins 50% et la classe C est indéniablement absente

TAB. 4.1 – Différents types de classification.

La nature du terrain réel étant différente de celle observée sur les images de télédétection, nous avons observé que l'amélioration de la représentation de l'information utilisée en expertise (c'est-à-dire passer d'une représentation *hard* à une représentation floue, par exemple), qui se rapprocherait d'une représentation réaliste, peut conduire à l'amélioration des performances.

4.3 Entités manipulées par les règles

Cette section permet d'aborder la définition des termes que nous utiliserons par la suite pour décrire de manière générique une technique de représentation des règles, et les représentations que nous avons choisies, en particulier. Beaucoup de notions exposées ici sont connues en statistique, mais nous soulignerons les particularités liées à la classification d'images.

Nous devons commencer par définir formellement les entités que nous manipulerons. Comme nous l'avions souligné dans la section 3.2, les images sont définies dans un système comportant trois dimensions, la largeur (X), la hauteur (Y) et la dimension spectrale (S). Il en existe de nombreuses autres (par exemple, les dimensions temporelles, stéréoscopiques, etc) que nous n'allons pas aborder dans cette section. Nous définissons un *1-échantillon* (nommé simplement *échantillon* dans la suite) comme une donnée de base, indivisible et représenté par des types simples dans une ou plusieurs des dimensions de l'espace des données (booléens, entiers, réels ou vecteurs contenant des valeurs de type pré-cité). Il est indivisible dans le sens où l'échantillon ne peut pas être exprimé dans un système de dimensions différent ou plus simple, en tout cas pas de manière plus naturelle. Par exemple, un pixel de données brutes représentant un spectre de radiance est naturellement représenté comme l'ensemble des valeurs de radiances (réelles) le long de la dimension spectrale S :

$$<pixel> \equiv [s_1, s_2, s_3, \ldots, s_i, \ldots, s_n] \tag{4.5}$$

où n est le nombre total de bandes dans l'image, et s_i appartient à un corps spécifique pour S, par exemple $\mathbb{R}$.

Ainsi, pour créer un échantillon représentant un pixel donné, nous avons *agrégé* la dimension spectrale S et nous utilisons des valeurs distinctes exprimées dans toutes les autres dimensions (X,Y) pour créer d'autres échantillons. Dans notre modèle, S est dite *dimension d'agrégation* et X et Y sont appelées des *dimensions de parcours*. Nous pourrions tout aussi bien créer un échantillon de données en considérant la

première ligne de chaque image comme une donnée atomique, et Y et S comme des dimensions de parcours. Cependant, ce découpage n'est pas naturel et ne correspond pas à un usage adapté à la télédétection. Nous définissons donc chaque échantillon comme étant l'ensemble de toutes les valeurs exprimées dans les dimensions d'agrégation. Cette considération est surtout utile pour traiter des données multi-sources ou multi-expertes : l'ajout d'une image supplémentaire est alors simplement vu comme l'ajout d'une dimension d'agrégation si l'on souhaite traiter les données en parallèle, ou d'une dimension de parcours si l'on souhaite traiter les données en séquentiel. Il existe deux types d'échantillons : les *échantillons bruts* et les *échantillons experts*. La réunion d'un échantillon brut avec un échantillon expert est appelé *exemple d'apprentissage* ou *exemple de validation*. L'ensemble des échantillons est une partition de l'espace complet des données. Nous rappelons que, comme dans la terminologie des systèmes de classifieurs, ce qui est externe au système d'apprentissage est appelé *environnement*.

4.4 Formalisme des règles

Nous pouvons maintenant définir la notion de *règle*. Elle représente l'unité la plus élémentaire de la connaissance traitée par un algorithme à base de classifieurs. Typiquement, une règle prend en entrée un échantillon brut et renvoie une valeur booléenne (règle de décision), entière (règle de classification), réelle (règle de *démixtion spectrale*) ou de n'importe quel autre type. Elle doit attribuer à l'échantillon en question une *description* claire et compréhensible pour n'importe quel expert du domaine. Une règle ou un ensemble de règles représentent à eux seuls la connaissance apprise durant et à la fin de l'apprentissage, et sont indépendantes en exploitation et en validation. L'indépendance signifie que les paramètres contenus dans une règle doivent conduire un algorithme générique (ou un être humain) à pouvoir traiter n'importe quel nouvel échantillon brut, dans une phase que l'on qualifiera d'*interprétation automatique* ou d'*application*. L'expertise automatique produite est alors facile à comparer à l'échantillon expert correspondant afin de valider la règle. On peut ensuite éventuellement associer à la règle des paramètres externes (qualité, erreur, statistiques diverses) permettant de la caractériser de manière générale pour l'interprétation suivante. Il est à noter une différence importante entre une règle de classification et les classifieurs de Holland [Holland, 1986] : pour pouvoir garantir l'opérabilité des règles sur de nouveaux échantillons, elles devront être indépendantes des exemples, notamment de leur ordre de passage, c'est-à-dire concrètement ne pas altérer l'environnement du système ni conduire à la modification de registres ou de mémoire internes suite à leur application. Attacher un *contexte* à ces règles permettrait de donner au système un comportement *anticipatif*, l'autorisant à modifier ou affiner sa réponse en fonction de l'évaluation de plusieurs échantillons précédents. Malheureusement, la classification d'images se prête mal à la transformation de l'environnement en une machine à états : les capacités de généralisation de l'algorithme en seraient amoindries. De toute façon, la plupart des informations importantes, y compris *contextuelles* (échantillons voisins), peuvent être intégrées sous la forme d'attributs supplémentaires pour l'échantillon courant.

En reprenant la terminologie des systèmes de classifieurs, le formalisme général d'une règle de classification est le suivant :

$$<règle> \equiv <condition> \rightarrow <action> \tag{4.6}$$

Dans notre cas, ce formalisme correspond à la règle suivante :

$$\textbf{si } <pixel> \textbf{ satisfait } <condition> \textbf{ alors } <classe\ thématique> \tag{4.7}$$

Plus précisément, cela signifie que si un échantillon brut satisfait la condition donnée, il devra être considéré comme appartenant à la classe spécifiée par la règle.

On remarque immédiatement que la partie $<condition>$ joue un rôle extrêmement important. Elle seule conditionne réellement la performance de la règle, pour deux raisons évidentes :

1. Si le formalisme de représentation de la condition est trop restrictif, il y a un risque de produire des règles trop spécifiques, voire incohérentes. Destinées à être générées automatiquement par l'algo-

rithme, il y aura peu de chance que toutes les contraintes présentes dans la condition soient *activées* en même temps pour que le pixel satisfasse la règle.

2. Si à l'inverse la syntaxe est beaucoup trop générique, un allongement du temps de recherche et des problèmes de recouvrements apparaîtront : de nombreuses règles seront actives pour le même pixel et la performance de l'algorithme risque d'être dégradée.

La partie <*condition*> de la règle fait appel à un certain nombre de *variables*, prenant leurs valeurs directement dans les échantillons bruts, et d'*opérateurs* calculant de nouvelles valeurs avec ces variables qui sont soit considérées comme le résultat final de la condition, soit un résultat intermédiaire à destination d'autres opérateurs. Nous proposons que ces opérateurs et variables soient regroupés au sein de différentes structures. Nous en distinguons plusieurs.

Les *structures plates* sont des fonctions classiques n'utilisant qu'une seule variable par partie <*condition*>, par exemple :

$$<condition> \equiv <variable> <opérateur> <constante> \tag{4.8}$$

Les *structures arborescentes* ou *arbres de calcul* permettent de représenter des équations, voire des programmes complets, à l'image de la programmation génétique. Chaque nœud correspond à un opérateur et chaque feuille à une variable ou un opérateur d'arité nulle[3] (constante). De fait, un arbre de calcul se décompose en trois niveaux :

- Le sommet. C'est un opérateur évalué en dernier dans l'arbre qui renvoie le résultat correspondant au type de la règle (un indicateur de la classe de l'objet pour une classification, un pourcentage indiquant la proportion d'une classe donnée dans un pixel pour une règle de *démixtion spectrale*, ...). Le type de cet opérateur est déterminé par le problème à résoudre. Un type booléen sera typique de la résolution d'un problème de classification *soft*, tandis qu'un type réel sera typique d'un problème de classification floue.
- La partie interne. L'équation ou le calcul proprement dit. Chaque nœud est un opérateur prenant en paramètre un ou plusieurs arguments (entiers, réels, ...) et renvoyant un argument à destination de l'opérateur suivant.
- Les feuilles. Ce sont des opérateurs qui prennent directement leurs arguments parmi les valeurs des échantillons. Ces valeurs peuvent être de différents types selon qu'elles doivent caractériser des valeurs de radiance, des altitudes, des températures, ... Ces opérateurs sont spécifiques à l'application cible et doivent être créés en conséquence (fonctions renvoyant un spectre, indicateur de la texture dans le voisinage du pixel courant, date de la prise de vue, corrélation entre le spectre d'un pixel et un spectre de référence, ...).

Nous pouvons répartir les opérateurs correspondant aux nœuds des arbres en deux catégories, en fonction de leur niveau d'interaction avec les données : les opérateurs *context-free* et les opérateurs *context-dependent*. Les opérateurs *context-free* ne sont pas liés à une application particulière (opérateurs arithmétiques, booléens ou algorithmiques). Un algorithme capable de manipuler des arbres ne contenant que des opérateurs *context-free* est adaptable à n'importe quel jeu de données. Les opérateurs *context-dependent bruts* sont liés à l'interprétation des échantillons bruts (GetValue(), GetSpectrum(), ComputeTexture(), ...). Ils peuvent faire appel à des algorithmes plus ou moins évolués (segmentations basiques, reconstruction géodésique, ...) afin de transformer un échantillon en valeur exploitable. Enfin, les opérateurs *context-dependent experts* ne travaillent qu'avec les données expertes. Ils permettent de découvrir certaines propositions sans utiliser la moindre donnée brute. Par exemple, concernant une image, la proposition suivante « les arbres sont souvent ronds » peut être exprimée par la règle :

$$\textbf{si } \mathit{Circularité}\ (\ \mathit{Région\text{-}Connexe}\ (p)) > s \ \textbf{alors}\ \ \mathit{Classe}\ (p) = \mathit{Arbre} \tag{4.9}$$

où p est un pixel et s un seuil représenté par une constante réelle. Cette proposition peut être découverte par un algorithme d'apprentissage uniquement à partir d'une image de classification experte, l'objectif de l'algorithme étant d'affecter une valeur suffisamment élevée pour le seuil de la circularité s.

[3] L'arité est le nombre de paramètres d'une fonction. L'arité de "+" est de deux, l'arité de π est de 0.

Cependant, la découverte de ce type de règles, que l'on qualifiera plutôt de *prédicats*, dépasse le cadre de la classification d'images et ne sera pas abordé ici.

Enfin, des structures plus complexes peuvent être envisagées (réseaux de neurones, génomes, ...). Néanmoins, il faut garder à l'esprit que certains critères décrits dans la section 1.2 (page 5) dépendent de la bonne structuration des règles, comme par exemple la compréhensibilité ou l'indépendance algorithmique. Chaque structure facilite ou non l'utilisation des règles comme individus d'un algorithme évolutionnaire (création, croisement, ...). La capacité de découverte des opérateurs génétiques, que nous décrivons en détail dans l'annexe C, est donc clairement liée à l'expressivité mais aussi à la simplicité de la structure retenue pour les règles.

4.5 Découverte des règles

4.5.1 Un processus générique

Le formalisme général des règles présenté dans la section précédente permettant de décrire les connaissances du domaine est adapté au problème de la classification d'images. Les règles sont expressives et simples pour l'expert, en revanche, elles imposent à l'algorithme d'apprentissage une complexité assez élevée lors de la découverte automatique de la kyrielle de paramètres ou de valeurs diverses qui les composent. Par exemple, si nous choisissons une représentation arborescente pour discriminer un spectre de référence parmi tous les autres possibles, il faut une valeur de confiance – ou plutôt un intervalle de confiance – pour chacune des longueurs d'onde de ce spectre, ne sachant pas à l'avance (dans le cas général) quelle est la liste des longueurs d'onde permettant de discriminer deux classes. Dans le cas simple de SPOT (3 bandes), cela revient à demander à l'algorithme de trouver un point dans un espace à 6 dimensions. Ce nombre de dimensions explose rapidement dans le cas de l'hyperspectral ou si la représentation est un tant soit peu un peu plus évoluée.

On le voit donc, un algorithme déterministe de découverte de ces règles est tout simplement proscrit. De la même façon, un parcours exhaustif est de toute évidence impossible. Nous avons donc choisi l'algorithmique génétique, connue pour sa robustesse à traiter les problèmes d'optimisation et pour ses qualités puissantes d'exploration et de recherche [Eiben et Smith, 2003]. Ce modèle d'apprentissage stochastique a déjà fait ses preuves pour les problèmes NP-complets [Ruttkay et al., 1995], dont la découverte de règles de classification est un exemple. Avant de présenter le schéma général de notre algorithme, plusieurs notions fondamentales, s'articulant autour de l'*individu*, de la *fonction d'évaluation* et de la *génération de la population suivante* doivent être définies.

Ainsi, l'algorithme évolutionnaire introduit, comme dans le modèle de Darwin, une notion d'*individu* correspondant à une structure atomique qui doit porter toute l'information (l'ensemble est nommé *génotype*) permettant de caractériser son comportement, ses manifestations externes (nommées *phénotype*) dans un environnement donné [Goldberg, 1991]. Dans notre cas, l'individu en question est incarné par une règle permettant, grâce à tous les paramètres caractéristiques d'un échantillon de données (par exemple, les valeurs de radiance d'un pixel), de déterminer la classe thématique des pixels de l'image. Un individu représente à la fois le moyen de mémoriser les contraintes à optimiser au cours de l'apprentissage, et la solution au problème. Ces informations sont stockées conjointement en un ensemble de *gènes* appelé *chromosome* (structure contenant l'information utile) [Michalewicz, 1996].

Afin d'avoir une exploration équiprobable de l'espace, il nous faut une armée, une *population diversifiée* de tels individus. Pour détecter dans cette population les individus les mieux adaptés au problème posé, ou à l'inverse éliminer les moins bons, l'algorithme génétique utilise une fonction de qualité (*fitness*), dont le rôle est d'affecter à chaque individu une note en fonction de sa capacité à répondre au but attendu relativement aux autres individus [Eiben et Schoenauer, 2002] : dans notre cas, les meilleurs individus sont tout simplement ceux qui peuvent corréler au mieux la réponse de la règle avec le résultat attendu par l'expert.

Enfin, une stratégie doit être employée pour manipuler directement cette population : au fil des *générations* de l'algorithme certains individus sont conservés, d'autres sont éliminés, et d'autres encore donnent naissance à de nouveaux individus plus adaptés. Une génération couvre un cycle de l'algorithme,

c'est-à-dire l'évolution d'une population complète. Ce processus permet d'explorer la diversité des comportements possibles et d'en découvrir de nouveaux encore plus performants que ceux des parents. On distingue trois opérateurs génétiques importants concernant le passage d'une génération à l'autre : les *opérateurs de sélection*, sélectionnant les individus pour les opérateurs de variation ou pour le remplacement, les *opérateurs de variation*, permettant d'obtenir un (croisement) ou N (mutation) nouveaux individus à partir d'une sélection (avec remise) de parents et l'*opérateur de remplacement* qui sélectionne les individus (sans remise) pour la génération suivante [Goldberg et Deb, 1991].

ALGORITHME A3
PROCESSUS DE DÉCOUVERTE DES RÈGLES

$\rightsquigarrow$ R est une règle de classification et Γ_i ($i = 0, \ldots, n$) est une population de règles

$\rightsquigarrow$ Création d'une règle en fonction des exemples
R := RÈGLEINITIALE*(exemples)*
$\rightsquigarrow$ Initialisation d'un pool aléatoire depuis R
Γ_0 := INITIALISATION*(R)*
i := *0*
répéter
  $\rightsquigarrow$ Calcul de la *fitness* pour chaque règle
  ÉVALUATION*(Γ_i)*
  Γ_{i+1} := CROISEMENT*(Γ_i, p_χ)* $\cup$ COPIE*(Γ_i, $1 - p_\chi$)*
  Γ_{i+1} := MUTATION*(Γ_{i+1}, p_μ)*
  $\rightsquigarrow$ Nouvelle génération de règles
  Γ_{i+1} := REMPLACEMENT*(Γ_i,Γ_{i+1})*
  $i := i + 1$
jusqu'à ce que CRITÈREDETERMINAISON*(Γ_i, i)* = *vrai*
$\rightsquigarrow$ Sélection de la meilleure règle
R := MEILLEUR*(Γ_i)*
Résultat : R, la règle de classification pour une classe donnée

Algorithme 3: *Fonction* DÉCOUVERTE

Ces notions fondamentales sont la clef de voûte de notre algorithme évolutionnaire dont le déroulement général est présenté dans l'algorithme A3.

Notre système de classifieurs est en fait un ensemble de classifieurs indépendant, répondant chacun à une classe précise (de manière qualitative ou quantitative), et évoluant de manière distincte. Le résultat final est donc une population de règles comprenant soit une, soit plusieurs règles décrivant chaque classe donnée. Il existe plusieurs raisons pour le choix de l'indépendance des règles, plutôt qu'un modèle de type *compétitif* : les règles les plus efficaces dans la génération courante ne doivent pas perturber les autres règles qui mettraient, au cours des générations, plus de temps pour s'améliorer. Dans un modèle compétitif, la fonction de qualité évalue simultanément tous les individus et les plus mauvais risquent d'être éliminés prématurément. L'indépendance autorise l'élimination d'un ou plusieurs classifieurs au sein de la population de règles, sans que la variation de qualité souffre d'un rapport trop étroit aux autres classifieurs, ni même devoir attendre une convergence de leur part vers le même niveau de performance. Enfin, elle garantit l'impartialité des mesures de qualité obtenues pour une classe thématique donnée.

4.5.2 Un algorithme efficace

Au sens le plus large, l'objectif de toute méthode ou tout algorithme génétique est d'améliorer la compétitivité des individus. Cependant, il est bon de noter dès à présent, le fait que cet objectif

peut donner lieu à des interprétations bien distinctes. Plusieurs critères d'optimisation sont possibles. Doit-on optimiser la qualité des solutions ou le temps d'apprentissage (contradiction souvent observée en algorithmique génétique) ? Il est par exemple connu qu'il est préférable d'avoir un grand nombre de générations pour un faible nombre d'individus que l'inverse, pour le même temps d'apprentissage [Goldberg et al., 1992; Ceroni et al., 2001]. Malheureusement, avec un faible nombre d'individus, les recombinaisons entre parents sont moins nombreuses, ce qui empêche l'exploration de configurations structurelles trop complexes dans le cas d'une représentation arborescente. Nous avons dégagé une liste de facteurs pouvant réduire la complexité de notre système :

- Les facteurs dépendants de l'expert : celui-ci peut inclure, outre les échantillons, différentes expertises à l'intention de l'algorithme. La taille estimée des règles de classification peut être introduite comme pression de sélection (lors du calcul de la *fitness* ou dans les opérateurs de sélection). Le choix des opérateurs intervenant dans les règles peut limiter l'espace de recherche (indices à utiliser pour trouver une classe donnée, ...). L'élagage des échantillons bruts (sélection de bandes) et experts (sélection des points d'apprentissage, réduction du nombre d'échantillons à utiliser en fonction des classes, ...) limite lui aussi l'espace de recherche. Enfin, une interactivité durant l'apprentissage est possible par la visualisation des taux d'apprentissage, la mise en place de conditions d'arrêt sur les taux d'erreur, la possibilité de modifier les paramètres en cours d'apprentissage, ou de geler les modifications de certaines règles directement ou en fonction d'un paramètre de confiance.
- Le principe de fonctionnement de l'algorithme lui-même, notamment des opérateurs génétiques. L'opérateur d'initialisation est optimisé par rapport aux données (plusieurs exemples seront présentés dans le chapitre suivant) et le critère de terminaison est calculé à partir du nombre de générations et de la convergence de la fonction de qualité des règles. De plus, le taux de mutation est modifié en cours d'apprentissage. Enfin, l'utilisation du *niching* dans l'AG permet de maintenir une certaine diversité dans la population mais aussi de découper cette population en blocs d'individus selon leur contribution au problème à résoudre.
- Les paramètres de l'algorithme : Wilson [Wilson, 1998] a montré que la complexité d'un problème n'est pas proportionnelle à la taille de son espace de recherche mais au nombre de classifieurs permettant d'exprimer une généralisation. Des techniques d'élagage des règles existent, et l'une d'elle va être présentée dans le chapitre 6. De nombreux paramètres influencent la diversité des règles et l'exploration de l'espace de recherche, ainsi que les conditions de convergence de l'algorithme. Certains d'entre eux seront présentés avec les algorithmes eux-mêmes.

En fait, la complexité de traitement par l'algorithme évolutionnaire dépend surtout de la structure des règles et non de l'algorithme lui-même. Nous donnerons dans la suite, lorsque cela s'avérera pertinent, la complexité en mémoire des solutions (en terme de taille des individus), ainsi que la complexité de l'apprentissage (temps et nombre de générations), en regard de chacune des représentations proposées.

4.5.3 Une architecture et des opérateurs adaptés

Bien que le modèle vu jusqu'ici soit générique et puisse s'appliquer à de nombreux problèmes, notre objectif est de construire une architecture et des opérateurs adaptés aux données de télédétection, car celles-ci présentent une complexité et un caractère bien particulier, qu'il faut prendre en compte. Le processus générique de découverte des règles présenté dans l'algorithme A3 est marqué par la spécificité du problème de classification à plusieurs niveaux :

- tout d'abord, lors de la création des règles : que ce soit de manière automatique ou par l'expert, la création de la première règle est guidée par les données. Selon la représentation choisie pour cette règle, certains paramètres sont initialisés de telle sorte que la solution réelle du problème se trouve suffisamment proche, en terme de recherche combinatoire, du premier individu. Cette technique est nommée *look-ahead creation*.
- ensuite, à travers la fonction d'évaluation qui intègre une gestion des individus spécifique au problème à résoudre, concernant notamment le nombre de nœuds ou la profondeur des arbres de calcul lorsqu'il existe une information sur la complexité supposée de ces arbres. Par exemple, la représentation d'un indice de végétation nécessite un arbre de moins de 8 nœuds.

– enfin, au niveau de la pression de sélection (opérateurs de sélection de parents ou pour le remplacement) : nous présentons dans ce chapitre une série de mesures de qualité qui permettent d'évaluer automatiquement la lisibilité des règles ou l'homogénéité de la population, en fonction de critères importants en télédétection (matrices de confusion, validation graphique, compacité des classes, structure des règles, etc).

Ce modèle nécessite l'emploi de diverses fonctions génétiques comme la fonction d'évaluation, l'opérateur d'initialisation, de croisement, de mutation, de sélection, de recyclage générationnel et de convergence. Ces opérateurs sont assez généraux et sont utilisés dans divers problèmes. Nous les avons décrits en détail dans l'annexe C. Parmi ces opérateurs, la fonction *fitness* a pour rôle d'évaluer les individus par rapport aux autres individus de la population. Pour cette évaluation, la fonction nécessite la mise au point d'un certain nombre de mesures de qualité, spécifiques ou non à notre approche. Nous les présentons dans la section suivante.

4.6 Mesures de qualité

4.6.1 Calculées pendant l'apprentissage

Dans cette section, nous introduisons les mesures que nous avons définies pour évaluer la qualité des classifieurs en cours d'apprentissage (elles sont utilisées par la fonction *fitness*, les opérateurs de sélection, ...). Certaines de ces mesures sont génériques et peuvent être utilisées dans d'autres problèmes de classification, d'autres sont spécifiques à la classification d'images de télédétection. Nous utiliserons par la suite les termes qui sont d'usage, c'est-à-dire la terminologie anglaise.

4.6.1.1 Mesures basées sur l'expertise

En classification d'images, l'évaluation de l'apprentissage est habituellement basée sur une matrice de confusion mettant en relation la classification proposée par l'expert pour chacune des classes C_i avec celle obtenue par les règles de classification. La figure 4.3 définit les variables nécessaires pour construire les mesures de qualité qui vont suivre. La matrice de confusion possède autant de lignes que de règles différentes à traiter.

		Quantité d'échantillons experts de la classe ...					Q_{ppa}	Total (k_i)
		C_1	C_2	…	…	C_n		
Quantité d'échantillons bruts classifiés par les règles	C_1	$P_{C_1}^{C_1}$	$P_{C_1}^{C_2}$	…	…	$P_{C_1}^{C_n}$	Q_{ppa}^1	$\Omega(C_1)$
	C_2	$P_{C_2}^{C_1}$	$P_{C_2}^{C_2}$				⋮	⋮
	⋮	⋮		⋱			⋮	⋮
	⋮	⋮			⋱		⋮	⋮
	C_n	$P_{C_5}^{C_1}$				$P_{C_n}^{C_n}$	Q_{ppa}^n	$\Omega(C_n)$
Q_{sens}		Q_{sens}^1	…	…	…	Q_{sens}^n		

FIG. 4.3 – Matrice de confusion permettant de calculer la *précision* et la *sensibilité* en comparant les résultats des règles avec l'expertise.

Q_{accur} est une mesure très populaire évaluant la proportion d'échantillons correctement classés, par

rapport à tous les échantillons, pour n classes :

$$Q_{accur} = \frac{\sum_{i=1}^{n} P_{C_i}^{C_i}}{\sum_{i=1}^{n} \Omega(C_i)} \tag{4.10}$$

où $\Omega(C_i)$ est le cardinal de l'ensemble C_i.

La faiblesse de cette mesure est qu'elle prend en compte seulement les échantillons correctement classés. Pour rendre les résultats de classification plus précis, trois mesures relatives ont été utilisées : la *précision* Q_{ppa} (ang., *positive predictive accuracy*), la *sensibilité* Q_{sens} (ang., *sensitivity*) et la *spécificité* Q_{spe} (ang., *specificity*).

La *précision* (Q_{ppa}) mesure, pour une classe donnée, la proportion d'échantillons correctement classés par rapport aux échantillons classés par un classifieur :

$$Q_{ppa}^{i} = \frac{P_{C_i}^{C_i}}{\sum_{k=1}^{n} P_{C_i}^{C_k}} \tag{4.11}$$

La *sensibilité* (ou taux de *vrais positifs*) Q_{sens} mesure, pour une classe donnée, la proportion d'échantillons correctement classés par rapport aux échantillons classés par l'expert :

$$Q_{sens}^{i} = \frac{P_{C_i}^{C_i}}{\sum_{k=1}^{n} P_{C_k}^{C_i}} \tag{4.12}$$

La *spécificité* (ou taux de *vrais négatifs*) Q_{spe} mesure la proportion d'échantillons correctement classés comme n'étant pas dans une classe donnée, selon l'expert :

$$Q_{spe}^{i} = \frac{\sum_{j=1,j\neq i}^{n} \sum_{k=1,k\neq i}^{n} P_{C_k}^{C_j}}{\sum_{j=1,j\neq i}^{n} \sum_{k=1}^{n} P_{C_k}^{C_j}} \tag{4.13}$$

Les opposés des deux mesures précédentes, $1 - Q_{sens}$ et $1 - Q_{spe}$ sont respectivement appelés *faux négatifs* et *faux positifs*. Lorsqu'elles sont utilisées dans la fonction d'évaluation afin de guider l'apprentissage des règles, les quatre mesures ci-dessus peuvent donner une classification non fidèle au résultat attendu par l'expert si la distribution des échantillons est telle qu'elle perturbe l'apprentissage, comme nous le verrons dans la section 5.2.1.2 du chapitre suivant. Par conséquent, nous proposons une nouvelle mesure, ajustée pour chacune des classes :

$$N_{final}^{i} = \alpha \cdot Q_{sens}^{i} + (1 - \alpha) \cdot Q_{spe}^{i} \tag{4.14}$$

Cette mesure a quelques avantages : elle est indépendante de l'ordre de présentation des échantillons, de la taille des classes et se calcule en temps linéaire quelque soit la complexité des données (nombre de classes, nombre de bandes spectrales, nombre d'échantillons). Nous proposons alors, comme mesure de qualité globale, une moyenne de chacune des mesures N_{final}^{i}, pondérée par la taille des classes :

$$N_{global} = \frac{\sum_{i=1}^{n} \Omega(C_i) \cdot N_{final}^{i}}{\sum_{i=1}^{n} \Omega(C_i)} \tag{4.15}$$

4.6.1.2 Mesures indépendantes de l'expertise

Même si la mesure experte fait foi pour départager les classifieurs, de nombreuses autres mesures ont été utilisées en télédétection, notamment en non supervisé, pour juger de la qualité des classes obtenues. À cause de leur indépendance par rapport à l'expertise, ces mesures permettent d'obtenir des classes qui présentent certaines qualités (similarité intra-classe, ...) lorsque l'expertise fait défaut ou lorsqu'il s'agit de la remettre en cause. Ces critères sont employés en classification *hard* et se basent sur une distance $\text{dist}(o_i, o_j)$ définie entre deux échantillons (corrélation entre deux spectres, distance contextuelle, ...). Soit $\{o_1, \ldots, o_i, \ldots, o_N\}$ N échantillons, K le nombre de classes, g_k et σ_k le centre de gravité et la variance de la classe C_k, et σ la variance du jeu de données complet. Nous utilisons les mesures suivantes :

L'inertie intra-classe mesure la variance entre les échantillons et le centre de gravité de leurs classes respectives [Halkidi et al., 2001] :

$$I_t = \sum_{k=1}^{K} \sum_{i \in C_k} \text{dist}(o_i, g_k)^2 = \sum_{k=1}^{K} \sigma_k \tag{4.16}$$

La compacité mesure le degré de regroupement des échantillons au sein de leurs classes [Halkidi et al., 2001] :

$$Cmp = \frac{1}{K} \cdot \sum_{k=1}^{K} \frac{\sigma_k}{\sigma} \tag{4.17}$$

Le critère Xie-Beni est un critère indépendant de l'échelle des valeurs permettant de mesurer la séparation des classes [Xie et Beni, 1991] :

$$XB = \frac{I_t}{N \cdot \min_{i,j \in K, i \neq j}(\text{dist}(g_j, g_i))} \tag{4.18}$$

Le critère WG est aussi un critère indépendant de l'échelle mesurant la séparation et la compacité des objets des classes [Gançarski et Wemmert, 2005] :

$$WG = \frac{I_t}{N \cdot \min_{o_i \in C_k, k' \neq k}(\text{dist}(o_i, g_{k'}))} \tag{4.19}$$

Nous avons surtout utilisé ces critères comme mesure statistique de la qualité des classes obtenues après l'apprentissage, mais ils peuvent être employés dans la fonction d'évaluation pour servir le même objectif lors de la recherche des règles de classification. Les deux derniers critères sont particulièrement intéressants car ils sont indépendants de l'échelle des données.

4.6.1.3 Mesures spécifiques à la représentation

Nous allons voir dans cette partie un autre type de mesures, basées sur la représentation des règles plutôt que sur le domaine d'application. Les critères proposés ici s'appliquent sur chaque individu et, par comparaison des notes obtenues au sein d'une population, ils permettent d'évaluer la diversité ou au contraire l'homogénéité du pool génétique. Ces renseignements sont utiles pour la fonction d'évaluation (on teste si un individu est conforme aux désirs de l'expert), les opérateurs de sélection ou l'opérateur de convergence (renforcer les mutations génétiques si le pool est trop homogène, ou stopper l'apprentissage). Il s'agit de voir un individu génétique comme un arbre sans tenir compte de la sémantique des nœuds. Ces critères sont donc utilisables avec toutes les représentations non plates (tels que celles des algorithmes **ICU** ou **GramGen** décrits dans le chapitre suivant), sinon la mesure n'aurait pas d'intérêt.

Habituellement, on dit qu'un arbre est *parfaitement équilibré* si, pour chacun de ses nœuds n, tous les fils de ces nœud ont le même nombre de nœuds. Un arbre de type *AVL* est un arbre *bien équilibré* au sens des hauteurs, plutôt que du nombre de nœuds. Malheureusement ces mesures ne permettent pas de caractériser les arbres qui ne sont pas parfaits, comme c'est le cas des individus produits par une méthode évolutive. Pour cela, nous avons développé plusieurs mesures caractérisant les arbres par une note continue, permettant d'en avoir une évaluation graduelle. Nous proposons de regrouper ces mesures en cinq classes :

1. Celles qui sont affectées à l'arbre complet sont désignées par un symbole isolé (X).
2. Celles qui sont affectées à un nœud n donné sont désignées par X^n.
3. Celles qui consistent à prendre le maximum d'une mesure appliquée à l'ensemble des nœuds sont désignées par X_M.

4. Celles qui consistent à prendre la moyenne d'une mesure appliquée à l'ensemble des nœuds sont désignées par X_A.
5. Enfin, celles qui consistent à affecter une pondération à une mesure de nœud, qui décroît en fonction de la profondeur de l'arbre, sont désignées par X_G.

Soit F^n le nombre de fils du nœud n, T^n le nombre de descendants du nœud n et D^n la profondeur de l'arbre à partir du nœud n. Nous commençons par rappeler les mesures connues en théorie des graphes :

- T est la taille de l'arbre en nombre de nœuds,
- W_M est la largeur maximale de l'arbre (correspondant au nombre maximal de nœuds dans le même niveau),
- W_A est la largeur moyenne de l'arbre,
- L est le nombre de feuilles de l'arbre,
- D est la profondeur (maximale) de l'arbre.

Ces mesures sont utilisées par les algorithmes qui engendrent des représentations arborescentes en plus de nouvelles mesures, que nous avons développées, et qui sont les suivantes :

- $N_{D_{max}}$ est le nœud ayant la plus grande profondeur D_{max},
- $N_{D_{min}}$ est le nœud ayant la plus petite profondeur D_{min},
- Si $A^{n(i)}$ sont les sous-arbres engendrés par les fils $n(i)$ du nœud n, la différence entre les profondeurs minimale et maximale des sous-arbres est la *différence de profondeur* du nœuds n :

$$D_N^n = \max\left[D^{n(i)}\right] - \min\left[D^{n(i)}\right] \tag{4.20}$$

- la *différence de profondeur dégressive* de n est :

$$D_G^n = \begin{cases} 0 & \text{si } F^n = 0 \\ D_N^n + \frac{\sum D_G^{n(i)}}{2 \cdot F^n} & \text{si } F^n > 0 \end{cases} \tag{4.21}$$

où $n(i)$ est le fils numéro i du nœud n.

La plupart de ces mesures, c'est-à-dire T, W_M, W_A, L, D, D_{max}, D_{min}, D_N^n et D_G^n, peuvent être utilisées pour obtenir des caractéristiques intéressantes pour les individus engendrés depuis une représentation arborescente. Par exemple, la taille T et le nombre de feuilles L peuvent être utilisées comme des mesures de complexité des arbres. La différence de profondeur dégressive caractérise la différence de profondeur maximale des nœuds frères, au fur et à mesure que l'on s'enfonce dans l'arbre.

La figure 4.4 illustre le procédé. Trois arbres sont présentés : le premier est correctement équilibré, le second ne l'est pas, mais le troisième doit être caractérisé comme ayant un équilibrage intermédiaire par rapport aux deux premiers. Le calcul du critère D_G^n est analysé pour chacun des arbres. Le premier obtient la note 0 (correctement équilibré), le second la note 2.25 (mal équilibré) et le dernier la note 1.

Ces mesures permettent de trier, de comparer et d'apprécier la diversité d'un pool génétique. De plus, elles sont utiles pour la technique du *sharing*[4]. La détection de la convergence de la population vers des optima locaux peut être anticipée par l'analyse de la ressemblance des individus. D'autres mesures sont cependant utiles pour vérifier la qualité de l'apprentissage : nous les exposons dans la partie suivante.

4.6.2 Mesures calculées après l'apprentissage

4.6.2.1 Basées sur l'expertise

Comme lors de l'apprentissage, où l'expertise est utilisée afin de guider la recherche de la population de règles optimales, à la fin de l'apprentissage la qualité des règles produites peut être mesurée grâce à une matrice de confusion (MC) obtenue à partir d'un jeu d'exemples qui ont été dissimulés à l'algorithme. Soit M_n une matrice de confusion carrée définie par :

[4]Le *sharing* consiste à modifier la *fitness* utilisée par le processus de sélection pour éviter le rassemblement des individus autour d'un *mode* dominant : on pénalise leur évaluation en fonction du taux d'agrégation de la population dans le voisinage d'un individu. Les individus proches les uns des autres doivent partager leur *fitness* donc plus les individus sont regroupés, plus leur *fitness* est faible [A525G, 2005].

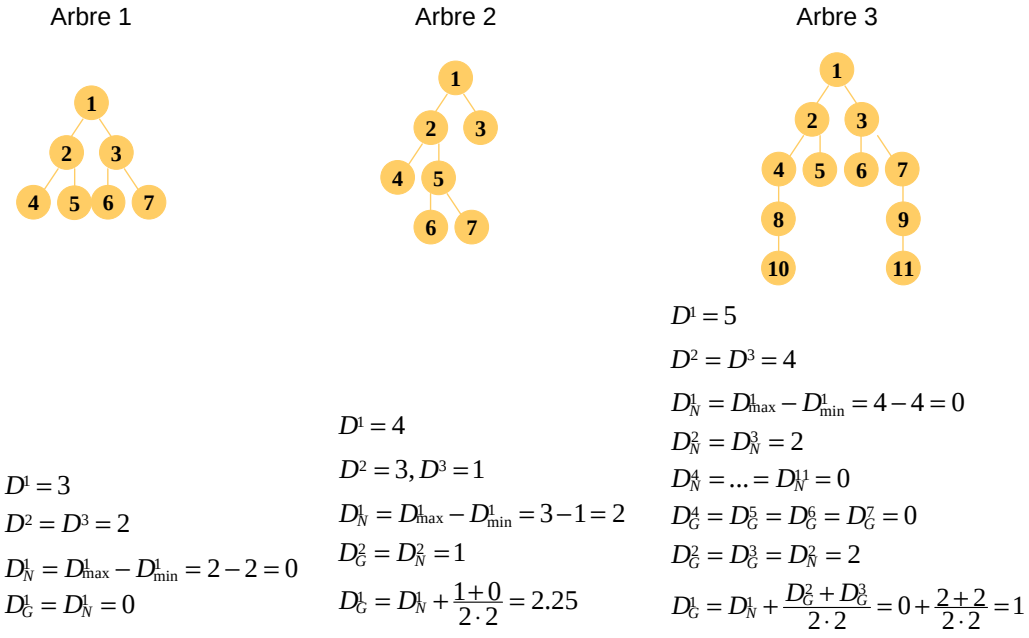


FIG. 4.4 – Exemple de trois arbres et de la mesure D^n_G associée à chacun des sommets.

$$M_n = \begin{bmatrix} m_{1,1} & \cdots & m_{1,n} \\ \vdots & & \vdots \\ m_{n,1} & \cdots & m_{n,n} \end{bmatrix} \tag{4.22}$$

où $m_{i,j}$ est le nombre d'échantillons classés par les règles dans la classe i alors que l'expert les classe dans la classe j. Les équations suivantes présentent deux mesures traditionnelles, la qualité globale G_{acc} [Landis et Koch, 1977] et l'index κ de Cohen [Cohen, 1960] :

$$G_{acc} = \frac{\sum_{i=1}^{n} m_{i,i}}{\sum\sum m_{i,j}} \tag{4.23}$$

$$\delta = \sum m_{i,i} \tag{4.24}$$

$$\eta = \sum\sum m_{i,j} \tag{4.25}$$

$$\lambda = \frac{\sum_{c=1}^{n} [\sum m_{i,c} \cdot \sum m_{c,j}]}{\eta} \tag{4.26}$$

$$\kappa = \frac{\eta\delta - \lambda}{\eta^2 - \lambda} \tag{4.27}$$

D'une façon générale, beaucoup de problèmes de classification sont solubles en mode *hard*. Cependant, certains algorithmes de conception moderne (comme ceux que nous verrons au chapitre suivant) établissent des règles de classification de manière parallèle pour les différentes classes, ce qui leur permet de résoudre des problèmes de type *soft* car un échantillon peut provoquer l'activation de plusieurs de ces règles. Néanmoins, pour les besoins du calcul de la matrice de confusion, le résultat de ces algorithmes doit être converti en classification *hard*. Le choix des classes se fait alors de manière déterministe, soit à

partir d'un seuillage, soit en fonction d'un tournoi entre classifieurs, par exemple en utilisant certains paramètres comme leur *fitness*. Outre le fait que ceci biaise les mesures de qualité obtenues depuis la matrice de confusion, il est relativement difficile d'extraire des informations, à titre instructif, d'une éventuelle déficience du pool de règles. Nous avons alors développé un autre style de matrice de confusion, nommé matrice de confusion directe (ang., *Direct Confusion Matrix* ou *DCM*) indiquant les performances réelles de ces classifieurs sans le biais de l'algorithme de conversion d'un résultat flou en résultat *hard*. Elles sont de même taille que les matrices standards, mais le nombre réel d'activations des classifieurs correspondant à la classe i est utilisé à la place de la valeur normalisée $m_{i,j}$ (dans l'équation 4.22). Ainsi, lorsque la somme des colonnes des matrices standards est de 100%, un résultat différent peut être obtenu dans une *DCM*. Par exemple, si la somme est nulle, cela indique que la classe n'a été apprise par aucun classifieur, alors qu'avec la méthode de conversion déterministe, il y a une probabilité non nulle de sélectionner la bonne classe de manière totalement artificielle ! On peut donc considérer que la *DCM* constitue un test plus difficile pour des algorithmes de classification *soft* ou floue. Toutes les mesures présentées ici (G_{acc} et κ) peuvent être appliquées sur une *DCM*.

4.6.2.2 Validation graphique

L'inconvénient des *CM* ou des *DCM* est le fait qu'il est difficile d'extrapoler depuis une matrice de confusion la localisation géographique – à destination des experts – des erreurs commises par les classifieurs. Les cartes de recouvrement, que nous avons conçues, viennent alors en soutien (voir la figure 4.5). Elles symbolisent le degré de recouvrement des règles pour chacun des pixels de l'image. Les pixels rouges indiquent les endroits qui ne sont couverts par aucune règle et en dégradé de gris les pixels qui activent de une (en noir) à plusieurs règles (en blanc). Une bonne carte de recouvrement est totalement noire. Les pixels rouges indiquent souvent les classes non apprises ou celles qu'il serait judicieux d'inclure dans l'ensemble d'apprentissage, rendant le processus de classification incrémental. Généralement, deux ou trois règles se recouvrent dans les cas les plus complexes.

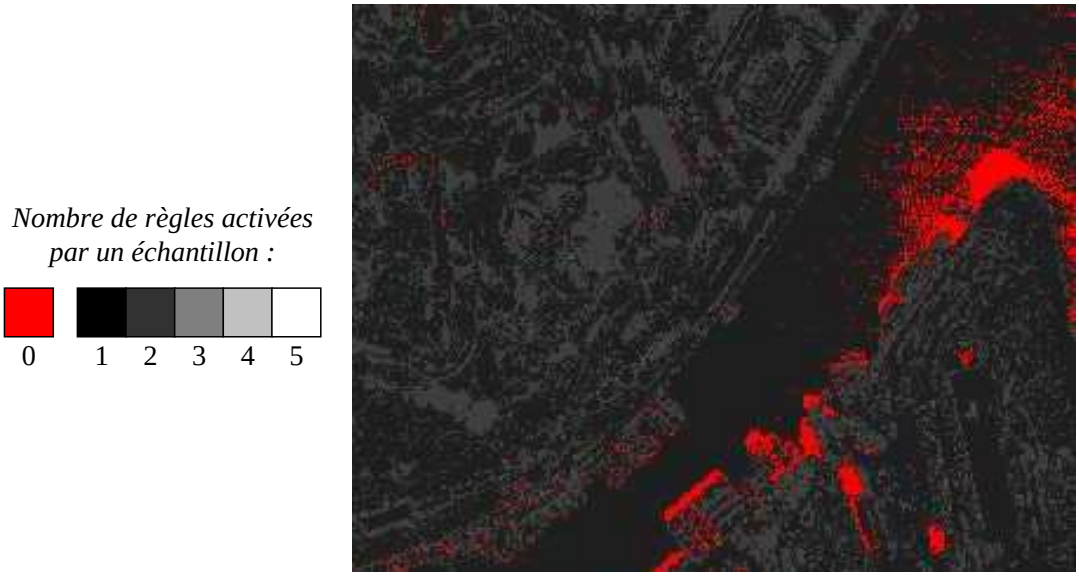


FIG. 4.5 – Carte de recouvrement pour la zone de Strasbourg.

La validation graphique donne souvent des conclusions intéressantes, mais les protocoles de validation permettent une automatisation plus pratique du processus de validation. Nous les exposons dans la partie suivante.

4.6.2.3 Protocoles de validation

Différents protocoles de validation ont été intégrés au sein d'une plate-forme nommée **VPlat** (ang., *Validation Platform*, voir l'annexe F), afin d'obtenir une mesure de qualité plus précise sur l'apprentissage lui-même, notamment concernant l' expansibilité des algorithmes, critère qui fait partie de la liste définie dans la section 1.2. Nous avons utilisé les stratégies par *holding-out*, *k-fold cross-validation* (validation croisée à k partitions), *bootstrapping* et *jackknifing*. On pourrait les qualifier de méta-apprentissage car le rôle de ces stratégies est de lancer différents apprentissages successifs afin de calculer l'erreur de généralisation en utilisant une mesure de qualité quelconque parmi celles présentées dans cette section, et car elles peuvent exhiber plusieurs ensembles de règles de qualité croissante [Hjorth, 1994].

La stratégie de validation croisée à k partitions [Plutowski et al., 1994] relance l'apprentissage k fois sur le jeu de données complet découpé en k partitions. Plusieurs stratégies sont possibles : en choisissant à chaque fois l'une des partitions pour les tests et les $k-1$ restantes pour l'apprentissage (variante *leave-one-out*), ou en regroupant $N < k$ partitions pour les tests et $k-N$ partitions pour l'apprentissage (variante *leave-N-out*). Chaque partition est donc utilisée en validation pour *une* expérimentation et en apprentissage pour $k-1$ expérimentations. La note finale est la moyenne des notes obtenues pour chaque expérimentation. Ces notes sont souvent calculées en utilisant Q_{accur}, mais l'index de κ est aussi recommandé. La stratégie du *holding-out* est une validation croisée dégénérée à deux partitions de tailles différentes.

Le *bootstrapping* [Shao et Tu, 1995] est une autre stratégie populaire dans laquelle les échantillons d'apprentissage et de validation sont sélectionnés aléatoirement. Ensuite, le fonctionnement est le même que pour la stratégie dite *holding-out*. Le principal intérêt est le fait que l'on peut lancer des cycles d'apprentissage en nombre illimité, tant que l'on espère réduire l'intervalle de confiance des mesures de qualité jusqu'à la taille désirée. On peut alors extraire un jeu de règles performant. Une variante consiste à préparer k jeux d'apprentissage composés pour chacun d'une proportion d'échantillons tirés au hasard dans le jeu de données, et complétés par des échantillons créés de manière totalement aléatoire. Les jeux de test correspondants sont constitués uniquement d'exemples (réels) qui n'ont pas été appris. L'intérêt est de tester la robustesse de l'algorithme ainsi que sa capacité de généralisation. Le nombre de jeux, leur taille et la proportion d'exemples aléatoires ne sont pas limitées par le nombre de données, ce qui est très utile, par exemple, lorsque le nombre de validations terrain est faible.

Enfin, une stratégie moins connue, le *jackknifing*, est quasi-similaire au *bootstrapping* ou à la variante *leave-one-out* de la validation croisée [Shao et Tu, 1995]. Il s'agit de tirer aléatoirement un échantillon pour la validation, d'apprendre sur le reste et de recommencer autant d'apprentissages que désiré. Des variantes existent en tirant plusieurs échantillons sans remise (les jeux de données étant alors susceptibles de contenir plusieurs fois le même échantillon), ou en tirant les échantillons pour l'ensemble d'apprentissage plutôt que pour la validation. Cependant, le *bootstrapping* est utilisé pour calculer l'erreur de généralisation, alors que le *jackknifing* est utilisé pour mesurer le biais d'une mesure : des mesures d'intérêt M_k sont calculées sur chaque sous-ensemble d'échantillons puis elles sont comparées à la mesure M obtenue pour le jeu complet, afin d'estimer son biais.

Une autre stratégie par méta-apprentissage existe, qui teste l'influence d'un paramètre de l'apprentissage plutôt que l'échantillonnage des exemples. Il s'agit des courbes ROC. Nous les décrivons dans la partie suivante.

4.6.2.4 Courbes ROC

La courbe ROC (ang., *Receiver Operating Characteristic*) est une mesure statistique conçue dans les années 1950 pour donner une vue multi-niveaux d'un test [Metz, 1978; Long et al., 1988]. Le test ne peut concerner qu'une seule classe (et sa non-classe). Il tient compte de $1 - Q_{spe}$ (en abscisse) et de Q_{sens} (en ordonnée). La courbe est associée à un paramètre p : un nouveau pool de règles $\Gamma_p = \Phi(\Gamma_0, p)$ est généré depuis un pool initial Γ_0 (correspondant à un pool de référence appris avec le jeu d'apprentissage), puis est appliqué sur le jeu de test. La fonction de génération Φ ainsi que le paramètre p doivent être choisis de telle sorte que pour $p = 0$, le test ne donne que des réponses négatives ($Q_{sens} = 1 - Q_{spe} = 0$) et que pour $p = 1$, le test ne donne que des réponses positives ($Q_{sens} = 1 - Q_{spe} = 1$). Plusieurs courbes

peuvent être comparées en utilisant des paramètres différents. D'autres mesures y sont souvent associées, comme l'aire sous la courbe (qui doit être la plus grande possible) ou la distance AC_d (qui doit être la plus courte possible, voir la figure 4.6). AC_d mesure la distance entre le point singulier de la courbe ROC et le point $(0, 1)$ est considéré comme le classifieur parfait. Elle est calculée par :

$$AC_d = 1 - \sqrt{p \cdot (1 - Q_{sens})^2 + (1 - p) \cdot (1 - Q_{spe})^2} \tag{4.28}$$

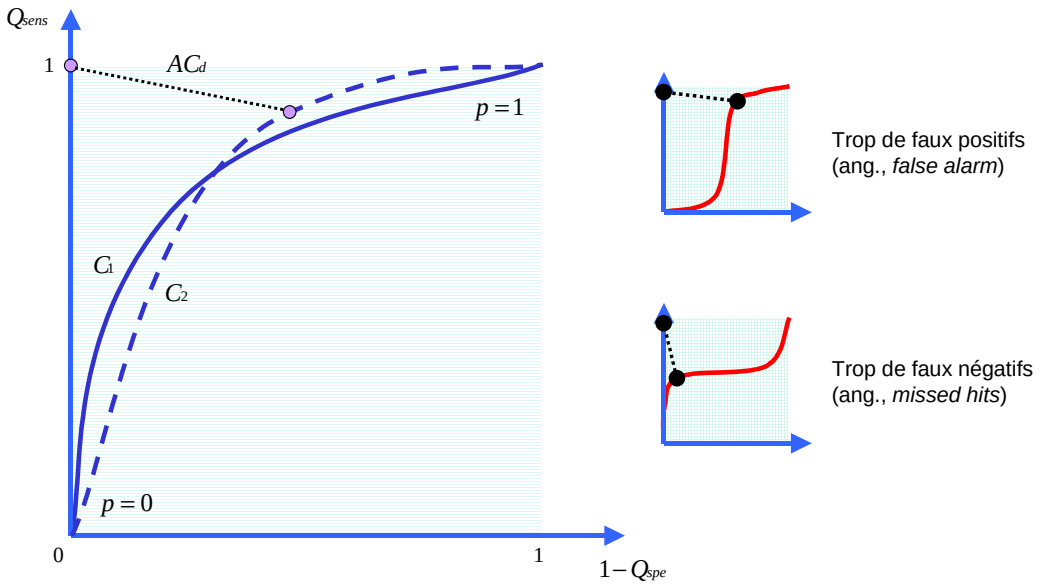


FIG. 4.6 – Exemple schématique d'une courbe ROC et d'une mesure associée.

Sur la figure 4.6, le test C_1 renvoie moins de faux positifs et de faux négatifs pour un paramètre restrictif ($p \to 0$), il est donc intéressant pour le *diagnostic* (le résultat est meilleur si le test est ciblé pour un échantillon donné). À l'inverse, le test C_2 renvoie moins de faux positifs et de faux négatifs pour un paramètre plus large ($p \to 1$), ce qui est intéressant pour le *dépistage* (et permet un brassage plus large des cas, le test étant meilleur en cas de sensibilité accrue par l'augmentation du paramètre). À titre d'exemple, dans nos études de cas, nous avons fait correspondre p à un coefficient de variation de la taille du domaine de définition du classifieur, par exemple la surface des différents intervalles, car c'est un point commun à beaucoup de représentations que nous avons choisies.

4.7 Conclusion

Nous avons présenté les notions concernant les différents problèmes de classification à résoudre (*hard*, *soft*, flous ou à intervalles flous), notions importantes pour la suite de cette thèse. L'existence de ces différents types façonne le formalisme des règles, à la fois au niveau de la représentation de leurs conditions, et au niveau de la représentation de la prise de décision pour la classification. Des solutions pour les classifications de type *hard*, *soft* ou floues sont déjà connues en télédétection, mais nous proposons, dans le chapitre suivant, un nouveau type d'algorithme pour les classifications à intervalles flous. À cette intention, nous avons présenté une représentation de règle générique ainsi qu'une terminologie qui nous servira de cadre de travail par la suite. Nous avons aussi défini l'algorithme général du processus de découverte des règles. De plus amples détails concernant le fonctionnement général des opérateurs génétiques peuvent être trouvés dans l'annexe C, ainsi que dans nos publications [Quirin, 2002; Korczak

et Quirin, 2003a; Korczak et Quirin, 2003c; Quirin et al., 2004]. Même si les algorithmes évolutionnaires sont connus pour s'exécuter dans un temps plutôt long, nous avons dégagé une liste de facteurs pouvant atténuer la complexité algorithmique de notre système. Notamment, certains paramètres au niveau des opérateurs génétiques ou de l'algorithme lui-même peuvent intervenir favorablement. Enfin, la dernière section a été réservée à la présentation des différentes mesures de qualité et protocoles de validation qui sont utilisés au sein des divers algorithmes présentés dans le chapitre suivant, mais qui ne pourront pas être développés pour chacun d'eux. Nous avons présenté ou proposé des techniques utilisables durant l'apprentissage, en test ou en validation et adaptées soit au domaine d'étude, soit à la représentation des règles afin de tester leur qualité ou servir comme base de comparaison. Nous avons maintenant les bases théoriques nécessaires pour aborder la description précise des algorithmes d'apprentissage.

Chapitre 5

Algorithmes de classification

5.1 Introduction

Au fil de l'étude des différentes qualités et faiblesses que nous avons rencontrées avec les diverses représentations, nous avons préféré développer plusieurs approches distinctes car nous n'en avons trouvé aucune qui disposait d'un potentiel suffisamment étendu pour s'aligner sur tous nos critères. À partir de l'étude des représentations, nous allons voir que chacune d'elles s'adresse à un type de problème différent. Nous étudierons donc les opérateurs génétiques et les diverses fonctions d'évaluation qui leur sont associées. Afin de présenter à l'utilisateur un modèle unique et par conséquent cohérent, la totalité des algorithmes dont nous allons parler a été *centralisée* sous une plate-forme unique, nommée **VPlat** (voir l'annexe F) qui répond à trois objectifs : (1) abstraire le format des données d'entrée (images, textes, bases de données, ...) et de sortie (classifications), (2) abstraire le paramétrage et le lancement des différentes méthodes d'apprentissage et (3) en profiter pour proposer des modèles de validation génériques manipulant finalement des données et des méthodes d'apprentissage abstraites (par exemple la validation croisée est un tel modèle). Puisque très technique, nous ne détaillerons pas ici le fonctionnement exact de cette plate-forme.

Afin d'apporter une réponse variée aux problèmes de classification d'images, c'est-à-dire aux problèmes *hard*, *soft*, flous ou par intervalles flous, nous avons développé quatre types de méthodes de classification différentes que nous exposerons dans les deux sections principales de ce chapitre. Dans chaque partie, deux algorithmes différents seront présentés. La première partie (section 5.2) détaille le fonctionnement des algorithmes adaptés aux problèmes que nous qualifierons de classiques et que nous prendrons comme référence dans la suite. Ces algorithmes sont adaptés au traitement de problèmes de type *hard* ou *soft*. Les algorithmes **ICU** et **XCS-R**, les représentations qu'ils mettent en œuvre, ainsi que les opérateurs correspondants seront détaillés. Il s'agit d'algorithmes permettant d'obtenir des règles de classification performantes, mais nombreuses dans le cas de **XCS-R**. Enfin, dans la seconde partie (section 5.3), nous adapterons les algorithmes précédents au problème de la classification floue. Nous parlerons notamment de **ICUX**, une extension de **ICU** et de **GramGen**, un algorithme à base de programmation génétique. Pour le lecteur non initié aux algorithmes évolutionnaires, nous avons décrit les principales notions les concernant dans l'annexe C.

5.2 Méthodes d'apprentissage adaptées à la classification *soft*

5.2.1 L'algorithme ICU

5.2.1.1 Représentation des règles

ICU (*I See You*) est un algorithme permettant de faire de la classification *soft*. Nous présentons dans cette partie et les parties suivantes la représentation des règles ainsi que certaines fonctions génétiques.

Le lecteur intéressé pourra se référer à [Quirin, 2002] et à [Korczak et Quirin, 2003b] pour d'autres informations.

Les règles manipulées par **ICU** se veulent simples et expressives, mais permettant tout de même d'aborder une certaine complexité dans les données, par exemple de celles évoqués sur la figure 3.9 (page 50), à propos des aspects multiples que peut revêtir un spectre de la même classe. **ICU** associe à un échantillon caractérisé par un *vecteur spectral* (ensemble des valeurs de radiance de cet échantillon pour chaque bande) l'instance de la classe thématique correspondante. Chaque règle est indépendante des autres et ne traite qu'une seule classe à la fois (les exemples des autres classes sont à chaque fois considérés comme des contre-exemples par l'algorithme d'apprentissage) et représente un seul individu génétique. La structure de données capable de mémoriser une telle association est décrite ci-dessous.

En classification *soft*, la partie <*action*> des règles indique la classe sous la forme d'un attribut nominatif. Concernant la partie <*condition*>, après plusieurs essais de structures légèrement différentes, nous avons décidé qu'elle reposerait sur la notion d'*intervalle spectral*. Un tel intervalle est un couple de nombres entiers, entre 0 et la valeur maximale possible pour un échantillon et pour une bande donnée (255 pour les échantillons définis sur 8 bits, 65536 pour les échantillons définis sur 16 bits, etc), qui permet de découper l'espace des valeurs spectrales en deux espaces : celui des valeurs souhaitées pour les échantillons d'une même classe, et le reste.

Pour généraliser ce concept et mieux cibler les classes, nous affectons à chaque bande disponible dans l'image brute un ensemble d'intervalles éventuellement disjoint. La forme définitive de la partie <*condition*> d'une règle est la suivante :

$$<condition> := E_1 \wedge E_2 \wedge E_3 \wedge \ldots \wedge E_n \tag{5.1}$$

où n est le nombre total de bandes et E_i sont des ensembles d'intervalles.

Les ensembles d'intervalles E_i sont définis pour toutes les bandes présentes dans les données brutes. Chaque ensemble défini un ou plusieurs intervalles spectraux de la forme suivante :

$$E_i := [m_{i_1}; M_{i_1}] \vee [m_{i_2}; M_{i_2}] \vee [m_{i_3}; M_{i_3}] \vee \ldots \vee [m_{i_p}; M_{i_p}] \tag{5.2}$$

où $[m_{i_j}; M_{i_j}]$ est un intervalle spectral et i_p leur nombre, qui dépend du numéro de bande i. m_{i_j} et M_{i_j} sont respectivement les radiances minimale et maximale autorisées d'un échantillon pour la bande i, pour que celui-ci active la règle (dans ce cas, $m_{i_j} \leq b_j \leq M_{i_j}$).

Les intervalles $[m_{i_j}; M_{i_j}]$ ne sont pas forcément disjoints : par expérimentation, nous nous sommes rendu compte que le fait de garder des intervalles non disjoints (au lieu de les fusionner puisque mathématiquement cela revenait au même) permettait à l'algorithme de mieux fonctionner. La fusion tend à diminuer significativement le nombre d'intervalles, et les règles n'ont pas une diversité suffisante pour pouvoir s'améliorer. Voici un exemple de fusion pour clarifier les choses :

$$E = [11; 105] \vee [138; 209] \vee [93; 208] \equiv E = [11; 209]$$

Enfin, le nombre d'intervalles spectraux i_p dans un ensemble est autorisé à varier pour chaque ensemble E_i et pour chaque classe en fonction du nombre de sous-conditions nécessaires à l'algorithme pour rendre la règle fiable. i_p est cependant inférieur à une limite fixée par l'utilisateur (par défaut 5) et doit être non nul ($0 < i_p \leq 5$).

Pour satisfaire la règle, un échantillon, défini *a fortiori* sur toutes les bandes de l'image, doit donc satisfaire chacun des ensembles d'intervalles spectraux de chaque bande. Pour satisfaire un ensemble d'intervalles spectraux, il doit satisfaire seulement l'un des intervalles de cet ensemble. **Les règles définissent donc des conjonctions de disjonctions d'intervalles**.

Cette représentation a surtout été choisie parce qu'elle apporte de la simplicité dans les résultats présentés à l'utilisateur et par le fait que l'expression de contraintes multiples reste tout de même compacte [Quirin, 2002]. De plus, la notion d'ensemble d'intervalles est capable de saisir l'hétérogénéité des échantillons qui appartiendraient à la même classe. Ces règles permettent enfin une rapidité d'évaluation qui peut être importante dans certains domaines d'application (flux d'images vidéo, ...).

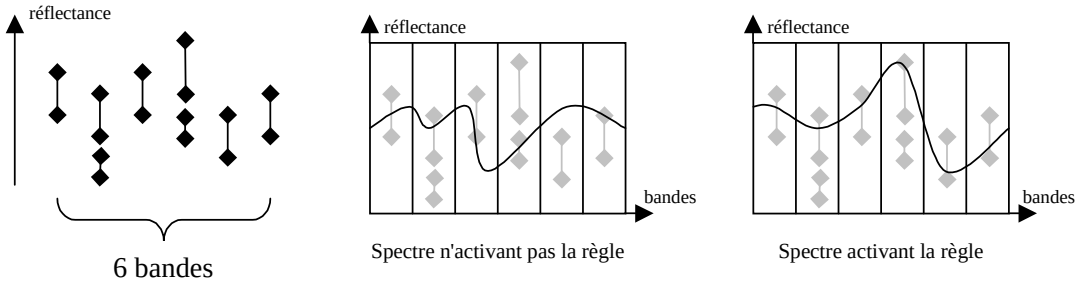


FIG. 5.1 – Correspondance entre une règle et un spectre donné.

La figure 5.1 indique dans quelle mesure une règle peut caractériser le spectre d'un échantillon donné. Les intervalles de la règle sont simplement testés les uns après les autres, jusqu'à ce que l'un d'entre eux ne correspondent plus aux valeurs de radiance de l'échantillon. Chaque intervalle est donc une contrainte sur le spectre : la règle complète forme une conjonction de disjonctions de contraintes.

5.2.1.2 Fonction d'évaluation

La fonction d'évaluation associe une note à chacune des règles de la population courante (voir l'annexe C). Son principe est simple. Elle tient compte de quatre quantités, présentées dans le tableau 5.2, en fonction du fait que la partie <*condition*> de la règle soit activée ou non par un échantillon brut et que la classe prédite par la règle correspond ou non à celle déterminée par l'expert.

		Quantité d'échantillons bruts	
		... activant R	n'activant pas R
La classe correspond à l'expert E	Oui	P_{exp}^{reg}	$P_{exp}^{\overline{reg}}$
	Non	$P_{\overline{exp}}^{reg}$	$P_{\overline{exp}}^{\overline{reg}}$

FIG. 5.2 – Caractéristiques de la fonction d'évaluation pour **ICU**.

Une première note, appelée N_{class}, concernant les échantillons classés au vu de l'expert dans la classe courante, est déterminée en fonction du nombre d'échantillons que la règle a su classifier correctement. Elle représente le pourcentage des échantillons classifiés dans la classe X par la règle par rapport à tous les échantillons étiquetés dans X par l'expert :

$$N_{class} = \frac{P_{exp}^{reg}}{P_{exp}^{reg} + P_{exp}^{\overline{reg}}} \tag{5.3}$$

Une deuxième note, appelée $N_{\overline{class}}$, concernant les échantillons classés au vu de l'expert comme n'appartenant pas à la classe courante, est déterminée en fonction du nombre d'échantillons que la règle n'a pas affectés non plus dans la classe courante. Elle représente le pourcentage d'échantillons écartés de la classe X par la règle par rapport à tous les échantillons non étiquetés dans X par l'expert :

$$N_{\overline{class}} = \frac{P_{\overline{exp}}^{\overline{reg}}}{P_{\overline{exp}}^{\overline{reg}} + P_{\overline{exp}}^{reg}} \tag{5.4}$$

Nous souhaitions résoudre le problème des classes sous-représentées : si une classe contient peu d'échantillons qui lui sont représentatifs, la note ne doit pas défavoriser cette classe. Par exemple, un problème contenant 500 échantillons d'une classe C_1 et 10 échantillons d'une classe C_2, risque de produire une règle du type : **si** *vrai* **alors** C_1. Cette règle obtiendrait alors une note quasi-optimale (correcte dans 98% des cas), mais elle aurait un faible pouvoir de généralisation. Prendre la moyenne des deux notes (5.3)

et (5.4) permet de résoudre parfaitement le problème des classes sous-représentées (et sur-représentées car la difficulté est symétrique). En effet, la note subira un sérieux déficit si la règle s'active alors qu'elle ne le devrait pas (dans l'exemple ci-dessus, la note obtenue serait égale à 0.5).

Néanmoins, il est peut être du désir de l'expert de vouloir pénaliser les classes représentées de manière extrême. Par exemple, des classes faiblement représentées que l'expert souhaiterait requalifier en examinant si les échantillons correspondants n'appartiendraient finalement pas à une autre classe. Il peut ainsi jouer avec une classe de type "*inconnue*" en laissant à l'algorithme la possibilité (si c'est pertinent) de reclasser les échantillons ailleurs, ou de les laisser en place. Bien que ce besoin soit assez rare, nous avons mis en place un *coefficient de reclassement* C_{class}, pour aboutir à la forme finale de la note :

$$N_{final} = C_{class} N_{class} + \frac{1}{C_{class}} N_{\overline{class}} \tag{5.5}$$

En situation réelle, ces coefficients n'ont pas réellement d'intérêt et nous conseillons de leur donner la valeur 0.5, car cette valeur permet d'obtenir une qualité convenable lorsque l'on ne dispose pas de connaissances préalables sur la distribution de chaque classe.

La fonction *fitness* F_i mesure l'adaptation ou la performance d'un individu i par rapport aux autres. Elle est calculée à partir de la fonction d'évaluation f_i :

$$F_i = \frac{f_i}{\sum f_k} \tag{5.6}$$

5.2.1.3 Création d'un individu initial

Nous avons proposé deux approches pour créer l'individu I_0, selon deux méthodes différentes : la méthode *GenMinMax* et la méthode *GenSpectro*, que nous décrivons ci-dessous. La première présente l'avantage d'être rapide, la seconde permet de traiter certains cas plus complexes.

La méthode GenMinMax. Elle consiste à comparer les données brutes avec l'expert. Elle définit, pour chaque échantillon appartenant à la classe demandée et pour chaque bande spectrale disponible, la valeur de radiance minimale et maximale observée sur les données brutes. Ensuite, elle duplique chaque intervalle un certain nombre de fois, spécifié par l'utilisateur afin de créer l'ensemble d'intervalles spectraux ($E_{i_1} \vee \ldots \vee E_{i_p}$). On obtient ainsi une règle de classification grossière, classant correctement tous les échantillons de la classe en question (puisque chaque échantillon sera forcément dans la plage encadrée par les bornes minimale et maximale). Bien entendu, des règles issues de plusieurs classes différentes se recouvriront. De plus, la méthode ne fonctionne pas correctement si pour la même classe, l'image contient des valeurs de radiance extrêmes. Cependant, pour des images de type SPOT (faible nombre de bandes et petit domaine de définition des valeurs de radiance) cette méthode simpliste convient parfaitement : en effet les échantillons appartenant à la même classe ont rarement des valeurs éloignées de la moyenne observée pour toute la classe, les écarts-types étant souvent très faibles.

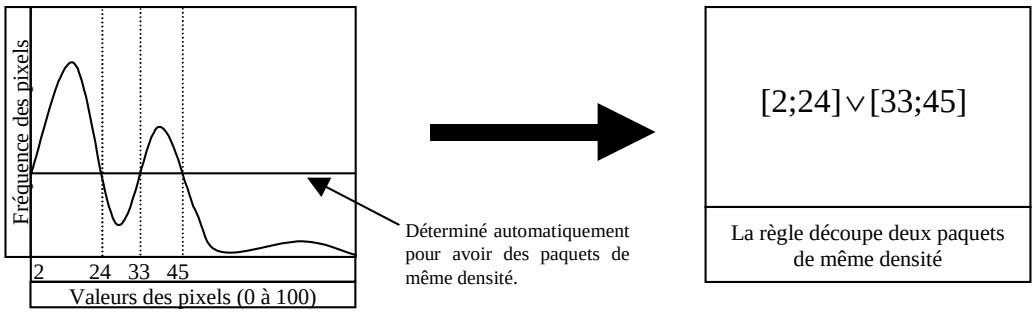


FIG. 5.3 – Méthode d'initialisation GenSpectro (**ICU**).

La méthode GenSpectro. Cette méthode beaucoup plus complexe est à réserver aux images présentant des domaines de définition plus vastes pour les valeurs de radiance. Le nombre de bandes n'est pas un facteur d'explosion combinatoire puisque chacune d'elle est analysée de manière indépendante comme dans la première méthode. Cependant, nous avons cherché à résoudre le problème des valeurs des échantillons extrêmes (ang., *outliers*), et à insuffler une première idée de discrimination à notre algorithme. Le principe consiste à découvrir pour une bande donnée, la répartition des valeurs des échantillons afin de créer autant d'intervalles que nécessaire pour décrire ces valeurs (toujours sous la forme d'une conjonction). La méthode repose sur la création d'un spectrogramme[1] à partir des données brutes et des données expertes. Grâce à ces vecteurs de probabilités, le programme détermine des paquets de même densité caractérisant le spectre des échantillons de cette classe. Le nombre de paquets maximal est déterminé par l'utilisateur et équivaut au nombre de disjonctions qu'il souhaite obtenir dans le pire des cas. La figure 5.3 présente la répartition des valeurs d'un échantillon dans une classe et une bande arbitraire, et la création de la règle originale, selon la méthode GenSpectro.

Dans le cas de données à valeurs extrêmes, autant d'intervalles sont créés pour les contenir, quelles que soient leurs positions dans l'espace des données. Cette méthode est indépendante de l'échelle et de l'irrégularité des données.

5.2.1.4 Opérateur de croisement

Il consiste à sélectionner, dans deux règles, un ensemble d'intervalles E_i correspondant à la même bande spectrale, de telle sorte que l'opérateur puisse être déclaré cohérent, puis à les échanger. La validation (vérification des bornes) peut se poursuivre ou non par une fusion (au choix de l'utilisateur), consistant à fusionner les intervalles au sens mathématique du terme. Nous montrons sur la figure 5.4 le résultat du croisement de deux individus contenant chacun un seul ensemble d'intervalles à échanger. Nous avons choisi d'appliquer, dans **ICU**, un croisement uniforme car nous en avons obtenu les meilleurs résultats.

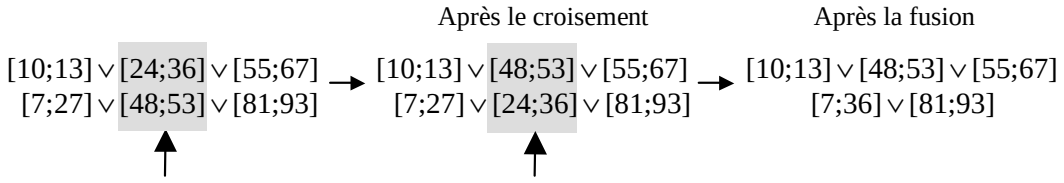


FIG. 5.4 – Illustration de l'opérateur de croisement et de la fusion (**ICU**).

Cependant, la fusion produit à long terme une diminution du nombre d'intervalles dans les règles (simplification implicite engendrée par l'opérateur de croisement), ainsi qu'une perte d'information : en effet, il est parfois important pour la génération suivante, de conserver deux intervalles distincts. Par exemple, supposons que la valeur 23 ne doive pas activer la règle, car elle est caractéristique d'une autre classe. Si nous partons de la règle obtenue après croisement (règle inférieure au centre de la figure 5.4), une mutation à la génération suivante peut modifier la borne 27 en 22, ce qui augmente la *fitness* de la règle (on obtiendra $[7;22] \vee [24;36] \vee [81;93]$). Par contre, si nous partons de la règle obtenue après fusion, les modifications nécessaires pour parvenir à supprimer la valeur 23 de la règle sont beaucoup plus lourdes, et celle-ci sera éliminée prématurément bien avant. De plus, il est intéressant de constater que l'effet positif ou négatif d'un intervalle sur la qualité globale de la règle peut être lié à d'autres intervalles présents dans la même règle. Lorsque l'on échange de tels intervalles, il est possible que la qualité de la règle baisse, ce qui permet de détecter cette collusion, alors que l'opération de fusion accentue la dissimulation de cette

[1] Sorte d'histogramme indiquant, pour une classe donnée, la probabilité qu'a la valeur d'un échantillon de cette classe de passer par un point donné du spectre. Les spectrogrammes apportent aussi une qualité de validation visuelle et seront décrits dans le chapitre 7.

dépendance. C'est pourquoi la correction par fusion est une opération que nous déconseillons par défaut, les expérimentations ayant montré une perte de performance [Quirin, 2002].

5.2.1.5 Opérateur de mutation

La mutation permet d'explorer l'espace de recherche. Les règles sont en fait un ensemble relativement important de variables, surtout lorsque l'on traite des images hyperspectrales. Nous avons donc choisi d'appliquer l'opérateur de mutation, avec une certaine probabilité P_{mut}, à plusieurs niveaux, c'est-à-dire que l'on va à la fois agir sur une bande complète, sur un intervalle et/ou sur une borne. Nous montrons, dans la figure 5.5, les différents *niveaux* d'opérations que nous proposons, représentées sous la forme d'un arbre.

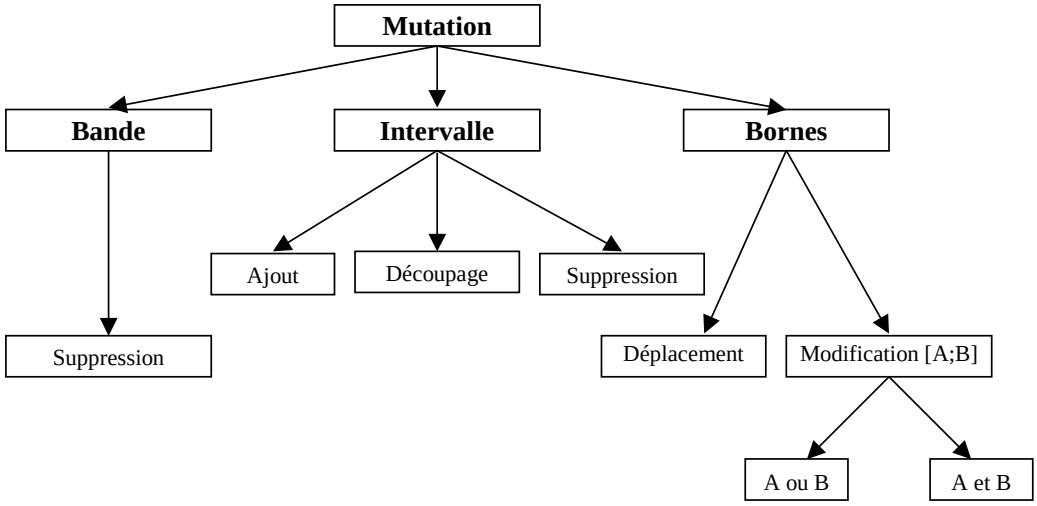


FIG. 5.5 – Opérateur de mutation (**ICU**).

La mutation ne concerne qu'un et un seul objet de la règle à la fois : soit l'on s'intéresse à la modification d'une bande complète, soit à un intervalle donné dans une bande bien précise, soit à la modification d'une ou deux bornes dans un intervalle bien précis. Ceci permet à la fonction d'évaluation de tester les modifications les unes après les autres et de privilégier le type de modification et l'effet correspondant au cas par cas. Le découpage en niveaux (bande, intervalle, bornes) est justifié par les remarques suivantes :

- La mutation de bande consiste à supprimer dans la règle les contraintes liées à certaines bandes sélectionnées. Son intérêt peut être résumé en deux points essentiels : tout d'abord ce type de mutation permet la généralisation et la simplification des règles. Ensuite, l'opérateur permet d'éliminer les bandes bruitées, ainsi que toute bande non bruitée et non porteuse d'une information discriminante pour la classe. L'utilisateur a le moyen de connaître la liste des bandes qui ont été éliminées en observant la taille des intervalles de la base de règles finale produite par l'algorithme, ce qui lui permet d'avoir une idée plus précise sur les bandes caractéristiques d'une classe.
- La mutation d'intervalle permet d'ajouter, supprimer ou découper un intervalle en deux. En cas d'ajout, la nouvelle règle est alors complétée par un intervalle centré de manière aléatoire, et dont la largeur est prise aléatoirement autour d'un seuil paramétré par l'utilisateur, en pourcentage de la valeur maximale possible. La suppression est, elle aussi, aléatoire. Le découpage d'un intervalle se fait à partir d'un point aléatoire (par exemple, le découpage de [10 ;100] peut donner [10 ;13] et [15 ;100]). Un découpage suivi d'une suppression permet de réduire le *tube spectral* pour lequel les valeurs spectrales des pixels sont discriminantes.

– La modification ponctuelle aléatoire d'une ou de deux bornes d'un intervalle est possible, ainsi qu'une opération de glissement (ang., *shift*) aléatoire de l'intervalle. L'amplitude est paramétrable par l'utilisateur et se calcule par rapport à la valeur des bornes avant la mutation.

L'utilisateur a le moyen de paramétrer la probabilité de déclenchement des différentes branches de la figure 5.5 : les paramètres $P_{\text{mut,band}}$, $P_{\text{mut,interval}}$ et $P_{\text{mut,border}}$ interviennent respectivement sur la mutation de bande, d'intervalle et de bornes. Il peut donc agir directement sur la complexité des règles produites par le programme. Même si ces paramètres peuvent sembler nombreux, il faut garder à l'esprit qu'ils ne sont pas essentiels : qu'une règle soit simple ou complexe, l'algorithme ne conserve de toute façon que la meilleure. Enfin, les règles produites ne sont pas forcément viables (exemple : [53 ;12]), surtout après avoir réalisé de nombreuses mutations en chaîne. Elles sont donc systématiquement soumises à une procédure de validation visant à rétablir leur cohérence (inversion des bornes, fusion, seuillage lors d'un dépassement de capacité, ...).

5.2.2 L'algorithme XCS-R

5.2.2.1 Description de XCS-R

Nous avons développé un autre algorithme, **XCS-R**, qui permet lui aussi de faire de la classification *soft*. L'architecture de ce système est plus complexe que celle d'**ICU**. Le fonctionnement général de l'algorithme est basé sur le modèle du système de classifieurs à valeurs continues proposé par [Wilson, 2000a] et sa description algorithmique [Butz et Wilson, 2002]. Cette partie présente l'architecture générale du système (figure 5.6), son fonctionnement ainsi que les améliorations que nous avons apportées par rapport au modèle de Wilson. D'autres détails sont décrits dans nos publications [Quirin et al., 2004; Quirin et Korczak, 2005a; Quirin et al., 2005].

XCS-R fait évoluer par algorithme génétique un ensemble de règles, appelées dans la terminologie du chapitre 2 « *population de classifieurs* » (ou *population set*). Contrairement aux systèmes de classifieurs originaux (LCS), la fonction d'évaluation, introduite par [Wilson, 1995], est basée sur l'exactitude de la prédiction de la récompense, plutôt que sur la prédiction de la récompense elle-même. Ainsi, non seulement **XCS-R** a été conçu pour permettre l'évolution de règles de classification qui maximisent la réponse de l'environnement, mais il détermine aussi une représentation complète du problème (ang., *complete mapping*), c'est-à-dire que l'algorithme apprend la récompense qu'il va percevoir pour chaque combinaison possible de parties <*condition*> et de parties <*action*>. Comme dans notre cas, les échantillons sont composés de valeurs réelles qui peuvent être très grandes, nous avons adapté le modèle original de Wilson au problème de la classification d'images. Nous avons choisi une représentation par séquences d'intervalles de valeurs continues et adapté certains paramètres pour tenir compte de l'amplitude très large des données. La réponse de l'environnement est calculée par une fonction d'évaluation des règles identique à celle vue pour **ICU**. D'autre part, la sélection des règles fait l'objet d'une attention particulière et sera détaillée dans la section 5.2.2.3.

Un classifieur se compose de cinq composants principaux :

1. La partie <*condition*> spécifie une conjonction d'intervalles, un intervalle pour chaque attribut de l'échantillon d'entrée.
2. La partie <*action*> spécifie la classe C correspondant à l'échantillon.
3. La prédiction de récompense p estime la moyenne des récompenses que le système va percevoir après avoir déclaré que l'échantillon activant la règle est de la classe C.
4. L'erreur de prédiction ϵ est basée sur l'écart-type des prédictions par rapport aux récompenses constatées.
5. La *fitness* F reflète la performance relative moyenne mesurée du classificateur par rapport à d'autres classifieurs qui s'activeraient avec la même condition.

L'apprentissage commence avec une population vide. Pour un échantillon e donné, tous les classifieurs de [P] dont les conditions sont activées par e forment le *Match Set* [M] (voir la figure 5.6). Si les valeurs de certaines actions ne sont pas représentées dans [M], un opérateur spécial, le *Covering Operator*, est

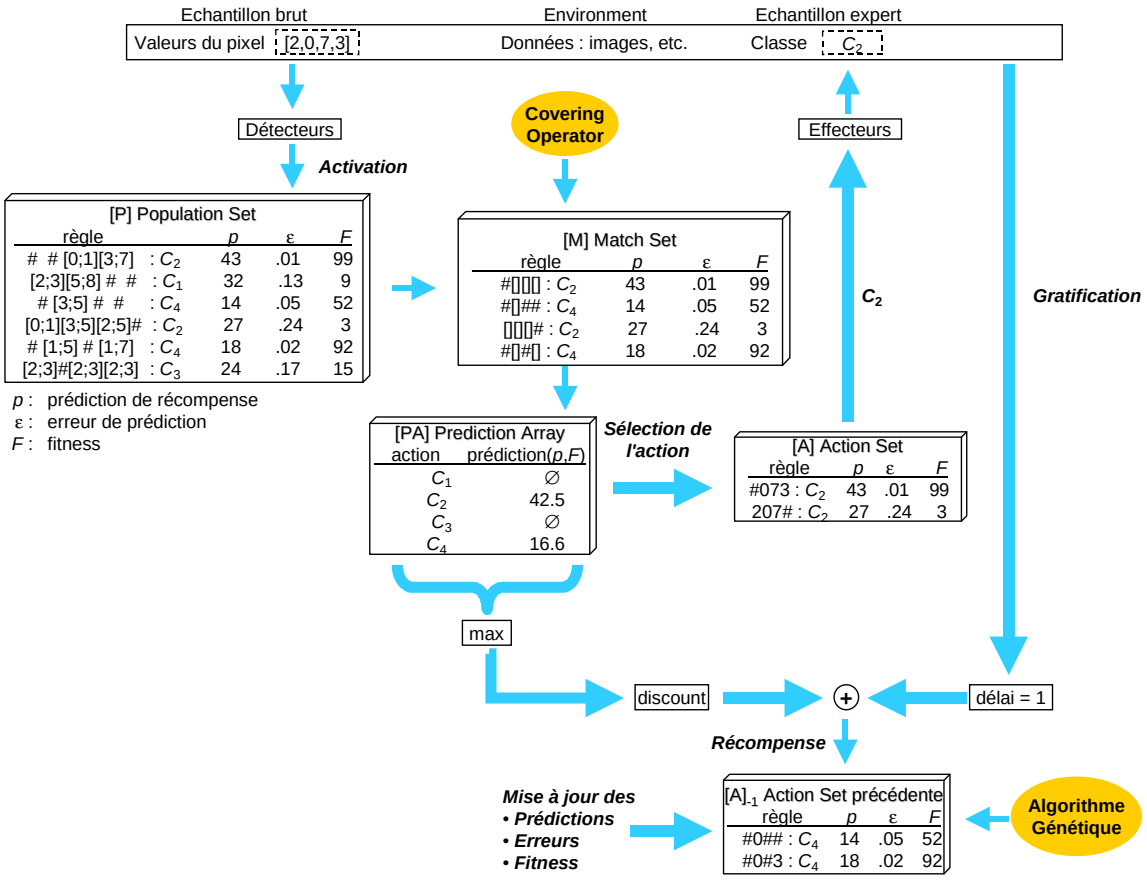


FIG. 5.6 – Architecture du système **XCS-R**.

appelé. Celui-ci crée un classifieur qui est activé par e et qui spécifie l'une des valeurs manquantes. Pour un *Match Set* donné, **XCS-R** construit une *prediction array* [PA] en estimant la récompense de chaque action possible. Basiquement, [PA] est construite à partir de la moyenne, pondérée par la *fitness*, des valeurs de toutes les prédictions de récompense des classifieurs de [M], pour chaque action possible. Durant la phase d'apprentissage, **XCS-R** choisit une action en utilisant la sélection par la roue de roulette (approche historique, [Holland, 1975; Butz et Wilson, 2002]) ou la sélection par tournoi (approche moderne, [Butz et al., 2003]). Durant la phase d'exploitation, l'action a_{max} ayant la plus grande valeur de prédiction $P(a_{max})$ est choisie.

5.2.2.2 Principe de l'apprentissage par renforcement

XCS-R met à jour la population de règles après chaque instance d'échantillon qui parcourt le système. Lorsque l'action choisie est comparée à l'échantillon expert, une gratification ou pénalisation est perçue et les paramètres de tous les classifieurs de l'*Action Set* courant [A] (tous les classifieurs de [M] qui spécifient l'action finalement choisie par **XCS-R**) sont mis à jour en conséquence.

Le *Covering Operator* s'assure que toutes les valeurs pour les actions susceptibles d'être déclenchées par un échantillon donné sont représentées par au moins un classifieur. Lors de son appel, chaque attribut du nouveau classifieur créé est initialisé en utilisant un paramètre nommé *cover-rand* qui spécifie l'intervalle maximal autorisé pour un attribut donné.

Ensuite, l'AG est appelé régulièrement, en fonction d'une probabilité de déclenchement θ_{GA}, consistant à croiser ou à muter des classifieurs de [P], puis un nouveau cycle débute. Lors de l'insertion d'un classifieur *enfant* (ang., *offspring*) dans la population [P], les classifieurs existants lui sont comparés. Si un classifieur plus général est trouvé, c'est-à-dire qui *englobe*[2] l'enfant, ce dernier est écarté et la *numerosity* (ang.) du classifieur général est augmenté. Le paramètre de *numérosité*[3] indique que le classifieur est présent virtuellement plusieurs fois dans la population (il est nommé *macro-classifieur*) et ne sert qu'à réduire l'occupation mémoire pour le traitement de grands jeux de données.

5.2.2.3 Sélection des règles

Comme dans **ICU**, le même échantillon peut activer plusieurs règles qui peuvent spécifier des classes différentes. En classification *soft*, ce comportement est attendu. En classification *hard*, nous utilisons directement les paramètres des classifieurs pour faire un choix unique. Nous avons proposé deux méthodes (*MaxConfident* et *ScoreConfident*) dans ce but :

1. Dans la première, seules les règles qui ont une valeur de prédiction de récompense élevée sont éligibles. Le seuil pour cette valeur correspond à la moitié de la gratification maximale que l'environnement peut verser. Ensuite, la classe la plus fréquente présente dans l'*Action Set* [A] est renvoyée.
2. La seconde est basée sur l'idée que les classifieurs intéressants ont une valeur de prédiction élevée mais n'ont pas été créés récemment. Un mécanisme existe dans **XCS-R** pour donner aux classifieurs nouveaux-nés une prédiction moyenne mais déjà forte, malgré leur *inexpérience* due à leur jeune âge, afin d'éviter qu'ils ne se fassent éliminer au tour suivant. Nous calculons alors une note qui tient aussi compte de la *fitness*. Pour une classe c et un classifieur r donnés, cette note est calculée de la façon suivante :

$$S_r = \frac{\sum P_r * F_r}{\sum F_r} \tag{5.7}$$

où P_r est la valeur de prédiction de la règle et F_r sa *fitness*. La classe possédant la meilleure note est renvoyée.

La seconde méthode donne généralement de meilleurs résultats que la première, car le fait de tenir compte de la *fitness* de manière progressive par rapport à la capacité de prédiction de récompense des règles est favorable aux règles performantes qui n'ont pas eu le temps de se développer [Quirin et al., 2005]. Ces deux méthodes ne sont utiles que lors de la phase d'exploitation d'une population de règles apprises en classification de type *hard*, ce qui correspond toujours à l'introduction d'un biais défavorable car la technique d'apprentissage de **XCS-R**, qui interagit avec un environnement externe et une présentation des échantillons tour à tour, est plus proche d'un apprentissage de type *soft*.

5.2.3 Synthèse

La structure des règles d'**ICU** et de **XCS-R** est finalement relativement basique :

si <*condition*> **alors**<*classe*>

Elle est néanmoins compacte et reste un bon moyen de capturer la complexité des données tout en restant lisible. À titre d'illustration, nous donnons un exemple d'une règle produite par **ICU** pour classifier les instances d'une classe de végétation marine (le *Juncus*) :

$(435 \leq B_0 \leq 1647) \wedge \ldots \wedge ((365 \leq B_{74} \leq 4023) \vee (15643 \leq B_{74} \leq 48409)) \wedge \ldots \wedge (668 \leq B_{79} \leq 4413) \Rightarrow$ [CLASS Juncus]

[2]Ce terme, n'ayant pas d'équivalents consensuels dans la littérature française sur les LCS, correspond à une opération ensembliste : par exemple, l'intervalle [3 ;13] *englobe* [5 ;7].

[3]Il s'agit d'un simple entier dans les implémentations actuelles.

où B_i est la valeur de radiance pour la bande i du pixel considéré.

Cette règle de **ICU** comprend 80 conjonctions dont une disjonction concernant la bande 74 mais elle reste plutôt expressive. Sur le même exemple, l'algorithme **XCS-R** nécessite 2300 règles comprenant 80 conjonctions chacune pour atteindre la même performance. En fait, un grand nombre de ces règles sont identiques, mais l'algorithme **XCS-R** nécessite d'être paramétré à 2300 règles pour des raisons de convergence. Une comparaison plus détaillée entre la représentation des règles de **ICU** et de **XCS-R** est fournie dans le chapitre 7 ainsi que l'étude de la performance de ces règles. Concernant la taille importante des bases de règles produites par **XCS-R**, le chapitre suivant se propose d'étudier des techniques de simplification. Nous nous intéressons, dans l'immédiat, à deux représentations adaptées à la classification floue.

5.3 Méthodes d'apprentissage adaptées à la classification floue

Outre le problème de la complexité de la base de règles, l'exploitation d'images satellites à haute résolution spatiale et spectrale pose un autre problème : comment arriver à traiter la grande quantité de données disponibles (voir la section 3.4.1), tout en exploitant les mesures très précises des thématiciens experts ? Par exemple, à cause de la faible résolution[4], en zone rurale comme urbaine, un même pixel peut contenir différentes sortes de végétation souvent en relation symbiotique les unes par rapport aux autres (*mixel*). Les pourcentages de composition de certains échantillons sont connus et peuvent être utilisés pour détailler le contenu des *mixels* et passer de la résolution du problème de la classification *hard* à celui de la classification floue (*unmixing*). L'estimation de la proportion relative des différents constituants des types de terrain présents dans chaque pixel permettrait d'obtenir des modèles de *mixels* plus précis.

Dans cette section, nous présentons deux de nos algorithmes adaptés aux problèmes de classification de type floue. Le premier, **ICUX** [Quirin et Korczak, 2005b], est une extension de l'algorithme **ICU**, présenté dans la section 5.2.1, et permet de produire des règles plates, mais permettant d'intégrer une expertise de type classification par intervalles flous, ce qui permet à l'expert de définir des intervalles de confiance pour les pourcentages de composition des spectres des classes. Le second, **GramGen**, est un algorithme à base de programmation génétique guidée par une grammaire, avec un ensemble d'opérateurs suffisamment vaste pour autoriser une grande souplesse à l'expert lors de la formalisation de ses expressions. Un troisième algorithme a été développé, **ProgGen**, qui possède les mêmes caractéristiques que **GramGen**, sauf pour la construction des individus génétiques, qui fait appel à des contraintes par probabilités plutôt que par grammaire. Comme cet algorithme est très proche de **GramGen** nous ne le présenterons pas dans une section isolée, mais nous montrerons leurs différences lorsque cela sera pertinent.

5.3.1 L'algorithme ICUX

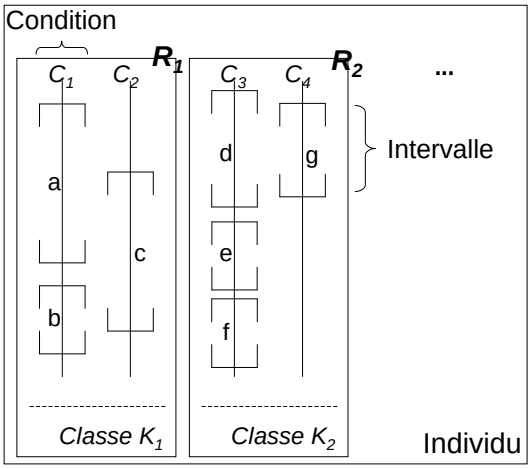
5.3.1.1 Introduction

La recherche d'une solution au problème de l'*unmixing* passe par la définition puis l'adaptation au problème posé d'un classifieur génétique prenant comme paramètre les valeurs de radiance d'un pixel (échantillon brut), et renvoyant en sortie, pour chacune des classes apprises *a priori*, la proportion de ces classes dans cet échantillon. La représentation choisie pour ce classifieur permet d'encoder, sous la forme de contraintes, la plage de valeurs admissibles du spectre pour chacune des classes. Ces contraintes sont en fait des conjonctions de disjonctions d'intervalles, soit des hyper-rectangles dans l'espace des données. Cette représentation permettrait à la connaissance apprise d'être facilement représentable à des fins de présentation et de validation à un expert humain. La présence de disjonctions dans les contraintes permet de résoudre des problèmes non linéaires. Enfin, ces règles sont évaluables directement sur les données, sans nécessiter de pré-traitements préalables de l'ensemble de l'image, elles sont donc rapides pour classer de nouvelles images.

[4] En fait, le problème se pose aussi avec une résolution élevée, car des petits éléments perturbants peuvent apparaître (buissons, voitures, piétons, ...).

5.3.1.2 Représentation d'un classifieur

Contrairement à **ICU**, un classifieur (individu génétique) est composé d'autant de règles qu'il y a de classes à traiter dans les données expertes. La représentation d'**ICUX** suit donc une approche de Pittsburgh, plutôt que de Michigan (voir la section 2.4). En effet, la classification floue se calculant par la position de toutes les règles dans l'espace des données, il faut faire évoluer ces règles en parallèle. L'apprentissage se fait donc en parallèle pour toutes les classes. Chaque règle contient autant de conditions qu'il y a de bandes dans l'image à traiter ou d'attributs exogènes pour chaque échantillon (voir la figure 5.7).



R$_1$: **si** $P \in (a \vee b) \wedge c$ **alors** prop. de la classe K_1 dans P **est** 100%
R$_2$: **si** $P \in (d \vee e \vee f) \wedge g$ **alors** prop. de la classe K_2 dans P **est** 100%
.....

FIG. 5.7 – Figure de gauche : représentation génotypique d'un individu génétique de **ICUX** (comprenant plusieurs règles R_i). Figure de droite : représentation sémantique associée à l'individu.

Les conditions sont liées entre elles par des conjonctions, par exemple $(a \vee b) \wedge c$. Chaque condition porte un nombre variable d'intervalles, dont le domaine de valeur et le type (réel, entier, booléen) dépendent des données à traiter. Les intervalles sont liés entre eux par des disjonctions. Chaque règle représente donc une contrainte sous la forme d'un hyper-rectangle pour une classe donnée. Une donnée est dite en "*matching parfait*" si l'ensemble des valeurs respectent la conjonction de disjonctions d'intervalles pour une classe donnée et se trouve hors des intervalles pour les autres classes. Par exemple, s'il y a 5 classes à apprendre, un *matching parfait* pour la classe 2 pourra être représenté par une décision du type $\{0, 1, 0, 0, 0\}$, comprenant une valeur maximale pour la classe en question et des valeurs minimales pour les autres classes. Pour permettre aux individus de décrire des concentrations de classes pures diverses sous forme de pourcentages continus, un calcul de corrélation est effectué lorsqu'une donnée ne respecte pas les intervalles d'une règle donnée. Le rôle de l'algorithme évolutionnaire est d'adapter la taille et la position de ces hyper-rectangles afin qu'ils se conforment au modèle de concentration fixé par l'expert. Les parties conditions des règles évoluant de manière indépendantes, il est possible pour l'individu d'affecter une concentration nulle ou maximale à toutes les classes à la fois (le cas échéant), ou bien de découvrir des classes qui se recouvrent partiellement ou non dans les données (attributs corrélés). La section suivante présente les techniques que nous avons utilisées pour quantifier le *matching* d'une règle et donc d'un individu pour un pixel donné.

5.3.1.3 Activation d'une règle

La fonction de *matching* $\mathcal{M}(R, P) = r$ est définie pour chacune des règles R composant un individu et renvoie un réel compris entre 0 et 1 correspondant à la proportion r de classe pure contenue dans le pixel $P = [p_0, \ldots, p_i, \ldots, p_n]$ (n est le nombre de bandes). L'application d'un individu sur un pixel consiste à appliquer les différentes règles contenues dans cet individu et à recueillir les différentes proportions pour

chacune des classes. Pour chaque intervalle $I_i = [a_i; b_i]$ $(0 \leq i \leq n)$ de la règle R, la fonction $\mathcal{M}(R, P)$ fait appel à une fonction $\mathcal{M}(I_i, p_i)$ qui peut être quelconque, mais qui prend son maximum dans l'intervalle $[a_i; b_i]$. Lors d'une disjonction d'intervalles, la valeur la plus grande de $\mathcal{M}$ est retenue.

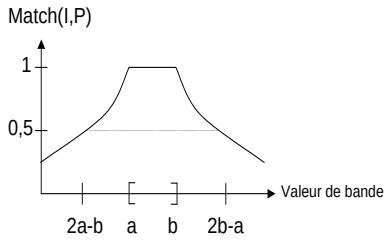


FIG. 5.8 – Représentation d'une fonction satisfaisante de *matching* $\mathcal{M}(I_i, p_i)$ pour un intervalle $[a; b]$ donné.

La fonction qui a été retenue en expérimentation est celle présentée sur la figure 5.8. Elle est définie de la manière suivante :

$$\mathcal{M}(I_i, p_i) = \begin{cases} 1 & \text{si } p_i \in [a_i; b_i] \\ \frac{b_i - a_i}{b_i - p_i} & \text{si } p_i < a_i \\ \frac{b_i - a_i}{p_i - a_i} & \text{si } p_i > b_i \end{cases} \tag{5.8}$$

où p_i est la valeur du pixel P pour la bande i et $[a_i; b_i]$ les valeurs de l'intervalle correspondant défini dans l'individu génétique. $\mathcal{M}$ vaut 1 pour un *matching* parfait et tend vers 0 sinon.

Cette fonction a la particularité d'être uniquement dépendante de la taille des intervalles, donc indépendante de l'échelle des données. L'algorithme évolutionnaire devra adapter la position et la taille des intervalles directement en fonction du contexte spectral analysé. La fonction est constante dans l'intervalle $[a_i; b_i]$ pour ne pas influencer la performance des règles si la répartition du spectre n'est pas homogène (hypothèse vérifiée).

La fonction de *matching* pour une règle complète R tient compte de la note obtenue pour chacune des bandes et se défini de la manière suivante :

$$\mathcal{M}(R, P) = \sqrt[\epsilon]{\frac{\sum_i^n \mathcal{M}(I_i, p_i)^\epsilon}{n}} \tag{5.9}$$

où n est le nombre de bandes et ϵ est un paramètre positif non nul contrôlant l'influence des valeurs extrêmes.

Le *matching* d'un pixel P avec un individu complet I est défini comme étant le vecteur normalisé des notes obtenues pour chacune des règles de l'individu. Chaque composante du vecteur indique la proportion de la classe correspondante dans le pixel P :

$$\mathcal{M}(I, P) = \left(\frac{\mathcal{M}(R_1, P)}{\sum_i \mathcal{M}(R_i, P)}, \cdots, \frac{\mathcal{M}(R_k, P)}{\sum_i \mathcal{M}(R_i, P)}, \cdots, \frac{\mathcal{M}(R_n, P)}{\sum_i \mathcal{M}(R_i, P)} \right) \tag{5.10}$$

Cette fonction sert à quantifier la distance entre les valeurs de l'individu I et du pixel P. Le résultat obtenu est à la fois utilisé dans la fonction d'évaluation pour noter l'individu et en exploitation, pour calculer les proportions estimées de chaque classe dans le pixel.

5.3.1.4 Vérification des conditions

Lorsque les règles sont créées ou ont été modifiées (par les opérateurs de croisement ou de mutation), il est possible que les individus ne soient plus cohérents. Une modification aléatoire de l'une des bornes d'un intervalle peut conduire :

- à rendre une borne MIN supérieure à une borne MAX,
- à rendre la valeur d'une borne inférieure ou supérieure à la plage admissible pour les valeurs des pixels,
- à produire des intervalles auto-intersectant (ex : [25 ;105] $\wedge$ [80 ;112] est équivalent à [25 ;112] et [12 ;13] $\wedge$ [14 ;18] à [12 ;18]).

ICUX contient une procédure pour vérifier que de tels cas ne se produisent pas, et pour les corriger le cas échéant par une fusion ou une inversion des bornes. Nous l'appellerons dans la suite *procédure de fusion*.

5.3.1.5 Opérateur d'initialisation

La procédure d'initialisation du pool génétique est construite à partir de l'image brute et de l'expertise (*mixels*). Un individu initial est créé puis est utilisé pour constituer le pool complet. Notamment, le nombre de disjonctions d'intervalles pour chacune des bandes et chacune des classes est estimé en tenant compte à la fois de l'image et de l'expertise.

Soit n le nombre de classes pures et m le nombre de bandes dans l'image, ou d'attributs dans les données à analyser. À chaque pixel P correspond une expertise $c_1, \ldots, c_n \in \mathbb{R}$ pour chacune des classes pures $K_1, \ldots, K_n$. La valeur de l'expertise c_i de ce pixel pour une classe pure K_i est vue comme la contribution de la valeur p_j de la bande j du pixel P pour cette classe K_i. Elle s'exprime par $h(p_j, j) = c_n$. Par exemple, pour un pixel $P = [1000; 1500; 1300]$ et une expertise $C = [0.7; 0.1; 0.05; 0.03; 0.12]$, nous considérons que la contribution de la valeur '1000' pour la première bande et la première classe est de 70%.

Si on se fixe une classe K_i, une bande j et une disjonction d'intervalles $DI = [a_1; b_1] \vee \ldots \vee [a_m; b_m]$, on peut calculer la contribution moyenne $\mu_1(i, j, DI)$ apportée par toutes les instances de pixels qui *matchent* DI et la contribution moyenne $\mu_0(i, j, DI)$ apportée par toutes les instances de pixels qui *ne matchent pas* DI.

L'algorithme se fixe alors un pas de recherche sur l'intervalle $[MIN; MAX]$ des valeurs constatées des pixels pour une bande donnée, et examine de manière exhaustive l'ensemble des disjonctions qu'il est possible de construire. Par exemple, si, pour une bande j, les valeurs des pixels de l'image se répartissent dans l'intervalle [1 ;1000] et qu'on s'est choisi un pas P_e découpant l'espace de recherche en 2 parties égales, on va examiner successivement les disjonctions '00', '01', '10' et '11', c'est-à-dire respectivement les disjonctions : "$\emptyset$" (vide), "$[501; 1000]$", "$[1; 500]$" et "$[1; 500] \vee [501; 1000]$" (qui se réduit en $[1; 1000]$ après l'application de la *procédure de fusion*). Pour chacune de ces disjonctions, on calcule la contribution μ_1 apportée par les valeurs *dans* ces intervalles et celle (μ_0) obtenue *hors* de ces intervalles. L'algorithme n'a plus qu'a sélectionner la disjonction provoquant l'écart de contribution maximal $\varepsilon = \mu_1 - \mu_0$, ce qui permet de s'assurer que la contribution d'une bande donnée pour une classe donnée est maximale dans la disjonction et minimale à l'extérieur.

Un individu dit "graine initiale" est construit en accumulant les disjonctions gagnantes pour chacune des bandes et pour chaque classe, selon la représentation présentée plus haut. Le pool initial est ensuite obtenu en dupliquant cette graine et en la bruitant légèrement autant de fois qu'il y a d'individus dans la population initiale.

Ces contributions sont dépendantes de la classe et de la bande analysée. La règle finale (pour une classe donnée) est constituée d'une conjonction de disjonctions obtenues pour chacune des bandes à analyser. En pratique, pour examiner exhaustivement un espace de recherche découpé en $P_e = 20$ parties égales (voir la figure 5.9), il faut 2^{20} soit un million d'itérations, ce qui représente un temps de calcul de moins d'une seconde sur une machine récente. De plus, seul la moitié des itérations est réellement effectuée, car en inversant μ_0 et μ_1, on déduit l'importance des disjonctions complémentaires (par exemple, la disjonction '10' se substituant alors à la disjonction '01'). Le nombre maximal d'intervalles d'une disjonction est borné par le découpage de l'espace de recherche (soit 10 pour 20 parties, car nous fusionnons les intervalles voisins).

L'une des particularités de notre algorithme est le fait qu'il puisse produire un nombre d'*interstices* variables (au sens du nombre d'intervalles dans une disjonction), en tenant compte à la fois des données (de

Bande	Disjonction	μ_1	μ_0
b = 0	[.....x.....]	0.0677	0.0000
b = 1	[.....x.....]	0.0619	0.0000
b = 2	[.....x.....]	0.0507	0.0000
b = 3	[.....x.....]	0.0547	0.0000
b = 4	[.....x....]	0.0700	0.0000
b = 5	[.....x.xx...]	0.0868	0.0000
b = 6	[.....xxx..]	0.0951	0.0000
b = 7	[.....xxxx.]	0.0996	0.0000
b = 8	[.....xxxxx.]	0.1650	0.0963
b = 9	[.....xxxxx.]	0.0951	0.0000
b = 10	[.....xxxxx.]	0.0949	0.0000
b = 11	[.....xxxxx.]	0.0939	0.0000
b = 12	[.....xxxxx..]	0.0939	0.0000
b = 13	[.....xxx.]	0.2200	0.0633
b = 14	[.....x]	0.3300	0.0382
b = 15	[.....x.....]	0.3300	0.0581
b = 16	[.....xx.....]	0.0563	0.0100
b = 17	[.....x.x.....]	0.0504	0.0000
b = 18	[.....x.....]	0.0479	0.0000
b = 19	[.....x.....]	0.0502	0.0000

FIG. 5.9 – Exemple des disjonctions trouvées pour la classe "*Spartina*" sur une image multispectrale de 20 bandes. Les "X" représentent les intervalles sélectionnés dans la disjonction. On voit apparaître verticalement le spectre de la classe. Notons les *interstices spectraux* visibles pour les bandes 5 et 17.

toutes les bandes) et des valeurs continues de l'expertise. Cependant, malgré la qualité de l'initialisation, l'affinage par l'algorithme évolutionnaire est nécessaire pour plusieurs raisons. Tout d'abord, à cause de la partialité introduite par la valeur fixée (par l'utilisateur) du pas de recherche. Ensuite, à cause de la répartition non homogène des valeurs des attributs au sein de l'espace de recherche.

La partie suivante présente les opérateurs génétiques qui ont été étudiés pour affiner la valeur des bornes des intervalles en dehors du cadre rigide fixé par le pas de recherche, ainsi que les découper, si la résolution choisie pour le pas était trop faible.

5.3.1.6 Opérateurs de croisement et de mutation

L'opérateur de croisement. Dans **ICUX**, le croisement est uniforme. Une classe K est sélectionnée au hasard, ainsi que les règles $R_{1,K}$ et $R_{2,K}$ correspondantes dans les deux individus à croiser. Pour chaque bande j, la condition $C_{1,K,j}$ de $R_{1,K}$, ainsi que la condition $C_{2,K,j}$ de $R_{2,K}$ ont une probabilité P_{cross} d'être croisées. Lorsque le croisement est sélectionné, un intervalle est choisi aléatoirement dans chacune des conditions et il est échangé. Ensuite, la *procédure de fusion* est appliquée sur les règles modifiées.

L'opérateur de mutation. Cet opérateur est appliqué sur la population en fonction d'une probabilité P_{mut} donnée. Il est présent à trois niveaux : au niveau d'une condition complète, d'un intervalle ou des bornes d'un intervalle.

La mutation d'une condition complète consiste à supprimer, pour une règle donnée, la condition correspondant à l'une des bandes avec une probabilité $P_{\text{mut,cond}}$. Plusieurs intérêts : la simplification de la règle, la généralisation, et l'épuration (l'algorithme considère cette bande comme bruitée ou pauvre en informations).

Le second type de mutation consiste à sélectionner avec une probabilité $P_{\text{mut,int}}$ un intervalle dans l'une des conditions C puis à l'éliminer, à le couper en deux, à le remplacer si le nombre d'intervalles autorisés pour cette règle particulière a été atteint, ou à ajouter un intervalle dans C. Le découpage d'un

intervalle consiste à choisir un point de coupe aléatoirement puis à les séparer si leur taille est suffisamment large (par exemple, $[10;100]$ est découpé en deux intervalles non consécutifs $[10;15] \wedge [17;100]$ mais $[1;3]$ est découpé en $[1;2] \wedge [2;3]$). Son intérêt est d'explorer les valeurs continues du spectre en-dessous de la résolution du pas de recherche de l'opérateur d'initialisation.

Enfin, la mutation des bornes d'un intervalle avec une probabilité $P_{\text{mut,borne}}$ modifie les deux bornes par glissement (*shift*), élargissement ou modification d'une seule des deux bornes de la façon suivante. Soit ω_1 et ω_2 deux variables aléatoires réelles dans l'intervalle $[0;1]$. Soit α la quantité de bruit ajouté, β l'augmentation de la taille des intervalles et γ le glissement des intervalles. Un intervalle $[a;b]$ est modifié aléatoirement en $[a''';b''']$ comme suit :

(ajout de bruit)

$$l = \alpha(b-a) \quad a' = (a-l) + 2\omega_1 l \quad b' = (b-l) + 2\omega_2 l \tag{5.11}$$

(augmentation ou réduction centrée de la taille des intervalles)

$$a'' = \frac{(a'+b')}{2} - \frac{\beta(b'-a')}{2} \quad b'' = \frac{(a'+b')}{2} + \frac{\beta(b'-a')}{2} \tag{5.12}$$

(glissement à gauche ou à droite)

$$a''' = a'' + \gamma(b''-a'') \quad b''' = b'' + \gamma(b''-a'') \tag{5.13}$$

Cet opérateur permet d'affiner pas à pas la contrainte spectrale découverte globalement par les autres opérateurs pour une classe donnée. Les règles modifiées sont toutes systématiquement validées par la *procédure de fusion*. Dans notre système, les probabilités affectées à chaque opérateur sont dynamiquement adaptées (nous les avons nommés *opérateurs à taux progressif*), en fonction de l'écart-type de la mesure de performance (*fitness*) observée sur l'ensemble des individus. Si l'écart-type décroît, la population devient monotone et un individu émergeant, peut être un minimal local, risque de coloniser le reste de la population. Dans ce cas, le taux de mutation est progressivement remonté ($P_{\text{mut}} = P_{\text{mut}} + (1 - P_{\text{mut}}) * T_{\text{up}}$, soit une fonction croissante, bornée par 1). À l'inverse, si l'écart-type est trop important, le taux de mutation est sans doute trop élevé, et il est diminué à chaque itération ($P_{\text{mut}} = P_{\text{mut}} - P_{\text{mut}} * T_{\text{up}}$, soit une fonction décroissante strictement positive). À l'utilisation, dans les problèmes de télédétection que nous avons testés, le taux se stabilise autour de 16% pour une initialisation à 30%.

L'intérêt de l'opérateur de mutation est le parcours non-linéaire de l'espace de recherche. Ainsi, un glissement (*shift*) de 10% à gauche des valeurs d'un intervalle est obtenu avec le même effort (un appel de l'opérateur) que l'ajout d'un intervalle dans une disjonction particulière, mais qui peut représenter une valeur de la fonction d'évaluation plus élevée pour la règle en question.

5.3.1.7 Fonctions d'évaluation et de sélection

Comme dans les autres algorithmes que nous avons vus, la fonction *fitness* définit la performance d'un individu particulier par rapport au reste de la population. Un individu I est évalué en calculant la corrélation entre le vecteur obtenu par $\mathcal{M}(I,P)$ et les informations expertes correspondantes pour un échantillon donné, puis en calculant la moyenne pour tous les échantillons disponibles dans le jeu d'apprentissage. La corrélation est définie comme la racine carrée de la moyenne des carrés des éloignements des proportions trouvées par rapport à l'expert. Les échantillons experts comprennent soit les proportions attendues pour chacune des classes (classifieur flou), soit des intervalles de proportions (classifieur à intervalles flous). Dans ce cas, la corrélation est établie entre les intervalles spectraux spécifiés par la règle et les intervalles experts.

De nombreuses stratégies de sélection ont été testées et sont disponibles pour l'utilisateur [Blickle et Thiele, 1995]. Différentes valeurs pour des paramètres comme le taux de croisement et de mutation, le nombre d'individus dans la population initiale et le nombre de générations ont été optimisées, et celles retenues pour les études de cas seront présentées dans le chapitre 7.

5.3.2 L'algorithme GramGen

5.3.2.1 Intérêts de la programmation génétique

La représentation offerte par le paradigme de la programmation génétique offre plusieurs avantages qui nous intéressent ici en classification d'images :

1. La représentation des règles de classification par des arbres syntaxiques est *a priori* un bon choix pour une lisibilité intuitive et une validation éventuelle par un expert humain.
2. La possibilité de définir des opérateurs terminaux personnalisés, adaptés au problème à résoudre.
3. Le traitement par l'algorithmique évolutionnaire laisse supposer une résistance accrue aux données bruitées et aux minima locaux très présents en télédétection.
4. La possibilité de définir soi-même les opérateurs génétiques permet une interaction plus approfondie avec l'expert humain (conditions d'arrêt spécifiques au domaine d'étude, fonction d'évaluation pouvant intervenir sur les nombreux paramètres des arbres produits : taille, profondeur, équilibrage, ...).
5. Enfin, le formalisme des solutions permet à la fois de résoudre des problèmes de classification comme de régression en télédétection.

Ces avantages permettent de résoudre efficacement des problèmes de classification d'images tout en produisant des classifieurs compréhensibles. Des traitements comme le calcul arithmétique entre bandes spectrales ont souvent été utilisés comme pré-traitement en vue de séparer certaines classes qui perturbent la classification d'images (par exemple, l'eau qui peut avoir une réponse hétérogène dans certains cas), comme moyen direct de classification ou comme moyen de construire de nouveaux canaux servant à reconstruire une image plus riche compensant éventuellement les informations manquantes.

Une application intéressante de la programmation génétique peut donc être la découverte d'index comme l'indice de végétation normalisé[5] ou l'indice de brillance[6], comme nous le montrerons dans le chapitre 7. La découverte automatique de tels index permettrait de proposer à l'expert humain de nouvelles formules, soit pour améliorer la performance des index existants, soit pour en proposer de nouveaux, mieux ciblés en fonction des classes à trouver.

Ce paradigme a donc été étudié sur un jeu d'images afin de retrouver des index connus. Commençons par rappeler, pour la suite, un certain nombre de termes utilisés dans ce domaine [Koza, 1992] :

- **Symbole non terminal**. Symbole de la grammaire ne servant que pour le support des dérivations, uniquement présent dans les arbres génotypiques.
- **Symbole terminal**, **opérateur terminal** ou **opérateur de calcul**. Symbole présent à la fois dans un arbre génotypique comme phénotypique. Dans les algorithmes présentés dans les sections suivantes, le terme ne désigne pas la valeur d'un nœud précis, mais plutôt le mot-clé correspondant à ce nœud. Par exemple, opMUL, opSIN ou opCST[7] sont des symboles terminaux.
- **Opérateur de nœud** ou **fonctions**. Désigne les symboles qui sont des feuilles dans l'arbre génotypique (jamais dans l'arbre phénotypique) et qui correspondent à des fonctions d'arité $N > 0$, comme opMUL.
- **Terminal feuille**. Désigne les symboles qui sont des feuilles dans l'arbre phénotypique et qui correspondent à des fonctions d'arité 0 (constantes, variables exogènes, ...) comme par exemple opCST ou opARG[8].
- **Disjonction grammaticale**. Partie à droite d'une règle de production contenant une ou plusieurs conjonctions et correspond à un choix de dérivation.
- **Conjonction grammaticale**. Partie d'une disjonction correspondant à un symbole terminal ou non terminal.

Chacune de ces notions a été introduite et développée dans notre algorithme, soit au niveau de la grammaire, soit au niveau des opérateurs génétiques, et elles sont détaillées dans les sections suivantes.

[5] NDVI = $\frac{XS3-XS2}{XS3+XS2}$

[6] BI = $\sqrt{XS2^2 + XS3^2}$

[7] Constante instanciée.

[8] Argument instancié, c'est-à-dire pointant vers l'un des attributs d'un échantillon.

5.3.2.2 Algorithme général

Le principe général de la découverte des règles par programmation génétique a déjà été présenté dans l'algorithme A3, sauf qu'ici, les règles introduites par l'algorithme correspondent à des structures arborescentes (arbres) formées d'opérateurs de nœud et de terminaux feuille. Nous avons utilisé ce principe pour implémenter deux algorithmes distincts, le premier (**ProgGen**) présentant une approche simplifiée de la programmation génétique, le second (**GramGen**) implémentant en plus un système de contraintes sur la représentation des arbres, en utilisant une grammaire définie au préalable par l'utilisateur.

La différence entre les deux algorithmes est située au niveau de l'opérateur d'initialisation et des opérateurs de mutation. Dans le cadre de **ProgGen**, la liste des opérateurs terminaux est fixée au début de l'algorithme par l'utilisateur. Il peut associer à chacun de ces terminaux une probabilité indiquant de manière statistique la fréquence de leur présence dans les arbres engendrés. Ces probabilités influencent l'opérateur d'initialisation ainsi que les opérateurs de mutation qui en tiennent compte lors du remplacement d'un sous-arbre complet par un autre. À l'inverse, le système **GramGen** utilise une grammaire, aussi paramétrée par l'utilisateur avant de lancer l'algorithme, dont les opérateurs génétiques tiennent compte. Ainsi, lors de la phase d'initialisation, du croisement ou de la mutation, les contraintes imposées par la grammaire seront toujours vérifiées. Notamment, pour des raisons d'égalité des chances lors de la sélection et la création des individus, la démarche retenue dans chacun des cas est une démarche *active*, c'est-à-dire que les individus ne sont pas écartés s'ils ne satisfont par les exigences de l'utilisateur (probabilités ou grammaire), mais que les opérateurs ont été directement écrits pour éviter que de telles occurrences ne surviennent, dans tous les cas où cela a été possible.

Les opérateurs génétiques de **ProgGen** permettent une plus grande liberté pour les opérateurs de croisement et de mutation. Par exemple, chaque nœud terminal peut être remplacé soit par l'un des opérateurs constants (opCST, ...) soit par un argument (opARG), à cause de l'absence de grammaire. Pour ne pas alourdir les sections suivantes, seuls les opérateurs, fonctions, paramètres et remarques concernant le système à base de grammaire **GramGen** est décrit. La section suivante présente donc le formalisme retenu pour la description des règles grammaticales guidant la découverte des arbres.

5.3.2.3 Formalisme de la grammaire

Les systèmes à base de programmation génétique sont connus pour amplifier la taille des arbres lors de la recherche d'une solution fiable. Une étude présentée dans [Ross et al., 2002] présente un arbre correspondant à un indicateur minéralogique nécessitant une cinquantaine de nœuds pour être efficace. L'arbre produit ne propose pas forcément une explication claire et intuitive pour cet indicateur à une personne non experte. Une interprétation rigoureuse des individus produits est pourtant nécessaire à fin de validation. Des contraintes pour simplifier les arbres ont été largement proposées, comme par exemple celles décrites dans [Montana, 1994] consistant à typer les nœuds et n'admettre que les constructions grammaticales autorisées par ce typage. Cependant en télédétection, les données ont souvent le même format au sein d'une image donc ce genre de typage n'est pas nécessaire. Des contraintes basées sur une grammaire nous semblaient plus appropriées.

Concernant le choix de la grammaire, les systèmes à base de *Context Free Grammar* (CFG) sont parmi les plus courants [Freeman, 1998; Javed et al., 2004]. Cette grammaire est formelle, simple, générale et peut être mise sous forme normale (ce qui est rapide et efficace pour les algorithmes de *parsing*). Elle est plus connue sous sa forme dérivée, la forme de *Backus-Naur* (BNF), utilisée pour décrire les langages de programmation.

Formellement, une CFG peut être définie comme un quadruplet $G = (V_t, V_n, P, S)$ où :
- V_t est un ensemble fini de terminaux,
- V_n est un ensemble fini de non-terminaux,
- P est un ensemble fini de règles de production,
- S est un élément de V_n et représente de manière unique le symbole de départ (*start symbol*).

Les éléments de P sont de la forme $V_n \rightarrow (V_t \cup V_n)^*$.

Dans le cas de **GramGen**, le *start symbol* est dérivé en suivant les règles de production jusqu'à produire un ensemble ne contenant que des symboles terminaux. La totalité de cette dérivation est

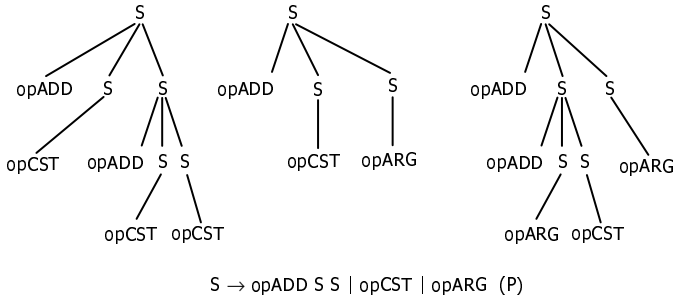


FIG. 5.10 – Arbres génotypiques pouvant être produits pour la règle de dérivation (P) dans le cadre de la syntaxe de **GramGen**.

reproduite sous forme d'arbre, désigné dans la suite par *arbre de dérivation* ou *arbre génotypique*. Ainsi, la traduction d'une règle $A \rightarrow B$ renvoie un arbre de sommet A lié à un nœud B. Une règle $A \rightarrow BC$ renvoie, quant à elle, un arbre de sommet A possédant deux fils B et C. La figure 5.10 montre quelques exemples d'une grammaire et d'arbres génotypiques associés.

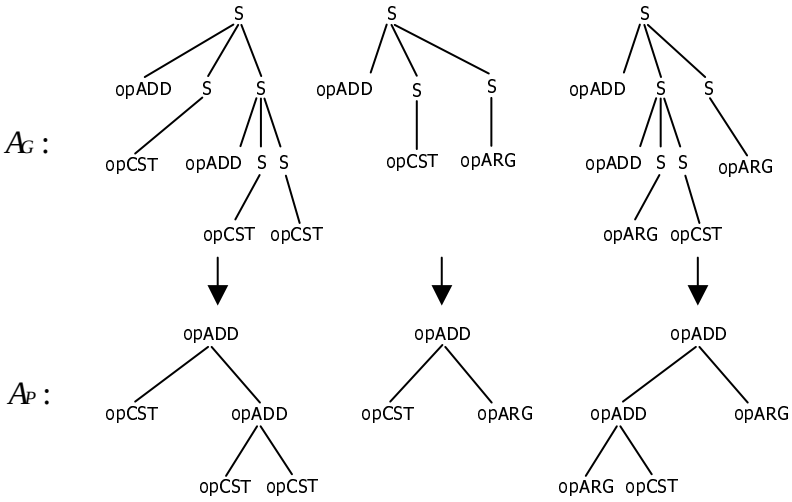


FIG. 5.11 – Conversion des arbres génotypiques en arbres phénotypiques.

À un instant donné, un symbole non terminal ne peut être dérivé (en un ou plusieurs autres symboles) qu'en utilisant une seule des règles de production, le symbole « | » séparant les disjonctions. La création des individus génétiques se fait en deux temps : la grammaire sert à définir dans un premier temps un arbre génotypique A_G, puis cet arbre est transformé en un arbre phénotypique A_P correspondant à la formule de calcul proprement dite. L'arbre A_P est obtenu en remplaçant tout sous-arbre génotypique A'_G contenant un sommet X et $N+1$ branches par un sous-arbre phénotypique A'_P dont le sommet correspond à la première branche de A'_G (représentant une fonction d'arité N) et dont les N branches correspondent aux branches 2 à $N+1$ de l'arbre A'_G. L'arbre phénotypique complet est obtenu en effectuant tous les remplacements nécessaires. La figure 5.11 illustre une telle procédure. Dans cette procédure, la grammaire n'est plus nécessaire. La première branche de chaque nœud est considérée comme la fonction et les branches suivantes ses arguments.

5.3.2.4 Opérateurs terminaux

Les opérateurs terminaux constituent des unités atomiques pour les arbres phénotypiques produits par programmation génétique. Nous distinguons deux sortes de terminaux : les terminaux feuilles (arguments ou constantes), présentés dans le tableau 5.1, et les fonctions, présentées dans le tableau 5.2. Nos expérimentations ont nécessité l'utilisation de tous les terminaux et de la majorité des opérateurs présentés dans ces tableaux, notamment les opérateurs mathématiques, les opérateurs d'arité 3 et les opérateurs d'arité N. En plus de l'implémentation des fonctions classiques, quelques fonctions intervenant souvent dans les problèmes de régression en télédétection ont été rajoutées. Il s'agit des fonctions d'arité N, capables de s'évaluer sur un pixel entier.

Symbole	**Description**
opFIXED	Constante entière ou réelle directement spécifiée dans la grammaire (ex : 1.03E-06).
opRANGE	Plage de constantes entières ou réelles (ex : [4 ;7]).
opCST	Constante instanciée aléatoirement lors de la création de l'arbre ou du nœud. Sa valeur ne changera qu'à la prochaine mutation génétique.
opARG	Argument de la fonction. Chaque échantillon d'apprentissage étant représenté par un vecteur de taille connue, chaque argument est instancié par l'une des valeurs de ce vecteur au moment de la création du nœud. Cette instanciation est ensuite modifiée au cours d'une mutation génétique.
opTRUE, opFALSE, opPI, opE, opPINF ($+\infty$), opMINF ($-\infty$), cstDIVIZERO (représente une division par zéro), cstOVERFLOW (représente un dépassement de capacité comme $log(0)$)	Constantes diverses.

TAB. 5.1 – Liste des terminaux feuilles accessibles à l'utilisateur pour la représentation des arbres.

L'utilisateur a le choix de l'ensemble des terminaux à utiliser dans les programmes génétiques, soit sous forme de probabilités dans les arbres créés avec **ProgGen**, soit dans les règles de production avec **GramGen**.

Les opérateurs de type opPUSH/opPOP sont des opérateurs spéciaux que nous avons mis au point car ils sont utiles en télédétection. Ils facilitent la gestion des files FIFO (ang., *First In, First Out*) permettant aux arbres de construire dynamiquement des tableaux dont la taille n'est connue qu'au moment de leur exécution. L'arbre étant interprété par profondeur d'abord (les nœuds fils sont toujours exécutés avant le nœud courant), il est dès lors possible pour un nœud d'effectuer des calculs sur une file contenue dans l'un de ses nœuds descendants. Les opérateurs permettant ce type de calcul sont présentés dans le tableau 5.2. La figure 5.12 montre un exemple d'une grammaire utilisant de tels opérateurs, un arbre génotypique qu'il est possible d'engendrer avec cette grammaire, l'arbre phénotypique associé et son interprétation sémantique.

Si nécessaire, des conversions inter-types ont lieu au niveau des nœuds lors du passage des valeurs (ainsi une constante strictement positive renvoyée par un opérateur mathématique est convertible en un booléen égal à *vrai*, et inversement). Enfin, chacun de ces opérateurs est associé à un paramètre indiquant s'il est commutatif ou non. Ce paramètre est utilisé lors de la comparaison structurelle de deux arbres, dont nous nous servons comme critère de diversité dans l'opérateur de terminaison.

5.3.2.5 Opérateur d'initialisation

Le rôle de l'opérateur d'initialisation des individus consiste à sélectionner les règles grammaticales ainsi que l'ordre de leur application afin de respecter un certain nombre de contraintes fixées par l'utilisateur. Ceci servira à créer la première population génétique ou les nouvelles branches réclamées par

Symbole	Description
opOPP, opINV, opCOS, opSIN, opTAN, opLOGE (logarithme népérien), opEXP, opSQRT, opABS, opCEIL, opFLOOR, opACOS, opASIN, opATAN, opCOSH, opSINH, opTANH, opACOSH, opASINH, opATANH, opLOG10, opSIGN, opFACT ($x!$), opSQ (x^2), opCUB (x^3), opLOG2, opP10 (10^x), opCURT ($\sqrt[3]{x}$), opP2 (2^x)	Opérateurs mathématiques d'arité 1.
opNOT, opCOMP1 (complément à 1), opCOMP2 (complément à 2), opSHL (*shift* gauche), opSHR (*shift* droite), opROTL (rotation gauche), opROTR (rotation droite)	Opérateurs booléens d'arité 1.
opIP (partie entière), opFP (partie flottante), opRND (arrondi entier), opARGN (sélection de l'argument N de la fonction courante)	Opérateurs divers d'arité 1.
opEMPTY (file vide), opPOP (renvoie le 1^{er} élément), opPUSH (empile le 1^{er} argument, renvoie le nombre d'objets empilés), opPSUM (somme des objets de la file), opPAVG (moyenne), opPMED (médian), opPAND (ET booléen entre tous les objets), opPOR (OU booléen), opPEQUI (*vrai* si tous les objets sont identiques, soit positifs, soit nuls, *faux* sinon)	Opérateurs d'arités diverses nécessaires à la gestion d'une file FIFO permettant de gérer des tableaux de taille dynamique. La figure 5.12 en présente un exemple.
opADD, opSUB, opMUL, opDIV, opPOW, opINF ($<$), opSUP ($>$), opINFE ($\leq$), opSUPE ($\geq$), opEGAL, opDIFF, opPRCT ($x * \frac{y}{100}$), opPRCTA ($x * (1 + \frac{y}{100})$), opPRCTS ($x * (1 - \frac{y}{100})$), opCOMB (C_y^x), opPERM (P_y^x), opAPRX ($\|x - y\| < 1E - 4$), opMOD (modulo), opQUOT (division entière), opXRT ($\sqrt[y]{x}$)	Opérateurs mathématiques d'arité 2.
opAND, opOR, opXOR, opXSHL (x est *shifté* à gauche de y bits), opXSHR (x est *shifté* à droite de y bits), opXROTL (rotation à gauche de x de y bits), opXROTR (rotation à droite de x de y bits), opIMPL (implication booléenne), opEQUI (équivalent booléen)	Opérateurs booléens d'arité 2.
opITE (si $x > 0$ alors y sinon z), opLERP (interpolation linéaire entre y et z : $x * (z - y) + y$), opINTER (appartenance à un intervalle : *vrai* si $x \in [y; z]$, *faux* sinon)	Opérateurs d'arité 3.
opSOMM (somme de tous les arguments de l'opérateur), opPROD (produit), opAVG (moyenne), opMED (median), opMIN, opMAX, opETYP (écrat-type), opVAR (variance), opSQSOM (somme des carrés), opSQAVG (moyenne des carrés), opMAND (ET booléen), opMOR (OU booléen), opMEQUI (*vrai* si on a l'équivalence des arguments), opSELECT (sélection de la valeur de l'argument N de l'opérateur)	Opérateurs d'arité N. Le comportement de ces opérateurs dépend du nombre de branches qui leur sont associées. L'utilisateur définira par la grammaire l'arité de ces opérateurs. Par exemple : $S \rightarrow$ opSOMM A B, $S \rightarrow$ opSOMM A B C, ...

TAB. 5.2 – Liste des opérateurs accessibles à l'utilisateur pour les nœuds des arbres.

l'opérateur de mutation. Si plusieurs possibilités se présentent, le choix de la règle de production à appliquer est déterminé en fonction de plusieurs critères, basés sur la taille des arbres et sur la profondeur souhaitée. Les deux phases principales consacrées à la construction des nouveaux individus sont les suivantes :

- Déterminer la hauteur $H(X)$ ou le nombre minimal de terminaux $T(X)$ qu'il est possible d'engendrer pour chaque symbole X dans le meilleur des cas.
- Lors de la construction d'un sous-arbre ou d'un arbre complet, déterminer le symbole à utiliser en fonction d'une certaine probabilité aléatoire liée à $H(X)$ ou $T(X)$.

La première phase n'est effectuée qu'une seule fois, à partir du moment où la grammaire a été paramétrée par l'utilisateur. Cette phase est décrite dans l'algorithme A4. La figure 5.13 présente une grammaire paramétrée à l'aide de l'algorithme A4. Le paramétrage représente la hauteur et le nombre minimal de symboles qu'il est possible d'obtenir au mieux dans un arbre génotypique dérivé d'un symbole donné. L'étude des valeurs maximales sont peu intéressantes pour la plupart des grammaires car elles sont infinies. L'algorithme converge même dans le cas de règles de grammaire *full-recursive* (ex : $A \rightarrow A$) dont les règles seraient alors automatiquement ignorées.

La seconde phase est effectuée lors de la création effective des individus ou à chaque fois qu'un opérateur génétique ordonne la création d'un sous-arbre. Elle est présenté dans l'algorithme A5. L'algorithme prend en paramètre une grammaire, le symbole non-terminal de départ (soit le *start symbol* S, soit un autre) ainsi que les contraintes de l'utilisateur en terme de taille et de hauteur d'arbre. Le résultat de

ALGORITHME A4
ALGORITHME D'INITIALISATION DES SYMBOLES DE LA GRAMMAIRE DANS GRAMGEN

$\rightsquigarrow$ HAUTEUR(X) - Sous-fonction de calcul de la hauteur minimale d'un symbole X
$\rightsquigarrow$ NOTE - $H(X)$ est un paramètre du symbole X

soit $h := \infty$
pour *chaque disjonction* D *de la règle* X ***faire***
  soit $hc := 0$
  pour *chaque conjonction* C *de* D ***faire***
    $hc := \text{MAX}(hc, H(C))$
  fin pour
  $h := \text{MIN}(h, hc)$
fin pour
$\text{HAUTEUR}(X) := h + 1$
————————————————————

$\rightsquigarrow$ TAILLE(X) - Sous-fonction de calcul de la taille minimale d'un arbre créé par la dérivation d'un symbole X
$\rightsquigarrow$ NOTE - $T(X)$ est un paramètre du symbole X
soit $s = \infty$
pour *chaque disjonction* D *de la règle* X ***faire***
  soit $sc = 0$
  pour *chaque conjonction* C *de* D ***faire***
    $sc = sc + T(C)$
  fin pour
  $s = \text{MIN}(s, sc)$
fin pour
$\text{TAILLE}(X) := s$
————————————————————

$\rightsquigarrow$ PARAMÈTRE - La liste des symboles $L = V_t \cup V_n$ d'une grammaire G
$\rightsquigarrow$ RÉSULTATS - La hauteur et la taille minimale de chacun des symboles de L (grammaire paramétrée)
pour *chaque symbole* X *de* L ***faire***
  si X *est terminal* ***alors***
    $H(X) = 0$
    $T(X) = 1$
  sinon
    $H(X) = \infty$
    $T(X) = \infty$
  fin si
fin pour
répéter
  pour *chaque symbole* X *non terminal de* L ***faire***
    $H(X) := \text{HAUTEUR}(X)$
    $T(X) := \text{TAILLE}(X)$
  fin pour
jusqu'à ce que *les paramètres des symboles de* L *aient convergé*
Renvoie la grammaire paramétrée

Algorithme 4: *Fonction* INITSYMBOLES

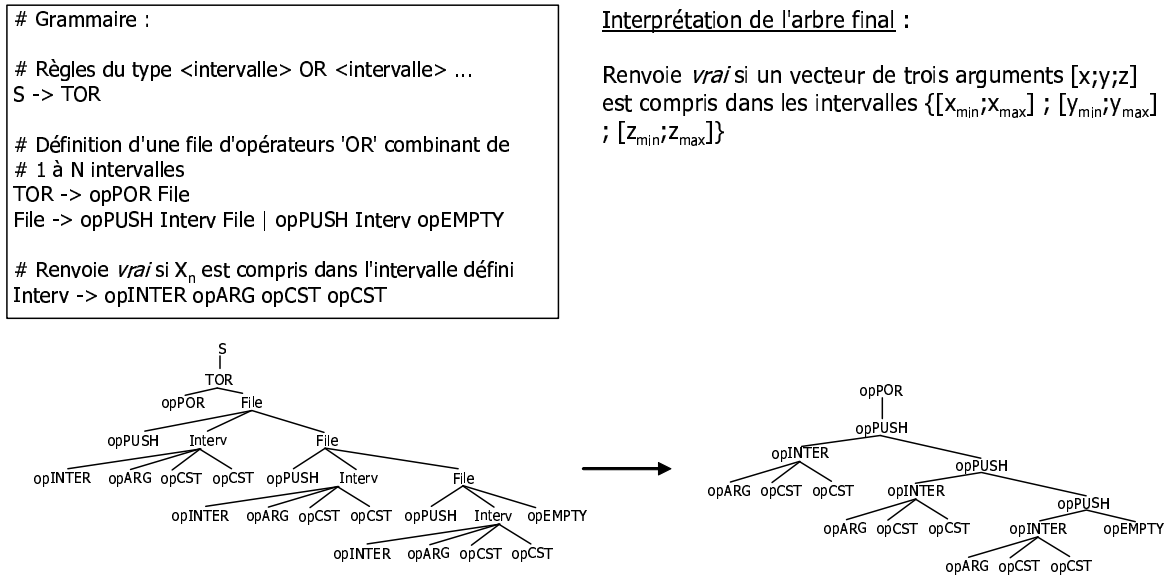


FIG. 5.12 – Grammaire utilisant l'opérateur opPUSH.

$S \rightarrow E$
$E \rightarrow OEE \mid V$
$O \rightarrow opADD \mid opMUL$
$V \rightarrow opARG \mid opCST \mid 5.34$

Symbole	H(Symbole)	T(Symbole)
S	3	1
E	2	1
O	1	1
V	1	1
opADD	0	1
opMUL	0	1
opARG	0	1
opCST	0	1
5.34	0	1

FIG. 5.13 – Exemple d'une grammaire et de son paramétrage.

l'algorithme est un arbre génotypique complet ou un sous-arbre qui peut être intégré dans un arbre plus vaste, dont le tout sera à convertir en arbre phénotypique avant l'évaluation de l'individu. L'algorithme fonctionne en contrainte exacte, c'est-à-dire que la taille spécifiée par l'utilisateur en nombre de nœuds sera toujours respectée au mieux. Si l'utilisateur choisit par exemple une taille paire N, et que la grammaire ne peut produire que des arbres dont la taille est impaire, il y a une probabilité de 0.5 qu'un individu de taille $N-1$ soit produit, de la même façon que pour un individu de taille $N+1$. Pour une valeur nulle ou négative, l'arbre produit sera le plus petit possible. Si l'utilisateur souhaite des tailles diverses, il doit choisir une plage de valeurs admissibles. Une valeur dans cette plage est alors sélectionnée de manière aléatoire puis passée en paramètre de l'algorithme. Celui-ci est appelé autant de fois qu'il est nécessaire pour constituer soit un individu, soit un pool complet. Ainsi, il est possible d'obtenir un pool dont la taille des arbres peut être spécifiée par différentes fonctions de probabilité : constantes, linéaires, gaussiennes, etc.

La détermination du symbole non terminal à dériver lorsque l'arbre de dérivation actuel en contient plusieurs, c'est-à-dire le style de dérivation retenu, n'est ni par la gauche (*leftmost derivation*), ni par la droite (*rightmost derivation*). En fait, nous avons obtenu les meilleurs résultats en choisissant à chaque fois de manière aléatoire le symbole non terminal dans l'arbre en cours de dérivation. De plus, cela garantit

une certaine diversité dans le pool, même si la grammaire est mal construite.

ALGORITHME A5
ALGORITHME DE CRÉATION D'ARBRES GÉNOTYPIQUES À PARTIR D'UNE GRAMMAIRE

$\rightsquigarrow$ PARAMÈTRES - Une grammaire paramétrée G, un symbole non terminal A de départ, la taille ou la hauteur f_{ask} souhaitée pour l'arbre
$\rightsquigarrow$ RÉSULTAT - L'individu créé
$\rightsquigarrow$ f est le critère à optimiser sur un symbole X, soit $H(X)$, soit $T(X)$
$\rightsquigarrow$ CHOIX(L) est une fonction qui choisi de manière uniformément aléatoire un élément de l'ensemble L
$\rightsquigarrow$ $\psi(n)$ est une fonction renvoyant la probabilité de sélection d'un symbole en fonction de la taille de dépassement estimée n pour le critère retenu f (différence entre la taille minimale prévue et la taille souhaitée)

soit *R un arbre de sommet A*
tant que *R contient pour feuille un symbole non terminal* ***faire***
  soit *L la liste des feuilles de l'arbre R*
  soit *$f_{min} := 0$*
  pour *chaque symbole X de L* ***faire***
    $f_{min} := f_{min} + f(X)$
  fin pour
  soit *L_{nt} la liste des symboles non terminaux de L*
  soit *$T :=$* CHOIX*(L_{nt})*
  pour *chaque disjonction D_i de la partie droite de la règle T* ***faire***
    soit *$f_{add} := f(D_i)$*
    soit *$f_{sub} := f(T)$*
    $D_{i,suppl} := f_{min} + f_{add} - f_{sub} - f_{ask}$
    $D_{i,proba} := \psi(D_{i,suppl})$
  fin pour
  Choisir une disjonction $D \in D_1, \ldots, D_n$ de la règle T en utilisant les probabilités de sélection D_{proba}
  soit *R_{ins} un arbre de sommet T et dont chaque branche est l'une des conjonctions de J*
  Remplacer dans l'arbre R le symbole T par le sous-arbre R_{ins}
fin tant que
Renvoyer R, un arbre de sommet A dont les feuilles sont des symboles terminaux

Algorithme 5: *Fonction* CRÉATIONARBRE

Nous avons sélectionné plusieurs fonctions probabilistes pour $\psi(n)$ (algorithme A5) qui présentaient des propriétés intéressantes au niveau de leurs tableaux de variations. Par exemple, nous avons testé $\psi(n) = \frac{1}{a+|n|}$ et $\psi(n) = e^{-\frac{(a*n^2)}{b}}$, avec a, b constantes. Celle retenue pour ses bonnes qualités de sélection fut $\psi(n) = e^{-\frac{(2*n^2)}{5.2}}$.

5.3.2.6 Opérateur de croisement

L'opérateur proposé pour croiser deux arbres génétiques est basé sur les deux faits suivants :
- les individus doivent rester cohérents à la sortie de l'opérateur. Notamment les règles de grammaire seront toujours respectées ainsi que les arités des nœuds (un opérateur terminal ne doit jamais changer d'arité),
- la probabilité de sélection de chacun des nœuds doit être identique. Même si dans la réalité, les

recombinaisons génétiques mettent en œuvre un système chromosomique très complexe, il est bon de garder la même équité de brassage comme dans le cas des chaînes binaires. D'autant plus que cela évite de privilégier les nœuds situés au sommet de l'arbre par rapport aux nœuds internes,
- l'algorithme doit apporter une garantie qu'à la fin de l'application de l'opérateur génétique, le ou les enfants résultant(s) devront être différents des parents.

On ne travaille que sur la description génotypique des arbres. À la fin de l'application de l'opérateur génétique, les arbres résultant sont convertis en arbres phénotypiques et insérés dans la nouvelle population. L'utilisation des arbres génotypiques garantit les trois points ci-dessus. Par exemple, l'échange de deux sous-arbres, même dérivant d'un opérateur terminal identique, ne sont pas forcément échangeables s'ils ne sont pas dans le même *contexte* grammatical, c'est-à-dire s'ils dérivent de deux règles de grammaires distinctes.

L'algorithme A6 présente le fonctionnement de l'opérateur de croisement.

ALGORITHME A6
ALGORITHME DU CROISEMENT DANS GRAMGEN

$\rightsquigarrow$ PARAMÈTRES - A_1 et A_2 sont les deux arbres génotypiques à croiser
$\rightsquigarrow$ RÉSULTATS - A'_1 et A'_2 sont les deux arbres génotypiques obtenus
$\rightsquigarrow$ $S(A)$ est une fonction qui renvoie la liste des symboles *left-defined* de la grammaire G définis dans l'arbre A
$\rightsquigarrow$ DERIV(A,X) est une fonction qui renvoie la liste des nœuds de l'arbre A correspondant au symbole X
$\rightsquigarrow$ CHOIX(L) est une fonction qui choisi de manière uniformément aléatoire un élément de l'ensemble L
$\rightsquigarrow$ CHERCHEFILS(A,n) est une fonction qui renvoie le fils numéro n du nœud A
$\rightsquigarrow$ X est un symbole de la grammaire
$\rightsquigarrow$ n_1 et n_2 sont des nœuds d'arbres génotypiques

pour *chaque arbre* $A \in \{A_1, A_2\}$ ***faire***
  tant que NBFILS(A) = *1* ***faire***
    $A :=$ CHERCHEFILS(*A, 1*)
  fin tant que
fin pour
soit $L := S(A_1) \cap S(A_2)$
soit $X :=$ CHOIX(L)
soit $N_1 :=$ DERIV(A_1, X)
soit $N_2 :=$ DERIV(A_2, X)
soit $n_1 :=$ CHOIX(N_1)
soit $n_2 :=$ CHOIX(N_2)
- *Échanger* n_1 *et* n_2 *dans les arbres* A_1 *et* A_2
- *Renvoyer les arbres résultants* A'_1 *et* A'_2

Algorithme 6: *Fonction* CROISEMENT

Le seul cas pour lequel les enfants seraient identiques aux parents est le cas d'un arbre génotypique filiforme. Ceci correspond à une grammaire constituée d'un seul symbole terminal, autrement dit l'arbre phénotypique correspondant est constitué d'un seul symbole. Dans ce cas, le croisement ne peut pas faire mieux que d'échanger ce symbole avec l'un des symboles correspondant de l'autre parent, en garantissant le respect de la grammaire. Après l'opération de croisement, il est possible que le critère de taille moyenne des arbres ne soit plus respecté. Ce critère intervient alors *a posteriori* dans la fonction d'évaluation.

Les paramètres définis par l'utilisateur (outre la grammaire) sont peu nombreux :

- la quantité Q d'individus à croiser. Beaucoup [Goldberg, 1989; Schoenauer et Michalewicz, 1997] considèrent que 80% est une valeur acceptable, mais nous l'avons fait varier entre 70% et 95%,
- et le type de sélection des individus à croiser.

5.3.2.7 Opérateur de mutation

Le principe incrémental, adaptatif et la plus grande latitude permise par cet opérateur nous a conduits à définir trois sous-opérateurs différents et complémentaires, qui sont sélectionnés chacun de manière uniformément aléatoire. Chacun de ces opérateurs prend en entrée un arbre génotypique et renvoie en sortie l'arbre modifié. La conversion en arbre phénotypique est nécessaire avant l'insertion dans la nouvelle population pour le calcul de la fonction d'évaluation. Dans tous les cas, les contraintes déjà exposées pour l'opérateur de croisement ont été respectées ici aussi (cohérence, équité de sélection, production d'enfants nouveaux).

Les trois opérateurs sont les suivants :

- Mutation d'un nœud : consiste à supprimer l'un des nœuds de l'arbre et à le remplacer par un équivalent respectant la grammaire.
- Mutation d'un terminal : consiste à modifier l'une des valeurs d'un terminal numérique (constante ou variable exogène).
- Mutation par auto-croisement : consiste à croiser l'arbre avec lui-même. Cette mutation est complémentaire par rapport aux autres car elle permet par exemple d'inverser numérateur et dénominateur, ce qui n'est pas possible dans les autres cas.

À cause du fonctionnement interne de cet opérateur, sa présentation est plus efficace sous la forme d'un diagramme que d'un pseudo-algorithme. Nous avons exceptionnellement représenté le principe de l'opérateur de mutation sur la figure 5.14, malgré son manque de clarté.

Globalement, les différents opérateurs garantissent que les contraintes numériques de l'utilisateur sont respectées au mieux (par exemple, le fait que la taille de l'arbre soit toujours dans un intervalle de tailles pré-définies) de même que la grammaire. Certaines procédures ont été mises en place pour gérer le cas des grammaires mal conçues : dans le cas du dernier opérateur (mutation par auto-croisement), il est impossible d'échanger un nœud avec l'un de la descendance de ce nœud. Cependant, cela peut arriver avec une grammaire autorisant les arbres filiformes. Dans ce cas, le problème est détecté et l'arbre non muté est renvoyé.

Les paramètres définis par l'utilisateur sont les suivants :

- la quantité Q d'individus à muter. Nous avons considéré des pourcentages compris entre 5% et 45%,
- le type de sélection des individus à muter,
- et dans le cadre de l'opérateur de mutation d'un terminal, la nouvelle valeur est sélectionnée dans un intervalle $[x-v; x+v]$ où x est la valeur précédente et v est un paramètre de variation dépendant de la taille de la plage autorisée pour cette valeur (rendant ainsi l'opérateur indépendant de l'échelle des données).

Malgré la complexité de l'opérateur de mutation, la plupart des opérations sont conduites de manière automatique par la grammaire, ce qui décharge l'utilisateur de tout paramétrage important. Néanmoins, cet opérateur peut être amélioré. Par exemple, le principe des *opérateurs à taux progressif* retenu pour **ICUX** n'a pas encore été testé dans le cadre de **ProgGen** ou de **GramGen**. D'autre part, la présence de constantes entraîne une augmentation considérable de l'espace de recherche. Il existe des techniques visant à geler certains individus et à optimiser localement les valeurs de leurs constantes, mais nous ne les avons pas encore implémentées. Lors de nos expérimentations, nous nous sommes contentés d'utiliser l'opérateur opCST que très parcimonieusement, ce qui permet toutefois d'atténuer l'effet du sur-apprentissage et de rendre les formules trouvées légèrement plus compatibles avec de nouvelles données. Ces perfectionnements seront sans doute apportés aux algorithmes par la suite.

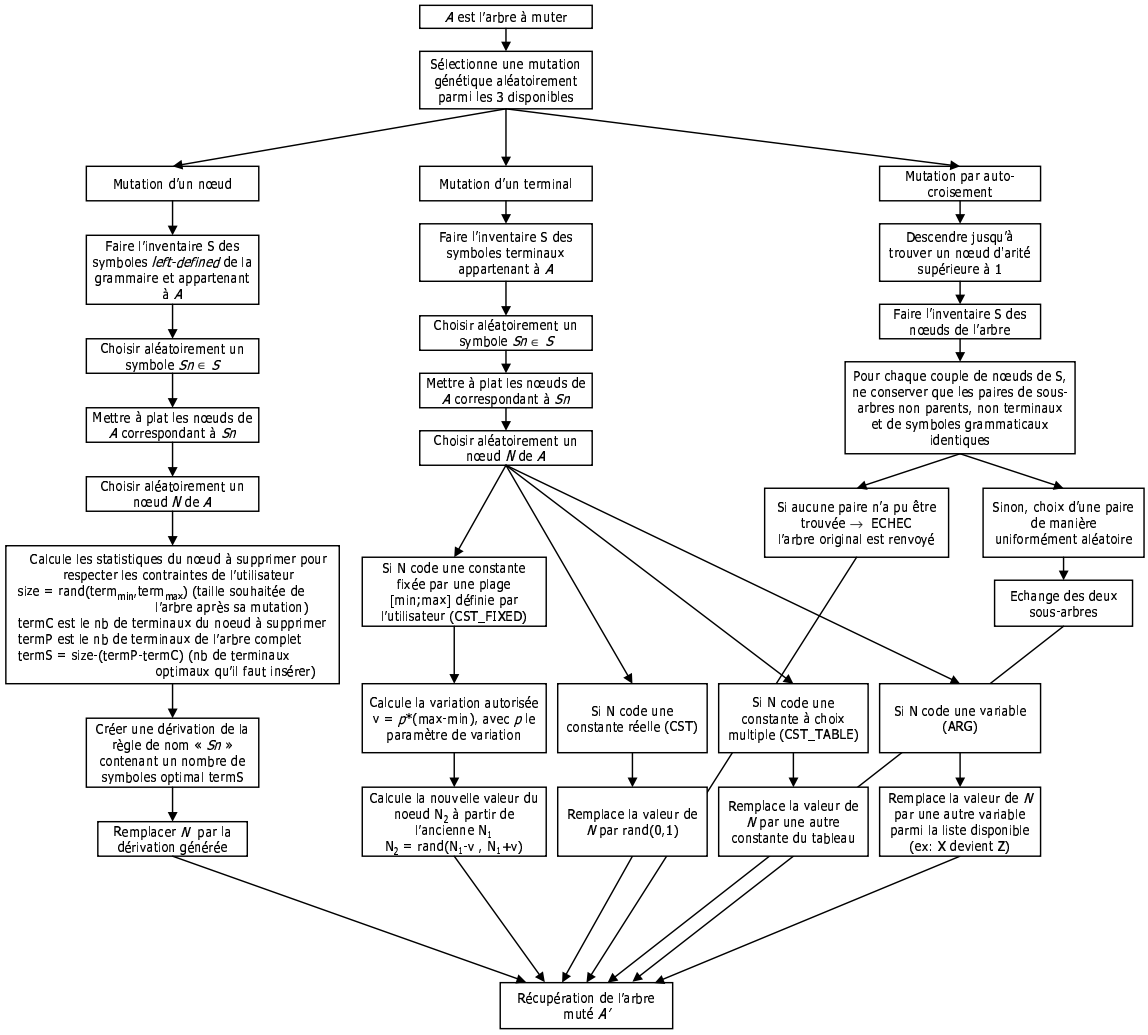


FIG. 5.14 – Opérateurs de mutation de **GramGen**.

5.3.3 Synthèse

Nous avons développé dans cette section deux nouvelles représentations, manipulées par les algorithmes **ICUX** et **GramGen**, découvrant des règles capables de traiter le problème de l'*unmixing* sans faire appel à un modèle linéaire (ang., *Linear Mixture Model* ou LMM), généralement utilisé pour cette approche [Kriebel et Koepke, 1987; Meerkoetter, 1990; Qin et al., 1996]. Cette représentation a nécessité la redéfinition de certains opérateurs génétiques en conséquence afin que les individus engendrés soient cohérents. Ces modèles étendent les représentations plus simples abordées par **ICU** et **XCS-R**. Même si ces deux modèles permettent d'obtenir des résultats très satisfaisants, surtout après post-traitements de la base de règles par algorithme génétique ou par extraction d'un arbre de décision, il nous a semblé nécessaire de pouvoir procurer à l'utilisateur une représentation qu'il peut directement manipuler, ce qui ne peut être apporté que par la programmation génétique.

L'intégration d'une expertise basée sur des intervalles flous dans **ICUX** permet à la fois d'utiliser de l'information experte mixte, mais aussi de l'information partielle (attributs non renseignés ou absents). À titre d'illustration, nous pouvons évoquer le cas des régions pour lesquelles l'expert est certain qu'une

classe X donnée ne s'y trouve pas. Pour inclure ce type d'expertise en apprentissage, il convient de présenter à l'algorithme des échantillons comprenant une contribution de 0 à 100% pour toutes les classes, sauf pour la classe X. Ce type de représentation permet d'inclure des connaissances expertes de niveau supérieur, mais celles-ci sont malheureusement plutôt rares et font encore l'objet d'un support oral.

5.4 Conclusion

Nous avons consacré deux sections à détailler plusieurs algorithmes évolutifs fonctionnant avec des représentations différentes. La base évolutive de ces algorithmes autorise, sous la condition d'un paramétrage adéquat, leur robustesse face à la complexité des données observées en télédétection. De plus, ils n'imposent aucune régularité sur la fonction étudiée (continuité, dérivabilité, convexité, etc). Au niveau de l'utilisateur, le choix de l'utilisation de l'un ou l'autre de ces algorithmes est relativement aisé. Il dépendra en premier lieu du type de problème à résoudre. En classification *hard* ou *soft*, on aura tendance à utiliser les algorithmes **ICU** ou **XCS-R**. En classification floue ou par intervalles flous, on a le choix entre **ICUX** et **GramGen**. Ensuite, tout dépend de la quantité ou de la structure interne des données elles-mêmes. Nous avons observé qu'en général **ICU** donne de meilleurs résultats que **XCS-R**, il est donc intéressant lorsque l'on recherche des règles simples avec une performance élevée. De plus, la représentation des règles de **ICU** est adaptée à la télédétection. Cependant, **XCS-R** possède une bonne capacité de généralisation, il pourra donc être employé lorsque l'expertise comprend un nombre de validations terrain très faible par rapport à la taille du terrain à étudier, ce qui était souvent le cas pour le projet TIDE. Concernant le choix entre **ICUX** et **GramGen**, il dépend essentiellement du type de données expertes : l'algorithme **ICUX** permet de répondre concrètement au problème de la classification par intervalles flous, actuellement émergent en télédétection et propose une représentation par blocs plus adaptée à des données hétérogènes (comme c'est le cas notamment des classes de végétation étudiées dans le projet TIDE). L'algorithme **GramGen** est, quant à lui, réservé à la découverte de fonctions continues dans les données, par exemple, des indices de végétation. Il autorise aussi l'expert à forger ses propres représentations : celle de **ICU** peut par exemple être reproduite dans **GramGen**. Le choix de l'algorithme peut donc être totalement déterminé en fonction des besoins des experts. Cependant, **XCS-R** a tendance à produire des bases de règles de taille importante. Nous montrons dans le chapitre suivant qu'il est possible de réduire ces bases, sans provoquer une diminution trop importante de leurs performances.

Chapitre 6

Post-traitements d'un ensemble de règles

Le post-traitement des bases de règles est une piste intéressante pour contourner certains défauts liés aux algorithmes d'apprentissage. Dans ce chapitre, nous parlons de post-traitements des bases de classifieurs obtenues avec **XCS-R** pour réduire leur taille et améliorer leur lisibilité. Ces post-traitements ont été développés durant nos travaux et concernent le raffinement des représentations des règles ou la recherche d'un certain nombre de caractéristiques que renferment ces représentations pour dégager des remarques intéressantes concernant la structure interne des données (par exemple, le fait que les règles puissent être rassemblées sous forme de *niches*).

Nous verrons dans les expérimentations présentées dans le chapitre suivant que l'algorithme **XCS-R** permet de produire des bases de règles d'assez bonne qualité. Le principal défaut que l'on pourrait cependant reprocher à ces bases est qu'elles contiennent souvent un nombre de règles trop élevé. Ce chapitre propose de nouveaux post-traitements afin de simplifier les bases de règles de **XCS-R**. Trois approches distinctes sont étudiées : une approche génétique pour combiner les règles de la population entre elles (section 6.1), l'étude d'une fonction d'appartenance pour améliorer la capacité de généralisation des règles (section 6.2) et enfin un algorithme combinant **C4.5** [Quinlan, 1993] avec **XCS-R** permettant de transformer une base de règles (représentation plate) en arbre de décision (représentation arborescente, section 6.3).

6.1 Post-traitements génétiques

Comme nous l'avons vu, **XCS-R** tente de construire une représentation complète du problème comprenant toutes les combinaisons possibles de paires (<*condition*> , <*action*>). Appliquée à de petits problèmes, cette représentation est très efficace car elle permet à l'algorithme de modéliser les conditions d'activation pour les échantillons représentant la non-classe $\overline{C}$ plutôt que directement la classe C, si cela s'avérait plus efficace. Cependant, pour les données multispectrales ou hyperspectrales, nous avons remarqué que cette représentation donnait lieu à une base de règles souvent très grande, par rapport au nombre de classes à apprendre (par exemple, 2300 classifieurs pour 5 classes, sur une image multispectrale de quatre bandes). Nous avons donc cherché à optimiser les classifieurs contenus dans un ensemble de règles *apprises* pour en réduire le nombre. Cette optimisation consiste à modifier le nombre de classifieurs, leur paramètre de *spécificité* et les valeurs des contraintes de la partie <*condition*> des règles, afin d'en étudier l'impact sur leurs performances. Comme l'espace de recherche est très grand, nous avons choisi de développer un algorithme génétique, spécifique à l'optimisation de chaque type d'informations contenues dans la population, ce qui nous a amené à proposer trois génomes différents. Le premier (G_1) concerne la simplification par la suppression directe des classifieurs de la base. Le second (G_2) s'intéresse à l'influence des transformations du type agrandissement, réduction et translation des contraintes des classifieurs. Le

dernier (G_3) permet de découvrir de nouveaux classifieurs en recombinant ceux du jeu de règles, à l'image de l'opérateur de croisement. Nous discutons dans chacun des cas, de la représentation choisie pour le génome et des principaux opérateurs (initialisation, fonction d'évaluation, croisement, mutation).

6.1.1 Réduction de la base de règles (génome G_1)

Soit N le nombre de classes. Le génome indique la liste des classifieurs que l'on choisi de garder parmi ceux de la population apprise. Sa longueur est quelconque. Il est interprété de la façon suivante :

1. Chaque gène est un entier représentant un classifieur du jeu de règles. Il est associé à l'une des classes à traiter de manière alternative : si le génome est plus long que N, le gène n_i correspond à la classe $c = i \mod N$.
2. Pour traiter la classe c, un classifieur est choisi dans le jeu de règles de la façon suivante. La valeur n_i correspond à l'index du classifieur parmi tous ceux qui s'activent pour la classe c. Puisque n_i est initialisé aléatoirement, cette valeur peut correspondre à un nombre plus élevé que le nombre total de classifieurs dans la population qui correspond à cette classe. Dans ce cas on en prend le modulo. Par exemple, si $n_2 = 123$ et qu'il y a 24 classifieurs dans le jeu de règles qui s'activent pour la classe 2, on choisira le classifieur d'index 3 ($123 \mod 24$). Le classifieur choisi sera donc le 3$^{\text{e}}$ classifieur qui code la classe 2.
3. Pour assurer la cohérence des génomes produits, les nouveaux jeux de règles obtenus après leur interprétation doivent être constitués de classifieurs différents. Si lors de l'initialisation, de la mutation ou du croisement génétique, un gène sélectionne le même classifieur qu'un autre gène déjà présent, la valeur de celui-ci est augmentée d'une unité et les étapes 2 et 3 sont exécutées autant de fois que nécessaire.

Les opérateurs génétiques. Le croisement et la mutation sont cohérents puisque l'attribution des classes pour chaque gène se fait toujours sur une classe qui leur est réservée (les gènes situés à la même position dans deux individus parents correspondent toujours à la même classe). Nous utilisons un croisement à un point et la mutation ne modifie aléatoirement qu'un seul gène par appel.

La fonction d'évaluation ϕ_1. La fonction *fitness* définit la performance d'un individu particulier par rapport au reste de la population. Le calcul de la fonction d'évaluation $\phi_1(L)$ pour une liste L de classifieurs extraits d'un jeu de règles appris en fonction de l'interprétation d'un génome G_1 donné est le suivant.

Soit une liste de classifieurs $L = \{C_k : A_i(k) \Rightarrow P_k\}$ où C_k est la partie condition du classifieur k, $A_i(k)$ est la partie action du classifieur k représentant la classe A_i et P_k est la récompense associée au classifieur k. On commence par chercher la classe A d'un exemple S à partir de la liste de classifieurs L, en associant à chacune des classes à apprendre A_i, une *fitness* pondérée $f_1(L_S, S, A_i)$ qui se calcule par :

$$f_1(L_S, S, A_i) = \frac{\sum_{i=0}^{n_r} p_i * F_i}{\sum_{i=0}^{n_r} F_i} \tag{6.1}$$

où $L_S = \text{Match}(L, S, A_i)$ représente les classifieurs de L qui sont activés par l'exemple S et qui codent pour la classe A_i, S est l'exemple dont il faut calculer la classe, n_r est le nombre de classifieurs dans L_S, et p_i et F_i représentent respectivement les valeurs de récompense et de *fitness* associées au classifieur i.

La classe A de l'exemple S est alors simplement la classe A_i qui maximise $f_1(L_S, S, A_i)$ pour toutes les classes à apprendre. On cherche alors les classes de tous les exemples de la base d'apprentissage. Pour obtenir la valeur de $\phi_1(L)$, on calcule le taux d'exemples bien classés par rapport au nombre total d'exemples.

6.1.2 Transformation de classifieurs (génome G_2)

Cette étude vise à connaître l'influence d'une modification directe des intervalles des classifieurs. Les deux types de modification étudiées sont l'élargissement (ou la réduction) de la taille des intervalles et la

translation (à gauche ou à droite) de ces intervalles dans l'espace de recherche. Ceci permet de généraliser ou de spécialiser les classifieurs en s'efforçant d'optimiser la performance de la base. Chaque gène du chromosome est un triplet codant chacun respectivement le numéro du classifieur qui va être modifié, le paramètre d'élargissement et le paramètre de déplacement à droite (voir la figure 6.1). Les valeurs des gènes appartenant à $\mathbb{R}$, nous avons adapté les opérateurs génétiques en conséquence.

n_1	λ_1	Δ_1	$\cdots$	n_i	λ_i	Δ_i	$\cdots$	n_k	λ_k	Δ_k

FIG. 6.1 – Représentation du génome G_2.

Les valeurs des gènes n_i sont interprétés comme précédemment. Les coefficients λ_i représentent l'élargissement par rapport à la taille originale de l'intervalle. Pour un intervalle $[a; b]$, la nouvelle taille sera de $\lambda_i(b-a)$. Une valeur négative indique une réduction de l'intervalle et 1 représente la valeur neutre. Les coefficients Δ_i représentent le déplacement à droite de l'intervalle. Pour un intervalle $[a_{\lambda_i}; b_{\lambda_i}]$ modifié selon λ_i, la nouvelle borne gauche vaudra $a'_{\lambda_i} = a_{\lambda_i} + \Delta_i(b_{\lambda_i} - a_{\lambda_i})$. Une valeur négative indique un déplacement à gauche et 0 représente la valeur neutre. L'utilisation de la taille et de la position originales des intervalles permet d'obtenir des opérateurs de mutation indépendants de l'échelle des valeurs des données.

Les opérateurs génétiques. D'après une expérimentation non reportée ici, les plages intéressantes[1] pour les différentes transformations sont les suivantes : $[0.5; 1.5]$ pour λ_i et $[-0.5; 0.5]$ pour Δ_i. Ces plages de valeurs sont utilisées par l'opérateur de mutation et d'initialisation. Le croisement et la mutation sont cohérents puisque l'application des gènes se fait toujours sur une classe qui leur a été réservée (le paramètre λ_i est toujours échangé avec un autre paramètre λ_i correspondant à la même classe). La fonction d'évaluation est du même type que celle présentée auparavant (ϕ_1). Les classifieurs sont extraits de la population initiale, modifiés et la performance globale du nouveau jeu de règles représente l'évaluation du génome.

6.1.3 Recombinaison de classifieurs (génome G_3)

Bien que les deux génomes précédents déterminent de nouvelles populations d'individus, aucune création de classifieurs n'a vraiment eu lieu, comme si l'on avait considéré que les populations apprises contiennent déjà la solution au problème ou tout du moins une solution proche. L'opérateur de croisement n'a, par exemple, pas été approfondi. Des classifieurs totalement différents peuvent pourtant être produits par recombinaison de deux classifieurs existants.

$type_1$	α_1	β_1	$\cdots$	$type_i$	α_i	β_i	$\cdots$	$type_k$	α_k	β_k

FIG. 6.2 – Représentation du génome G_3.

Ici, chaque gène d'un individu représente une combinaison obtenue en en utilisant deux autres (voir la figure 6.2). Les gènes α_i et β_i représentent respectivement les index des deux classifieurs à recombiner, en utilisant le même principe que celui décrit pour le génome G_1. Les gènes $type_i$ sont des entiers et servent à encoder le type de recombinaison. La recombinaison de deux classifieurs complets est obtenue en recombinant deux à deux chacun des intervalles composant ces classifieurs. La liste des recombinaisons est présentée dans le tableau 6.1. La taille du génome peut être quelconque.

Les opérateurs génétiques. L'opérateur de mutation est défini de la même façon qu'auparavant : différents types de recombinaison et différents index pour les classifieurs sont essayés. Une mutation spéciale permet d'inverser α_i et β_i dans le génome. Après chaque croisement ou chaque mutation, les bornes d'un intervalle $[a; b]$ sont permutées si $b < a$. La fonction d'évaluation n'a pas été modifiée.

[1] Celles qui ont conduit à la meilleure généralisation sur l'ensemble de test.

Type	Nom	Commentaire
T_1	Copie de I_1	Aucune modification
T_2	Copie de I_2	Aucune modification
T_3	Élongation à droite	L'intervalle devient $[a_1; a_2]$
T_4	Recouvrement	L'intervalle devient $[a_1; b_2]$
T_5	Remplissage	L'intervalle devient $[b_1; a_2]$
T_6	Élongation à gauche	L'intervalle devient $[b_1; b_2]$
T_7	*OR*	$[\min(a_1, a_2); \max(b_1, b_2)]$ (dilatation de I_1 par I_2, c'est-à-dire que I_1 s'allonge jusqu'à couvrir I_2)
T_8	*AND*	$[\min(b_1, b_2); \max(a_1, a_2)]$ (on ne conserve que la partie du chevauchement)
T_9	Par contrainte	I_1 s'allonge à gauche ou à droite jusqu'à la limite de I_2, mais s'évide si I_2 chevauche déjà I_1. Post-condition : I_1 et I_2 sont contigu. I_1 est l'intervalle résultant
T_{10}	Gauche	Intervalle dont le centre est le plus à gauche
T_{11}	Droite	Intervalle dont le centre est le plus à droite
T_{12}	*MIN*	Intervalle le plus fin
T_{13}	*MAX*	Intervalle le plus large
T_{14}	Fusion	Intervalle dont le centre est celui de I_1 et le rayon celui de I_2

TAB. 6.1 – Types d'intervalles qu'il est possible d'obtenir en recombinant deux intervalles $I_1 = [a_1; b_1]$ et $I_2 = [a_2; b_2]$.

6.1.4 Synthèse

Nous avons vu trois génomes, G_1, G_2 et G_3, qui permettent d'optimiser la base de règles de **XCS-R** à l'aide d'un algorithme génétique classique. Ces trois génomes correspondent chacun à des modifications spécifiques de la base et peuvent être appliqués séquentiellement : la base est d'abord réduite (G_1), les classifieurs sont ensuite transformés (G_2) puis recombinés (G_3). D'après les expérimentations que nous présenterons dans le chapitre suivant, les génomes les plus intéressants concernant l'amélioration de la performance des règles sont les génomes G_1 et G_2.

6.2 Étude d'une fonction d'appartenance

Il est généralement reconnu qu'un être humain est plus à l'aise en présence de concepts simples que complexes. La notion de complexité dans un système de classifieurs peut être rattachée, toutes choses égales, au nombre de règles ou à la taille des règles (au sens du nombre de conditions) contenues dans la population. En effet, une bonne interprétation d'une population ne peut pas être exprimée correctement si cette population est trop large. Wilson a montré dans [Wilson, 2000b] que pour le problème du *Wisconsin Breast Cancer Database* (UCI 1998), près de 4500 classifieurs ont été nécessaires pour obtenir une performance satisfaisante au bout de 2.10^6 itérations d'exploration. Dans [Studley et Bull, 2005], 3200 classifieurs sont nécessaires pour atteindre une prédiction de récompense optimale dans le cas d'un problème de type labyrinthe. Ce problème est même relevé par Wilson lui-même dans [Wilson, 2000c], p. 14 : "*Only a fraction of [the conditions of the classifiers] is directly interpretable. It would be desirable to find algorithms for extracting all the classifiers´ implications as rules of thumb and in other representations.*"

Ce type de remarque nous a conduits à explorer une nouvelle forme de post-traitement, basée à la fois sur la réduction *a posteriori* de la base de classifieurs, mais aussi sur la transformation de cette base dans un autre formalisme, plus compréhensible. Parmi tous les formalismes disponibles, l'organisation des données sous la forme d'arbres nous paraît la plus pertinente. Cette section décrit une méthode de réduction d'une base de classifieurs apprise par la découverte de *niches* puis l'extraction de relations entre ces niches et leur représentation sous forme d'arbres. Le concept de *niches*, tel qu'il est décrit dans le rapport [Butz et al., 2004], fait appel à la notion de *modèles* représentatifs d'un ensemble de classifieurs codant pour la même action. Par exemple, les classifieurs 011 : 01 et 001 : 01 peuvent être représentés

par le modèle $0*1:01$.

6.2.1 Le concept de niche et leur découverte

Dans **XCS-R**, une niche est définie à partir de la notion de schéma connue pour les AG depuis les travaux publiés dans [Holland, 1975]. On entend par le terme *niche*, l'ensemble des classifieurs appartenant à un *modèle* donné (par exemple, $*0**01:01$) comprenant les conditions et l'action du classifieur. Un classifieur appartient à une niche s'il est au moins aussi spécifique que le modèle, les bits instanciés et l'action devant être identiques. Par exemple, les classifieurs $\#0\#\#01:01$ et $\#01001:01$ appartienne à la niche de modèle $*0**01:01$. On appelle *instance* l'un des classifieurs qui fait partie d'une *niche*.

L'étude des niches (et notamment de leur taille par rapport à un modèle) nécessite de faire l'inventaire de la totalité des niches pour un problème donné. Il a été montré dans [Butz et al., 2004] qu'il est plus facile de déterminer les niches à partir de la spécification complète d'un problème, qu'*a posteriori*, c'est-à-dire uniquement à partir d'une population finale, ce que nous souhaitons faire ici puisqu'une telle spécification n'est pas réalisable en télédétection.

Dans le cadre du problème de la télédétection, nous devons commencer par redéfinir le concept de niche dans le cadre de données réelles, puisqu'elles ne peuvent plus être représentées par des chaînes binaires.

6.2.2 Définition d'une niche dans XCS-R

Dans l'algorithme **XCS-R** traditionnel, l'ensemble des niches permet de définir la solution la plus générale à un problème donné. Toute l'information doit pouvoir être modélisée par les niches (par exemple, pour le problème du 11-multiplexeur décrit dans [Butz et al., 2004], la solution est représentable par exactement 32 niches) et aucune information supplémentaire ne doit être stockée (pas de bits instanciés inutilement). Cependant, il peut y avoir des recouvrements si la représentation de la solution l'implique. Par exemple, dans le problème car_k,[2] les modèles $000**:0$ et $00*0*:0$ se recouvrent.

Formellement, une niche est construite sur le modèle suivant :

$$M(1)\ M(2)\ \ldots\ M(n) : action \tag{6.2}$$

où chaque condition $M(i)$ est un caractère de l'alphabet $\{0, 1, \#\}$ dans le cas d'une niche de séquences binaires et un intervalle $[M(i)_{min}; M(i)_{max}]$ de valeurs réelles dans notre cas.

Si la valeur de $M(i)_{max} - M(i)_{min}$ est faible, on est en présence d'une condition à *spécificité* élevée. Dans le cas contraire, on est en présence d'une condition générique (correspondant au caractère *joker* dans le formalisme original).

6.2.3 Définition de l'appartenance d'un classifieur à une niche

La fonction d'appartenance est définie par $C \times M \rightarrow \{true, false\}$, où C est l'espace des classifieurs et M l'espace des niches. Pour qu'un classifieur appartienne à une niche, il faut qu'il soit aussi spécifique que celle-ci. Si $C(i)$ est la condition i du classifieur C et $M(i)$ la condition i du modèle M, la fonction d'appartenance de C à M est vraie lorsque les actions spécifiées sont identiques et que C_i est inclus (non strictement) dans $M(i)$ pour tout i.

Par exemple, si nous définissons un modèle M par $[2;4][7;9][1;9]:10$, le classifieur C_1 « $[2;4][7;7][1;9]:10$ » appartient à M car il est plus spécifique sur la condition $C_1(2)$ et le classifieur C_2 « $[3;3][8;8][5;5]:10$ » appartient lui aussi à M car il est plus spécifique sur toutes les conditions. Par contre, le classifieur C_3 « $[2;4][5;9][1;9]:10$ » n'appartient pas à M.

[2] Boolean Carry of size k. Ce problème consiste à découvrir la retenue de la somme de deux séquences de k bits. Par exemple, pour $k=3$, $x=100$ et $y=101$, $\text{car}_k(100101)$ renvoie 1 (retenue sur 3 bits de 1001).

6.2.4 Hypothèses de travail

Les niches et leurs relations (modèles de niches) ne peuvent pas être déterminées lorsque l'on ne fait aucune supposition *a priori* sur la structure interne des solutions du problème posé, comme c'est le cas en général dans le cadre de la classification hyperspectrale. Pour extraire les modèles de niches d'une population de classifieurs apprise, nous avons émis les deux hypothèses suivantes :

Discrimination des classifieurs [*H1*]. Certains classifieurs dans la population P peuvent être pris pour modèle de niches et d'autres classifieurs de la même population appartiennent à ces niches. Nous avons donc deux sous-populations, P_{model} et $P_{content}$, formant une partition de P.

Critère de confiance [*H2*]. Il y a assez de classifieurs spécifiques dans la population et qui appartiennent à ces niches pour mettre en valeur ces modèles. Si la *fitness* des niches est élevée, celles-ci sont proches des modèles de niches optimaux.

Dans ce cas, la découverte des niches peut se faire en formant des couples de classifieurs (C, M) dans lesquels M représenterait le modèle et C le classifieur qui appartiendrait au modèle. Il ne reste alors plus qu'à comparer les classifieurs deux à deux pour découvrir les niches (hypothèse H1). La confiance des modèles ainsi obtenus est déduite du nombre de classifieurs appartenant à chaque niche. Une bonne confiance dans le modèle final est obtenue lorsque l'hypothèse H2 est vérifiée.

Examinons la viabilité de ces hypothèses. H1 se vérifie par la nature même de **XCS-R** : le but de l'algorithme est de découvrir une population comportant des niches optimales et dont le peuplement par niche (nombre de représentants par niche) est maximal. Ainsi, la partition $P_{content}$ est constituée de classifieurs spécifiques créés par le *Covering Operator*. H2 est peut être un peu forte. La proportion de classifieurs spécifiques dans **XCS-R** est relativement faible puisqu'ils sont éliminés par l'AG [Wilson, 1998]. Un certain nombre de classifieurs spécifiques peuvent cependant être créés par le *Covering Operator* ou l'AG en ajustant certains paramètres (par exemple, θ_{GA}, *doGeneralizationMutation*, *dontCareProb* et θ_{Sub}, des paramètres classiques de **XCS** [Butz et Wilson, 2002]).

6.2.5 Découverte de niches *a posteriori*

Pour tester l'appartenance d'un classifieur à une niche représentée par un autre classifieur, nous avons utilisé l'algorithme A7. La fonction BelongsTo prend deux classifieurs C et M en paramètre et renvoie *vrai* si C est inclus dans M.

En ordonnant deux classifieurs, cette fonction définit sur une population complète une relation d'ordre partiel. Cette relation permet de retraduire graphiquement la population en un graphe acyclique (orienté, sans circuit) G_a, dans lequel chaque sommet représente un classifieur. Les arcs de G_a peuvent être vus comme définissant une relation entre un classifieur et sa niche. La niche la plus générale est donc celle qui se trouve ordonnée avant toutes les autres, dans un sous-graphe connexe donné. Nous avons donc développé une représentation graphique pour visualiser ces graphes. Un exemple d'un tel graphe est représenté sur la figure 6.3. Chaque nœud fait référence à un classifieur de la population originale. Les relations d'appartenance y sont représentées uniquement pour les classifieurs correspondant à la classe *eau* : le graphe complet comprend neuf graphes connexes.

6.2.6 Résultats obtenus

Nous pouvons constater que le graphe résultant est complexe à interpréter directement, mais déjà plus simple qu'un jeu de plusieurs milliers de classifieurs. Un autre test qui a été effectué consistait à connaître la fréquence des niches et la répartition des couples (*classifieurs*, *modèles*) au sein de la population. Ces couples, calculés par la fonction BelongsTo, peuvent être présentés sous la forme d'une matrice d'adjacence binaire (voir la figure 6.4). Pour plus de clarté, un extrait est présenté sur la figure 6.5. Chaque ligne et chaque colonne représente un classifieur. La population complète (comprenant ici 2300 classifieurs) peut donc être représentée sur la même figure. Sur cette figure, un pixel situé sur la colonne i et la ligne j représente le fait que le classifieur C_i appartienne au modèle M_j. Le pixel est représenté en blanc lorsque la relation est fausse et dans une couleur correspondant à la classe de l'exemple lorsque la relation est vérifiée.

ALGORITHME A7
RENVOIE L'APPARTENANCE D'UN CLASSIFIEUR À UN MODÈLE

⇝ PARAMÈTRES - C est le classifieur à tester et M le modèle correspondant
⇝ RÉSULTAT - *vrai* si l'appartenance est vérifiée, *faux* sinon

si $C.action \neq M.action$ ***alors***
  Renvoyer **faux**
fin si
pour *chaque condition* $C(i)$ *de* C *et* $M(i)$ *de* M ***faire***
  si $M(i)$ *est une condition* joker ***ou*** $(M(i)_{max} - M(i)_{min}) > seuil$ ***alors***
    Passer à la condition suivante
  sinon
    si $C(i)_{min} < M(i)_{min}$ ***alors***
      Renvoyer **faux**
    sinon
      si $M(i)_{max} < C(i)_{max}$ ***alors***
        Renvoyer **faux**
      sinon
        Passer à la condition suivante
      fin si
    fin si
  fin si
fin pour
S'il n'y a plus de conditions à tester, renvoyer **vrai**

Algorithme 7: *Fonction* BELONGSTO*(C, M)*

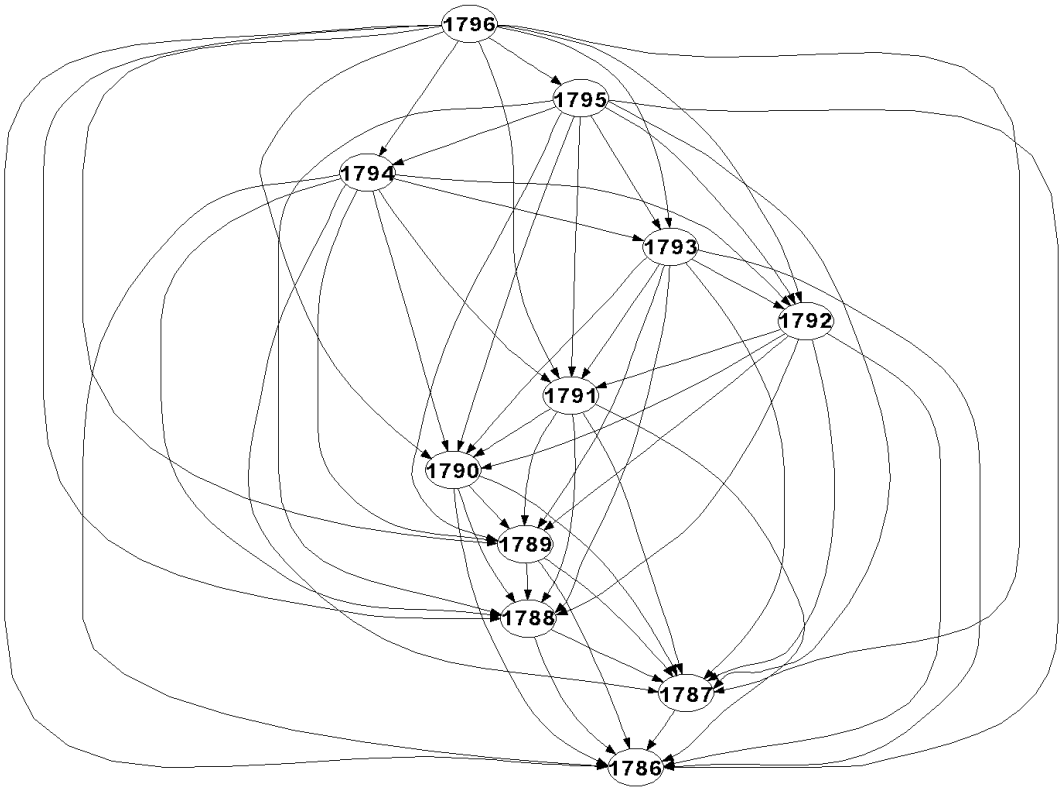


FIG. 6.3 – Représentation des relations d'appartenance sous la forme de graphe.

Ce test nous a conduits à considérer les relations d'adjacences obtenues comme étant très complexes à interpréter humainement. De nombreux classifieurs se recouvrent partiellement et pour relier deux classifieurs du graphe, il existe plusieurs *chemins* possibles. Un algorithme a alors été conçu pour détecter et éliminer les arcs supplémentaires du graphe G_a, c'est-à-dire les arcs inclus dans des chemins non directs. L'algorithme A8 prend en paramètre deux nœuds A et B et renvoie *vrai* si les nœuds ne sont reliés que par un seul chemin. L'algorithme A9 a alors été défini pour simplifier le graphe.

La figure 6.6 montre deux extraits simplifiés de la matrice d'adjacence présentée sur la figure 6.4. Dans la figure 6.6, les sommets[3] des arbres représentent les modèles de niches pour les classifieurs symbolisés par les nœuds fils. La topologie de ces arbres est très variable d'une classe à l'autre, depuis des structures simples à deux nœuds jusqu'à des structures d'une centaine de nœuds. Les structures sont tout à fait liées à la complexité réelle des données, ici des classes de végétation : les végétations qui sont situées à la fois en bordure d'eau ou à plus haute altitude comme le *juncus maritimus* présentent les structures les plus riches. Nous sommes allés jusqu'à l'étude détaillée de certains classifieurs pour mieux comprendre l'apport des graphes produits. Étudions, par exemple, ceux d'identifiants 117, 438, 566 et 828. Leurs performances et leurs relations au sein de la niche sont représentées sur le graphe central de la figure 6.6 et leurs parties <*condition*> sont représentées de manière graphique sur la figure 6.7. Les représentations montrent clairement que le classifieur 117 généralise les trois autres : ce classifieur est au sommet de sa niche (figure 6.6) et deux de ses parties <*condition*> sont plus grandes et recouvrent les conditions des autres classifieurs (les deux conditions sont représentées chacune sur un graphique de la figure 6.7). Les performances observées sur la figure 6.6 confirment cela : la performance du classifieur 117 est en effet supérieure à celle des autres classifieurs (0.805 contre 0.799). Ces graphes ont plusieurs

[3] Un sommet est un nœud sans arcs incidents intérieurement.

ALGORITHME A8
TESTE L'ABSENCE DE CHEMINS INUTILES DANS LE GRAPHE D'APPARTENANCE

⇝ PARAMÈTRES - G_a est le graphe d'appartenance, A et B sont les deux nœuds à tester
⇝ RÉSULTAT - *vrai* si A et B ne sont reliés par aucun chemin inutile, *faux* sinon
⇝ SUCC(G,X) est une fonction renvoyant le successeur du nœud X dans le graphe G

soit $L = \{\}$
pour *chaque nœud* N *de* G_a ***faire***
  si N = SUCC(G_a,A) *et* $N \neq B$ ***alors***
    $L := L \cup \{N\}$
  fin si
fin pour
tant que $|L| > 0$ ***faire***
  Choisir un nœud M *dans* L
  $L := L \setminus M$
  pour *chaque nœud* N *de* G_a ***faire***
    si N = SUCC(G_a,M) ***alors***
      $L := L \cup \{N\}$
    fin si
    si $N = B$ ***alors***
      Renvoyer **faux**
    fin si
  fin pour
fin tant que
S'il n'y a plus de nœuds à tester, renvoyer **vrai**

Algorithme 8: *Fonction* NEIGHBOR(G_a, A, B)

ALGORITHME A9
SIMPLIFIE UN GRAPHE D'APPARTENANCE

⇝ PARAMÈTRE - G_a est le graphe d'appartenance
⇝ RÉSULTAT - le graphe simplifié

soit $G_b = G_a$
pour *chaque nœud* N_1 *de* G_b ***faire***
  pour *chaque nœud* N_2 *de* G_b *tel que* N_2 = SUCC(G_b,N_1) ***faire***
    si NEIGHBOR(G_b, N_1, N_2) = faux ***alors***
      Supprimer de G_b *l'arc entre* N_1 *et* N_2
    fin si
  fin pour
fin pour
Renvoyer G_b

Algorithme 9: *Fonction* SIMPLIFY(G_a)

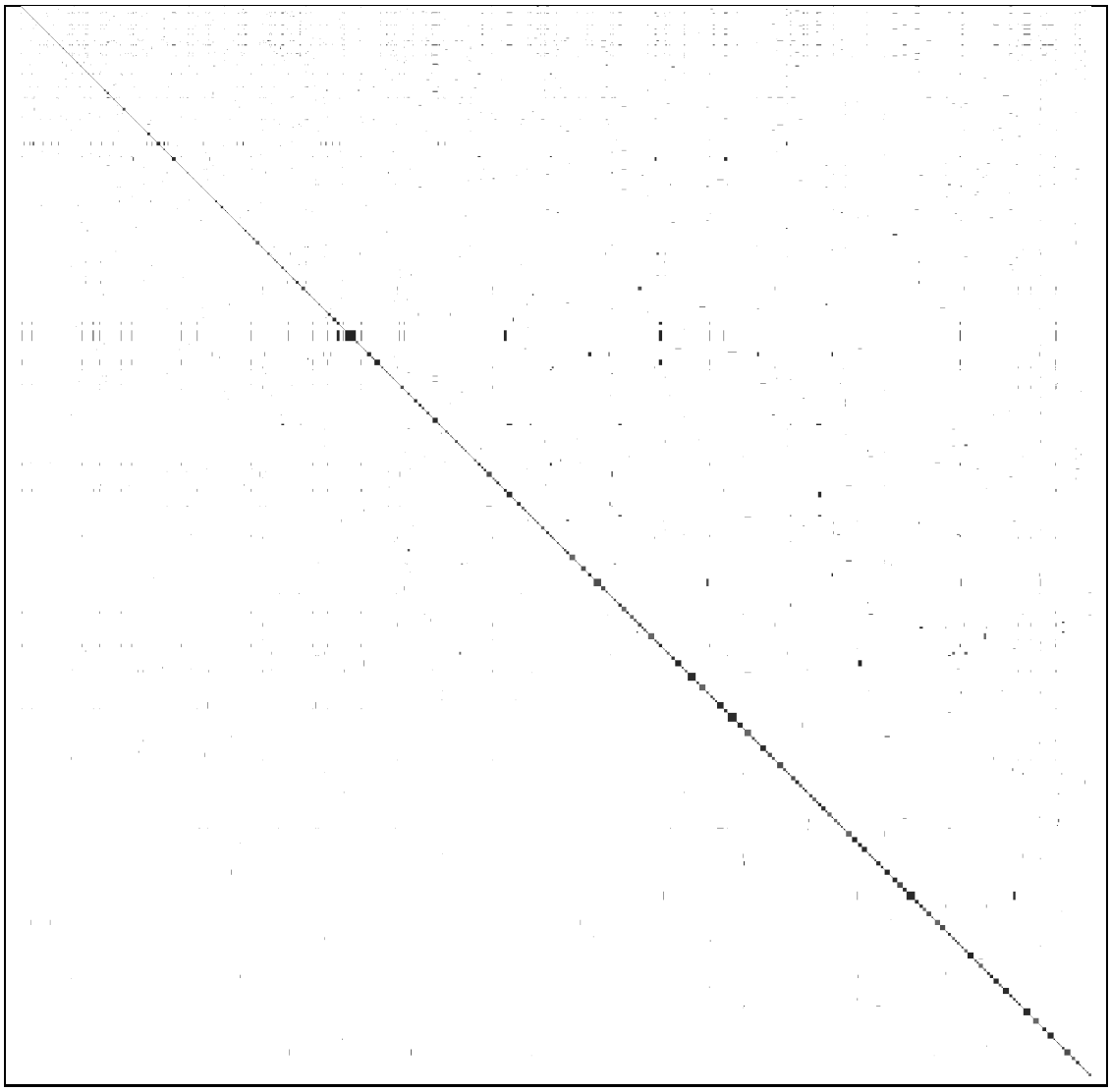


FIG. 6.4 – Représentation des relations d'appartenance sous la forme de matrice.

intérêts : ils autorisent une interprétation plus aisée des relations entre les classifieurs et mettent en valeur des classifieurs qui peuvent être considérés comme des modèles de niches. Il est ensuite possible, lorsque cela s'avère nécessaire, de se rapprocher d'avantage du modèle optimal par optimisation locale des paramètres des meilleurs modèles trouvés, par exemple en utilisant un AG classique (*tuning*). Si la structure des données révèle une relation épistatique entre deux gènes[4], on peut noter que la représentation hiérarchique exposée ici ne la mettra pas en évidence. En effet, pour un classifieur donné A, soit les conditions correspondantes à N gènes épistatiques sont ensembles plus larges ou plus étroites que celles d'un classifieur témoin B et dans ce cas les classifieurs A et B seront situés dans le même graphe d'appartenance. Soit les conditions correspondantes à ces N gènes doivent évoluer indépendamment et dans ce cas, les classifieurs correspondants seront séparés dans deux graphes différents. Il est possible qu'une étude par gène et non plus par classifieur puisse résoudre ce problème.

[4]L'épistasie est observée lorsqu'un gène altère l'effet d'un autre.

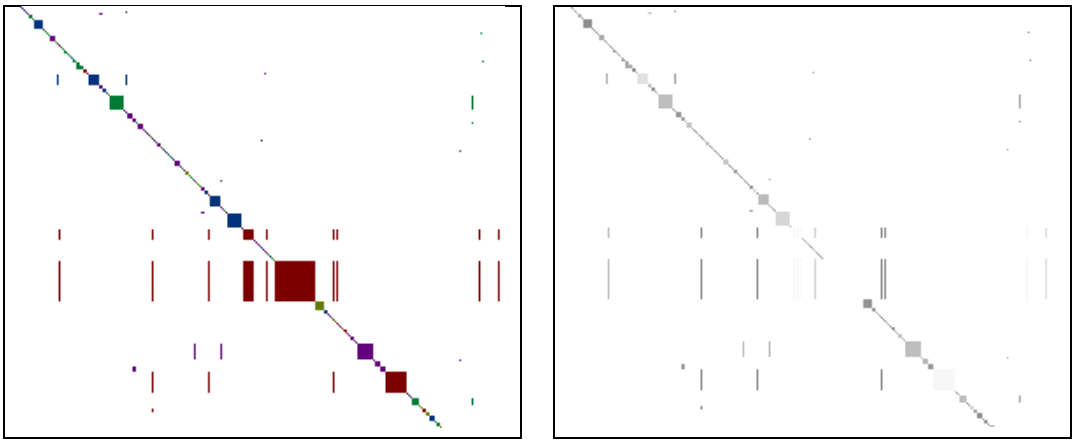


FIG. 6.5 – À gauche : extrait de la matrice d'adjacence. À droite : la saturation (couleur) de chaque point a été fixée en fonction de leur qualité sur un jeu de test. Plus le point est sombre, plus le classifieur correspondant est performant.

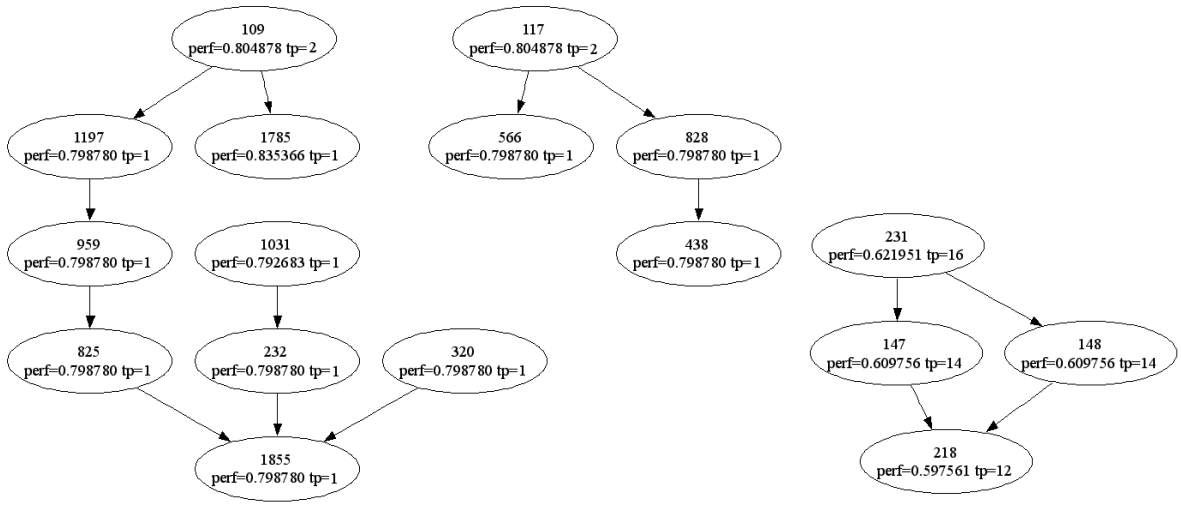


FIG. 6.6 – Simplification du graphe d'appartenance en utilisant l'algorithme A9 Chaque nœud indique l'identifiant du classifieur, la qualité et le nombre de vrais positifs (*TP rate*). La relation indiquée par les flèches est « ce modèle inclut ce classifieur ».

6.2.7 Synthèse

Cette étude présente une technique pour ordonner *a posteriori* les classifieurs d'un jeu de règles appris. Cet ordonnancement a permis de tirer un certain nombre de conclusions sur la performance des modèles par rapport à une population complète et offre la possibilité de visualiser et d'interpréter la hiérarchie des modèles et des classifieurs par rapport à la réalité du terrain.

Cependant, une partie des résultats obtenus n'a pas pu être exploitée directement. Il s'agit des classifieurs *orphelins* ou *bi-connexes*. Les classifieurs *orphelins* sont des nœuds de graphe de degrés intérieur et extérieur nuls. La *bi-connexité* dénote deux classifieurs dont l'un est à la fois *modèle* et *instance* de l'autre : dans la grande majorité des cas, ces classifieurs dégénérés sont deux classifieurs identiques obtenus tardivement par réplication génétique et qui n'ont pas eu le temps d'évoluer ou de deux classifieurs âgés qui ont évolué séparément mais qui ont finalement donné lieu à des génotypes similaires sous la pression génétique.

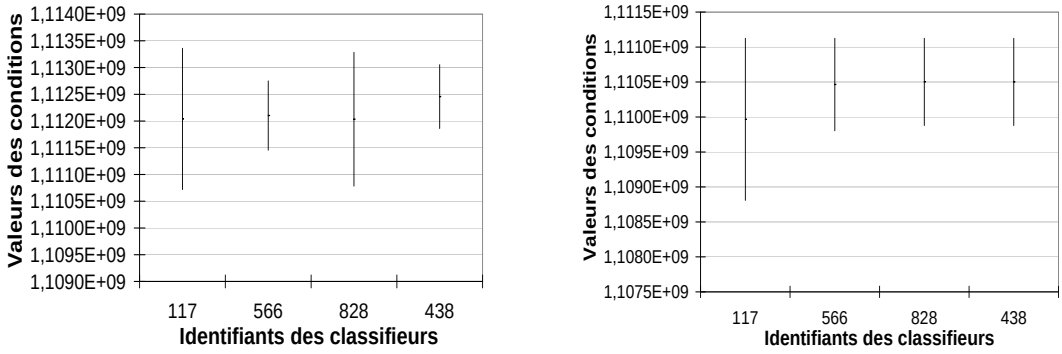


FIG. 6.7 – Étude comparée de parties <*condition*>. Le graphique de gauche (respectivement de droite) montre les premières (respectivement les secondes) conditions de quatre classifieurs différents. Les marques noires représentent les centres des intervalles.

Concernant les niches, malheureusement, peu de suppositions peuvent être faites à leur égard. Si elles présentent une performance élevée sur une base de test (en utilisant, par exemple, les mesures de qualité TPR[5] ou TNR[6]), elles ont de fortes chances d'être des modèles. Cependant, aucune information dans la population ne permet réellement de déduire le fait qu'elles le sont vraiment ou non. On ne peut déduire leur capacité de généralisation du nombre d'exemples validés par ces classifieurs puisque l'on ne possède aucune information sur le degré de représentativité de ces exemples par rapport au problème posé. Par exemple, si un classifieur *orphelin* active de nombreux échantillons, cela ne suffira pas à en faire un classifieur générique : les propriétés dérivées à partir de l'étude d'exemples qui sont activés par un classifieur dont la capacité de généralisation n'est pas connue ne peuvent pas servir à argumenter sur cette capacité de généralisation elle-même.

6.3 Extraction d'un arbre de décision

L'étude précédente a démontré la pertinence d'une optimisation intervenant en aval d'un apprentissage de **XCS-R**. Les principaux avantages en sont le gain de temps par rapport à une optimisation génétique des classifieurs, une simplification de la base des classifieurs en terme de taille de population et de représentation, mais aussi le fait que l'adaptation de ce type de post-traitements à d'autres systèmes de classifieurs s'effectue sans difficultés. Cependant, une critique forte que l'on peut adresser à cette étude concerne la hiérarchisation du jeu de règles sous la forme de relations modèle-instance : en définitive, pour chaque arbre connexe, seul le modèle inscrit au sommet est conservé et se substitue à l'ensemble des classifieurs du reste de l'arbre. La taille de la population est certes réduite mais on perd beaucoup d'informations apportées par les classifieurs. Notamment leur position dans la hiérarchie ainsi que la répartition des instances dans l'espace de définition du modèle : en effet, on pourrait tirer parti de la structure du *sous-modèle* qu'ils engendrent. De plus, les classifieurs *orphelins* ne sont pas exploités correctement puisque, malgré le fait qu'ils sont relativement nombreux (environ 90% des classifieurs selon nos expérimentations), on ne peut que les supprimer ou les inclure à l'identique (sans rectification) dans les graphes.

Nous avons donc tenu à proposer une autre orientation dans laquelle la totalité des classifieurs pouvait être *a priori* utilisés comme nœud d'un arbre (ce type de représentation étant considéré au fil de nos travaux comme tout à fait convenable et conviviale) et dont la profondeur au sein de cette hiérarchie dépendait directement de la performance de ces classifieurs sur une base d'exemples. Ce type de représentation fait appel à la notion de gain de performance qu'apporterait la classification d'un certain

[5]True Positive Rate.
[6]True Negative Rate.

nombre d'exemples par un classifieur donné plutôt qu'un autre. Pour simplifier la base de classifieurs, nous souhaitions aussi détecter et éliminer les classifieurs qui ne sont pas discriminants. Ces principes se retrouvent justement dans certains algorithmes inductifs, qui fonctionnent à partir d'une mesure de gain et qui intègrent un processus d'élagage. Une hybridation entre le système de classifieurs **XCS-R** et l'algorithme **C4.5** [Quinlan, 1993] a alors été proposée dans cette optique. L'algorithme conçu, nommé **XCS5**, fonctionne de la même façon que **C4.5**, sauf que les tests étiquetant chacun des nœuds de l'arbre ne sont plus déterminés au cours de l'apprentissage mais sont représentés par des classifieurs piochés au sein d'une population apprise. Le formalisme des classifieurs autorise déjà leur utilisation comme test binaire, avec une partie <*condition*> déclenchant l'action correspondante (branche "OUI") et la négation de la condition déclenchant l'autre branche (branche "NON"). Le rôle de **XCS5** est donc de sélectionner les classifieurs discriminant la base d'exemples de la manière la plus efficace (au sens d'une mesure de gain d'information) et de les placer au sommet de l'arbre de décision.

Un arbre produit par **XCS5** est un arbre de type inductif, dont un nœud représente un test T, une branche représente un résultat de ce test et une feuille représente l'une des classes des exemples à apprendre. Soient $P = \{C_0, \ldots, C_i, \ldots, C_n\}$, un jeu de règles obtenu à partir de **XCS-R** ou d'un quelconque système de classifieurs qui a convergé et C_i, l'un des classifieurs de P qui possède la forme suivante :

$$C_i := [c_{i,1}, \ldots, c_{i,n}] : A \tag{6.3}$$

$$:= [[\alpha_{i,1}; \beta_{i,1}], \ldots, [\alpha_{i,n}; \beta_{i,n}]] : A \tag{6.4}$$

Le type du test T est alors de l'un des quatre types présentés dans le tableau 6.2.

Type	**Description**
Type 1	L'exemple est comparé au classifieur complet C_i (activation ou non).
Type 2	La valeur de radiance correspondant à la bande j de l'exemple est comparée à la condition $c_{i,j}$ du classifieur C_i avec $j \in [1; n]$ (activation ou non).
Type 3	Identique au type 2, mais le résultat du test est ternaire (en-dessous de $c_{i,j}$, dans $c_{i,j}$, au-dessus de $c_{i,j}$).
Type 4	La valeur de radiance correspondant à la bande j de l'échantillon est comparée à la borne $\alpha_{i,j}$ ou $\beta_{i,n}$ de la condition $c_{i,j}$ du classifieur C_i (inférieure, supérieure).

TAB. 6.2 – Tests de l'algorithme **XCS5**.

Le fonctionnement de l'algorithme **XCS5** fait appel à certains fondements de l'algorithme **C4.5**, que nous rappelons ici. Soit E un ensemble d'exemples associant un pixel à $\kappa \in K$ la classe de ce pixel. Si E est partitionné en $k = |K|$ ensembles distincts $E_1, \ldots, E_k$, alors l'intérêt apporté par le partitionnement pour identifier la classe d'un exemple de E est $\text{Info}(E) = \text{I}(P)$, où P est la distribution de probabilité de la partition $(E_1, \ldots, E_k)$ et $I(P)$ l'entropie de P [Quinlan, 1993] :

$$\text{I}(P) = -\sum_{i=1}^{n} p_i * log(p_i) \tag{6.5}$$

où :

$$P = \left(\frac{|E_1|}{|E|}, \frac{|E_2|}{|E|}, \ldots, \frac{|E_k|}{|E|} \right) \tag{6.6}$$

Si E est partitionné en n sous-ensembles disjoints $E_1, \ldots, E_n$ sur la base d'un test donné T, l'information apportée par le test pour discriminer les exemples selon leurs classes est la moyenne des informations $\text{Info}(E_i)$, pondérées par la proportion de E_i par rapport à E :

$$\text{Info}(T, E) = \sum_{i=1}^{n} \frac{|E_i|}{|E|} * \text{Info}(E_i) \tag{6.7}$$

Enfin, le gain d'information fournit par l'application du test T est calculé par :

$$\text{Gain}(T, E) = \text{Info}(E) - \text{Info}(T, E) \tag{6.8}$$

Cette notion de gain est utilisée ensuite pour ordonner les tests selon leur pertinence. On affecte ensuite à un nœud le test avec le meilleur gain parmi l'ensemble des tests non encore considérés dans la branche située entre la racine de l'arbre et le nœud courant. L'algorithme A10 décrit l'élaboration complète de l'arbre de décision.

La notion de gain définie ci-dessus tendait à favoriser les tests binaires plutôt que ternaires. Le test de type 3 partitionne un ensemble en trois sous-ensembles dont l'un a une taille significativement plus faible que les deux autres. Quinlan a suggéré l'utilisation d'une mesure, utilisée dans **C4.5**, dont nous nous sommes inspirés dans **XCS5**. Le calcul du gain a été remplacé par la fonction GainRatio définie par :

$$\text{GainRatio}(T, E) = \frac{\text{Gain}(T, E)}{\text{SplitInfo}(T, E)} \tag{6.9}$$

où :

$$\text{SplitInfo}(T, E) = \text{I}\left(\frac{|E_1|}{|E|}, \frac{|E_2|}{|E|}, \ldots, \frac{|E_n|}{|E|}\right) \tag{6.10}$$

Une différence, par rapport à **C4.5**, doit être notée. Ici, les valeurs des tests ne sont connues que pendant l'apprentissage, contrairement à **C4.5** où les valeurs possibles pour chaque attribut sont connues avant l'apprentissage. Il peut arriver qu'un test ne renvoie qu'une seule valeur, même si les exemples d'apprentissage sont nombreux. Il n'est alors pas acceptable de renvoyer un arbre ne tenant pas compte des autres valeurs possibles du test. L'algorithme a été adapté en construisant des branches supplémentaires, pointant vers des feuilles dont la valeur correspond à la classe la plus fréquente observée sur le jeu d'exemples utilisés pour construire le nœud parent, en supposant que la proportion des classes intervenant dans cette partie de l'arbre est assez représentative des différents tests traversés.

Généralement, les arbres produits sont très simples, et puisqu'ils reprennent les règles obtenues avec **XCS-R**, ils ont une bonne capacité de généralisation. Il faut cependant noter que leurs performances sont légèrement inférieures à celles obtenues par les bases non post-traitées de **XCS-R** car de nombreux classifieurs spécifiques qui améliorent la qualité sont ici élagués. Néanmoins, les performances restent supérieures à celles de **C4.5** dans la plupart des cas.

6.4 Conclusion sur les post-traitements

Nous venons de voir, dans ce chapitre, trois types de post-traitements que nous avons développés et qui s'appliquent à des bases de règles produites par **XCS-R**. **XCS-R** a une propension à produire des bases de règles importantes donc difficiles à interpréter. Nous avons alors conçu un ensemble de post-traitements qui peuvent se révéler efficaces dans l'élaboration d'une représentation plus aboutie : une série d'algorithmes génétiques permet de simplifier directement la base de règles, l'algorithme **XCS5** permet d'en extraire un arbre de décision et l'étude d'une fonction d'appartenance permet la hiérarchisation des règles entre elles. Pour l'utilisateur, le choix de l'un ou l'autre de ces algorithmes dépend de la représentation qui lui paraît la plus adaptée pour le problème à résoudre. Le post-traitement génétique, comme les autres algorithmes, permet d'améliorer la lisibilité du jeu de règles, mais sans modifier sa structure. L'étude d'une fonction d'appartenance et l'algorithme **XCS5** proposent soit une hiérarchisation des règles, soit la construction d'un arbre de décision, qui sont des représentations plus faciles à comprendre. Ces différents algorithmes, ainsi que ceux présentés dans le chapitre précédent, sont tous illustrés sur des exemples choisis pour leur pertinence, dans le prochain chapitre.

ALGORITHME A10
EXTRACTION D'UN ARBRE DE DÉCISION DEPUIS UNE POPULATION DE CLASSIFIEURS

$\rightsquigarrow$ PARAMÈTRES - E est l'ensemble non vide des exemples d'apprentissage, E_e est l'ensemble des exemples de test pour l'élagage, R est la population de règles obtenue avec **XCS-R** et τ est un type de test parmi ceux du tableau 6.2.
$\rightsquigarrow$ RÉSULTAT - un arbre de décision
$\rightsquigarrow$ $E^{\mathtt{C}}$ est l'ensemble des classes de l'ensemble des exemples de E.
$\rightsquigarrow$ $\Omega(E)$ renvoie le nombre d'éléments de l'ensemble E.
$\rightsquigarrow$ TERMINAL(E) renvoie une feuille dont la valeur est la classe la plus fréquente dans l'ensemble des exemples de E.
$\rightsquigarrow$ ENRACINE(A,t,A_f) ajoute un nœud fils à la racine de l'arbre A dont la valeur est A_f et le label de l'arc entre la racine et le nouveau nœud est t.
$\rightsquigarrow$ ÉLAGUE(A) élague l'arbre en remplaçant tous les sous-arbres, dont l'erreur estimée sur le jeu de test E_e est supérieure à un seuil fixé par l'utilisateur, par une feuille correspondant à la valeur de la classe la plus fréquente du sous-arbre.

si $\Omega(E^{\mathtt{C}}) = 1$ *ou* $\Omega(R) = 0$ ***alors***
  Renvoyer TERMINAL(E)
fin si
$\rightsquigarrow$ La population de règles R est vue comme un ensemble de tests R_τ selon τ.
soit $T = \{T_i \mid \text{GAIN}(T_i, E) = \max \text{GAIN}(T_k, E)\ \forall\ T_k \in R_\tau\}$
soit $V_T = \{t_i | i = 1, 2, \ldots, n\}$ *les valeurs des tests* T
soit $\{E_i | i = 1, 2, \ldots, n\}$ *les sous-ensembles correspondants, partitionnés selon les valeurs* V_T *du test* T *appliqué sur les exemples de* E.
soit A *une feuille labellisée par* T
pour i *de 1 à* n ***faire***
  si $\Omega(E_i) = 0$ ***alors***
    A_i = TERMINAL(E)
  sinon si $\Omega(E_i) = \Omega(E)$ ***alors***
    $\rightsquigarrow$ E est un jeu d'exemples dont les classes sont éventuellement différentes, mais renvoyant tous la même valeur pour le test T
    A_i = TERMINAL(E)
  sinon
    A_i = XCS5$(R \setminus \{T\}, E_i, E_e, \tau)$
  fin si
  A = ENRACINE(A, t_i, A_i)
fin pour
Renvoyer ÉLAGUE(A)

Algorithme 10: *Fonction* XCS5(R, E, E_e, τ)

6.5 Synthèse des algorithmes proposés

Nous présentons dans le tableau 6.3 la liste des algorithmes que nous avons proposés dans les deux chapitres précédents, avec quelques informations synthétiques les concernant. Ces informations peuvent servir à comprendre les différences entre chacun de ces algorithmes et à en faciliter le choix par un utilisateur intéressé.

Le nombre important d'algorithmes provient surtout du fait qu'il y a de nombreux types de problèmes différents en classification. Le choix de l'algorithme à utiliser est donc, dans la plupart des cas, directement déterminé par le type des données brutes à traiter et de la question de savoir si une donnée brute peut être labellisée ou non par plusieurs classes. Les données entières ou binaires font appel à un type de classification dur ou doux et les données réelles à un type flou ou par intervalles flous. La représentation des règles intervient ensuite dans le choix de l'algorithme. D'après nos expérimentations, nous avons constaté que les représentations arborescentes, indiquées par les colonnes (1) et (2), sont plus simples à interpréter et à visualiser mais nécessitent un paramétrage plus complexe et sont parfois moins performantes que les représentations plates. La colonne (4) précise la représentation interne des règles (individus génétiques) : l'approche de Pittsburgh réunit plusieurs règles par individu tandis que l'approche de Michigan n'en réunit qu'une seule. La colonne (5) indique le type de résultat final obtenu pour chaque algorithme. Un résultat formé d'une population de règles pour toutes les classes est plus complexe à représenter (graphiquement ou non) et à interpréter (par un autre algorithme ou par un expert humain). La colonne (6) liste les paramètres les plus importants qui doivent être maîtrisés par l'utilisateur pour obtenir des résultats intéressants, en plus des paramètres génétiques classiques qui sont communs à tous les algorithmes. Enfin, la colonne (7) compare de manière approximative la durée d'exécution de ces algorithmes.

	(1) **Structure** (voir la section 4.4)	(2) **Format**	(3) **Type de problème de classification** (voir la section 4.2)	(4) **Approche** (voir la section 2.4)	(5) **Base de règles**	(6) **Paramétrage**	(7) **Durée CPU** (3 GHz)
ICU (p. 77)	Plate	Intervalles réels	Doux	Michigan	Une règle par classe	Taille des règles	Moyenne (30 min)
XCS-R (p. 83)	Plate	Intervalles réels	Doux	Michigan	Une population pour toutes les classes	Taille de la population	Longue (1h)
ICUX (p. 86)	Plate	Intervalles réels	A intervalles flous	Pittsburgh	Une règle pour toutes les classes	Taille des règles	Longue (1h)
ProgGen (p. 93)	Arborescente	Expression arborescente	Flou	Michigan	Une règle par classe	Taille des arbres, probabilités par opérateurs	Longue (2h)
GramGen (p. 92)	Arborescente	Expression arborescente	Flou	Michigan	Une règle par classe	Grammaire	Longue (4h)
Post-traitements génétiques (p. 105)	Plate	Intervalles réels	Doux	Michigan	Une population pour toutes les classes	Le paramétrage de XCS-R	Courte (10 min)
XCS5 (p. 116)	Arborescente	Arbre de décision	Dur	Michigan	Une règle pour toutes les classes	Le paramétrage de XCS-R	Courte (5 min)

TAB. 6.3 – Synthèse des algorithmes proposés.

Chapitre 7

Expérimentations et validations

7.1 Introduction

En télédétection, la validation des résultats fournis par les algorithmes est une étape essentielle : elle permet de confronter ces résultats avec les mesures au sol et éprouve les algorithmes avec des données réelles. Nous avons effectué de nombreuses expérimentations avec les algorithmes présentés dans les chapitres précédents. Pour ce chapitre, nous avons sélectionné celles qui illustrent au mieux le choix de tel algorithme plutôt qu'un autre, ainsi que celles qui sont les plus pertinentes pour chacune des représentations développées dans ce mémoire.

La visualisation de ces objets n'étant pas triviale, nous avons introduit plusieurs représentations graphiques permettant de faire le lien entre échantillons bruts, échantillons experts et bases de règles. Nous les présenterons en premier lieu, dans la section 7.2.

Les études de cas que nous exposons ensuite seront regroupées par type de problème de classification. Dans la section 7.3, des classifications de type *hard* et *soft* à partir de classifieurs à structure plate sont réalisées à l'aide des algorithmes **ICU** et **XCS-R**. En utilisant l'une des bases de règles obtenues avec **XCS-R**, nous essaierons de démontrer les avantages à utiliser un post-traitement génétique afin de réduire la taille de la base sans décroître la qualité de classification. Ensuite, la partie suivante sera consacrée à la classification de type floue à partir de classifieurs à structure arborescente, mettant en œuvre les algorithmes **ProgGen** et **GramGen**.

Le post-traitement génétique n'est pas le seul moyen de rendre une base de règles plus performante. L'optimisation par *tuning* des paramètres de **XCS-R**, dans la section 7.4, étudie l'influence du nombre de générations, de la taille du jeu de règles et de la *spécificité* des classifieurs de **XCS-R** par rapport au temps d'apprentissage, à la qualité du jeu de règles généré ou aux taux de faux positifs et faux négatifs.

La section 7.5 est réservée à la comparaison des algorithmes développés dans le cadre de cette thèse à d'autres algorithmes connus pour être robustes en classification. Notamment l'algorithme d'extraction d'un arbre de décision par **XCS5** est comparé à **C4.5** et l'algorithme de classification floue **ICUX** et comparé à une approche à base de réseaux de neurones et de machines à vecteurs supports.

Tous ces algorithmes produisent un ensemble de résultats assez conséquent. À titre de conclusion, la dernière section propose une technique basée sur un *vote consensuel* qui permet de combiner les résultats obtenus par les algorithmes que nous venons de citer. Le concept général est présenté, suivi d'un exemple combinant les résultats de huit algorithmes, dont deux non supervisés, à partir d'une image traitée dans le cadre du projet TIDE.

7.2 Visualisation des classifieurs

Malgré leur simplicité, les classifieurs restent tout de même des objets difficiles à appréhender, même pour un expert. De toutes les représentations possibles, la possibilité de visualiser directement ces classi-

fieurs représente sans doute la meilleure méthode de validation possible pour un être humain. Nous avons donc développé une série de trois méthodes de *visualisation scientifique*, procurant à l'utilisateur une description abstraite des règles de classification. Cette description peut être comparée aux données afin d'en tirer des conclusions intéressantes pour le raffinement des règles ou l'optimisation de l'apprentissage. Enfin, elle permet de rendre les classifieurs plus lisibles et contribue à leur compréhensibilité.

7.2.1 Analyse par spectrogramme

La création des individus génétiques est une étape difficile. Des techniques fiables sont nécessaires, au sein de l'opérateur d'initialisation, pour ne pas créer une population trop éloignée de la solution à trouver. Le *spectrogramme* est un outil que nous avons mis au point afin d'avoir une pré-analyse des données spectrales pour l'apprentissage. Il indique pour une classe et un attribut donné la répartition des valeurs de l'attribut pour les échantillons appartenant à cette classe. Les étapes de la création du spectrogramme sont les suivantes :

- pour chaque classe k donnée, déterminer l'ensemble E_k des échantillons bruts étiquetés de la classe k par l'expert,
- pour chaque échantillon $e_{k,i} \in E_k$ comprenant n attributs $\{ a_{k,i}(1), \ldots, a_{k,i}(j), \ldots, a_{k,i}(n) \}$, assombrir les pixels de coordonnées $(j; a_{k,i}(j))$ pour $j \in [1; n]$.

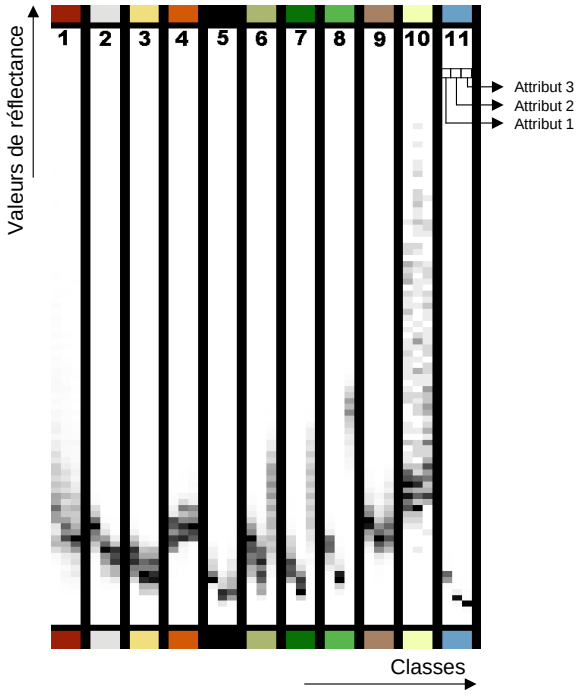


FIG. 7.1 – Spectrogramme d'un complexe sportif de Strasbourg (Stade Vauban) capturé par SPOT (3 canaux).

Les figures 7.1 et 7.2 présentent de tels spectrogrammes. Chaque colonne, séparée par une bordure noire, correspond à une classe, identifiée par la couleur de la légende experte et par un numéro. Ces colonnes sont elles-mêmes séparées en sous-colonnes, pour chacun des attributs correspondant. Pour les échantillons représentant des pixels spectraux, le spectrogramme d'une classe peut être vu comme les dessins successifs au crayon des spectres de chacun de ces pixels, les valeurs sombres correspondant donc aux valeurs les plus fréquentes. Le degré d'assombrissement est normalisé de telle sorte que le contraste

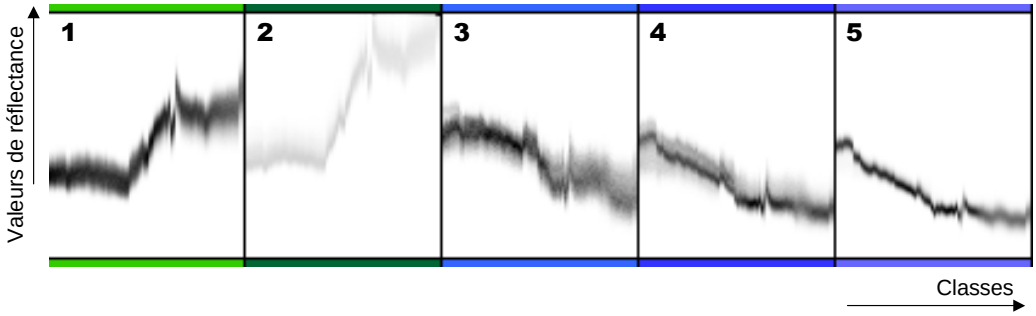


FIG. 7.2 – Spectrogramme de la zone de San Felice (Venise) capturée par ROSIS (80 canaux). Les deux premières classes sont des classes de végétation, les trois dernières sont des classes d'eau.

soit maximal sur tout le spectrogramme. Les points situés en bas correspondent à des valeurs d'attributs (radiances) faibles, les points du haut correspondent à des valeurs élevées.

Le spectrogramme permet d'apporter une réponse aux points suivants :

- il permet d'analyser la qualité de l'expertise et des données : les classes bruitées y apparaissent clairement (par exemple, la colonne 10 de la figure 7.1 représentant le béton apparaît plus diluée) ainsi que celles qui sont très corrélées (ainsi les colonnes 3, 4 et 5 de la figure 7.2),
- il autorise l'approximation de la répartition statistique des valeurs pour chaque classe, ce qui permet d'en tirer des conclusions utiles pour certains paramètres génétiques : par exemple, une répartition étroite invite à diminuer la fréquence des mutations et à consolider le seuil de convergence de l'algorithme car l'initialisation des individus sera proche de la solution à trouver,
- enfin, la validation de la représentation des règles est facilitée puisque l'on peut comparer d'une part, les proportions des valeurs de radiances des spectres des données de test et d'autre part, les positions correspondantes des contraintes spectrales des règles.

Sur la figure 7.1, nous pouvons voir les spectres les plus fréquents de onze classes, observés par le capteur SPOT. D'une manière générale, un ensemble de pixels noirs correspond à un spectre très ciblé (comme l'eau de la colonne n°11) et des traînées en gris clair correspondent à une variabilité spectrale plus importante (comme l'ombre de la colonne n°5, classe souvent confondue avec l'eau). Les spectres diffus peuvent correspondre à des classes bruitées ou bien à des classes composites comme les classes d'ombres qui recouvrent plusieurs régions aux spectres variés. De plus, une même bande n'offre pas forcément la même qualité selon la classe observée : sur la figure 7.2, l'une des bandes présente un artefact prononcé pour la classe n°1, largement atténué pour la classe n°5. Concernant la représentation des règles et leur validation, les spectres diffus doivent conduire les règles à généraliser (caractère *joker*, intervalles plus larges, ...). Quant aux spectres présentant des altérations, ils doivent conduire la règle à se complexifier (ajout de disjonctions d'intervalles, profondeur des arbres plus élevée, ...). Examinons plus précisément les colonnes n°5 et n°11 de la figure 7.1. Voici les règles associées, découvertes par **ICU** :

- Colonne n°5 (ombre) : [6 ;64] [3 ;21] [11 ;42] (évaluation : 95.38%)
- Colonne n°11 (eau) : [6 ;38] [0 ;27] [1 ;15] (évaluation : 95.58%)

D'après le spectrogramme, on remarque que pour l'eau la quasi-totalité des pixels des deux dernières bandes ont une valeur bien précise, alors que ce n'est pas le cas pour l'ombre. Ceci est très fortement visible dans les règles obtenues : l'intervalle de la dernière bande pour l'ombre est deux fois plus grand que celui de l'eau.

Le spectrogramme est toujours le même pour des couples d'échantillons bruts et experts donnés. Peu affecté par la taille des données (nombre d'échantillons et nombre de bandes spectrales), il se calcule très rapidement (moins d'une seconde). Puisqu'il donne une idée générale des règles que l'on va obtenir, nous l'avons utilisé pour valider les règles produites, en tant qu'indicateur pour les phases de pré-traitement

des données et comme composante de l'opérateur d'initialisation de **ICU** (voir la section 5.2.1.3).

7.2.2 Visualisation de la base de classifieurs

Pour les représentations à base de contraintes spectrales modélisées sous forme d'intervalles (**ICU** et **XCS-R**), nous avons développé l'algorithme de visualisation **RuleView**. Celui-ci fonctionne de la façon suivante :

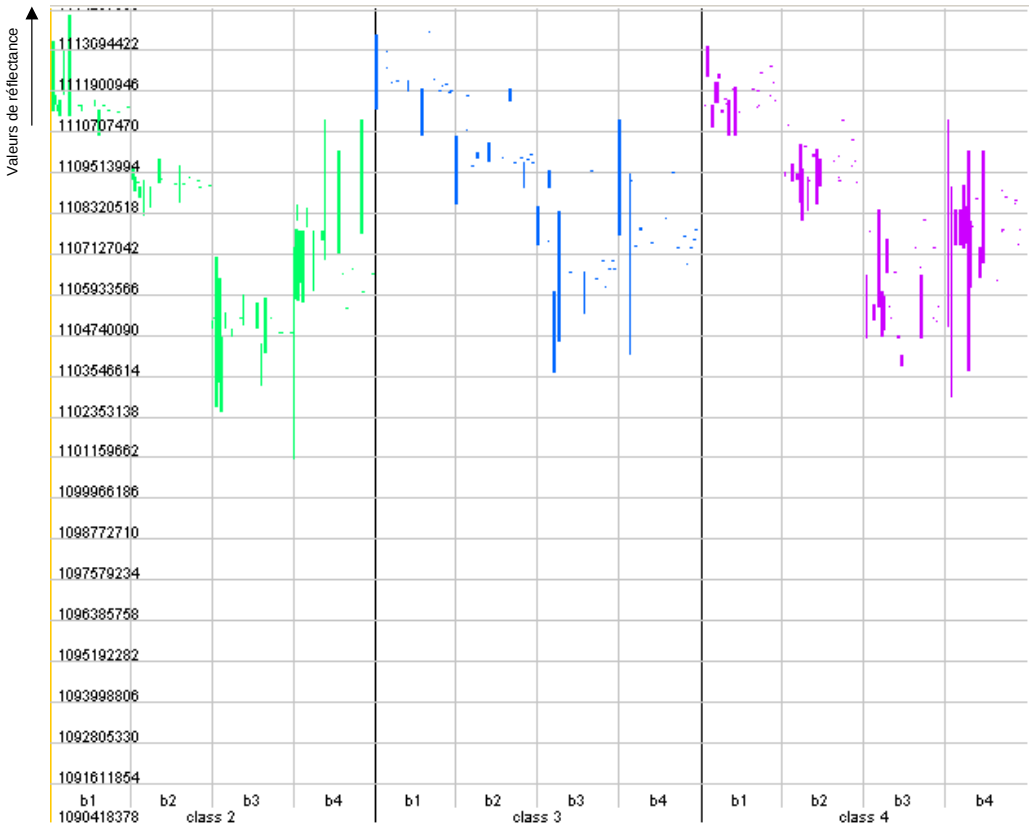


FIG. 7.3 – L'algorithme de visualisation **RuleView** appliqué sur la base de règles de **XCS-R** pour une image de QuickBird (4 canaux). Les noms des classes sont indiqués en abscisse.

- les règles sont divisées en plusieurs groupes, selon leur partie <*action*> (ici, il s'agit bien évidemment de la classe pour laquelle chaque classifieur travaille),
- les règles de chaque groupe sont alors découpées selon leurs différentes parties <*condition*> et chacune de ces conditions est affichée à un endroit bien précis du diagramme. Par exemple, sur la figure 7.3, la condition n°1 des règles s'activant pour la classe n°2 est affichée dans la colonne nommée « b1 » de l'ensemble « class 2 »,
- les conditions sont affichées en utilisant les bornes des intervalles qu'elles contiennent : un intervalle $[a; b]$ sera affiché par une ligne verticale depuis l'ordonnée a jusqu'à l'ordonnée b,
- les conditions concernant les mêmes attributs mais appartenant à des classifieurs différents sont présentées ensembles, mais avec des abscisses légèrement décalées de telle sorte que tout le jeu de règles puisse être virtuellement présenté sur la même image.
- ces conditions peuvent être triées par un critère au choix de l'utilisateur, par exemple la qualité des classifieurs (comme sur la figure 7.3) ou leur spécificité.

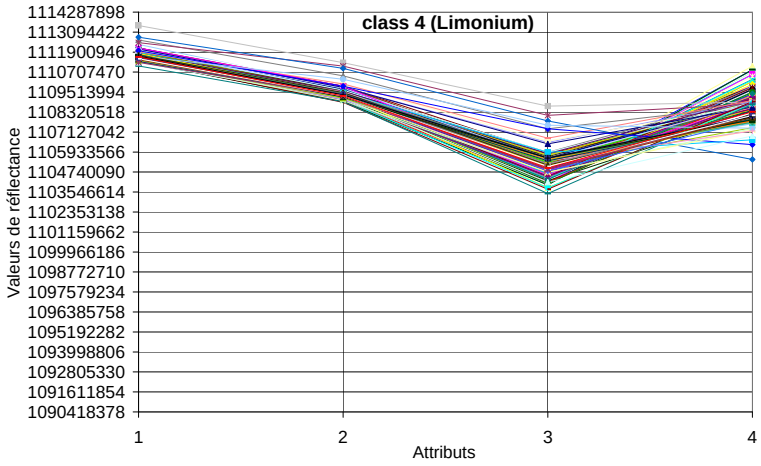


FIG. 7.4 – Spectres des échantillons de test pour la classe 4 (avec le capteur QuickBird), pour comparaison (en abscisse les canaux spectraux, en ordonnée les valeurs de radiance).

L'algorithme affiche uniquement les règles avec une qualité dépassant un certain seuil. Cette qualité est calculée d'après les paramètres des règles (par exemple, pour **XCS-R**, nous utilisons la *fitness* F et la prédiction de gratification p). Le seuil est défini par l'utilisateur selon qu'il souhaite visualiser le jeu de règles en entier ou uniquement les meilleures règles.

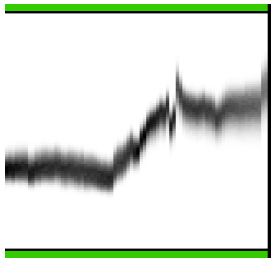
La figure 7.3 présente le résultat de l'algorithme pour un jeu de 6000 règles créées par **XCS-R**. Les intervalles imprimés par les règles doivent correspondre aux spectres des classes, ce qui permet de contrôler visuellement la base de règles, et de la comparer aux spectres réels. En effet, on peut remarquer que l'ensemble des spectres des échantillons de test (figure 7.4) ont une forte corrélation avec la visualisation du jeu de règles pour la classe 4. De plus, le regroupement des attributs par classe permet de donner une idée, pour chacun d'eux, de la proportion de règles spécifiques (matérialisées par un point ou un trait très court) par rapport aux règles qui ont été généralisées (matérialisées par un trait plus large). On peut remarquer que les conditions des règles ne sont pas toutes identiques : chaque règle modélise un sous-ensemble d'échantillons, l'essentiel étant que le jeu de règles complet soit fiable. Il faut néanmoins avoir une idée de l'échelle des données : par exemple, les valeurs de QuickBird sont définies sur 32 bits : l'espace de définition des données a une taille de 10^{10} donc chaque pixel du diagramme représente 50000 unités. Nous avons, dans certains cas, été contraints de *zoomer* sur le diagramme pour étudier les attributs séparément.

7.2.3 Analyse par confrontation

Alors que le spectrogramme affiche les relations entre les classes de l'expert et les spectres des échantillons bruts, que l'algorithme **RuleView** affiche les relations entre les classes de l'expert et le contenu des règles, les relations entre les échantillons bruts et le contenu des règles sont affichées à l'aide d'un troisième type de graphe, le *cover-graph*. Ce type de graphe est surtout indiqué pour les données hyperspectrales, car il apporte une vue globale sur un ensemble vaste de canaux. Il permet de valider les règles séparément ou simultanément, en les comparant au spectrogramme.

La partie gauche de la figure 7.5 montre la représentation d'une règle de **ICU** pour une classe de végétation avec le spectrogramme associé à ces données. La partie droite de la figure présente le *cover-graph* correspondant : la courbe inférieure représente les bornes *min* des conditions et la courbe supérieure représente leurs bornes *max*. Cette visualisation permet d'étudier les conditions des règles une à une, et classe par classe. En comparant le spectrogramme et le *cover-graph* de cet exemple, on peut noter que l'artefact des canaux 51 à 54 de ROSIS se répercute sur la règle car il est intégré à tous les échantillons.

Classe *Spartina*	
Règle	Condition
1	[289 ;1082]
2	[324 ;977]
3	[478 ;1176]
...	...
80	[461 ;1716]



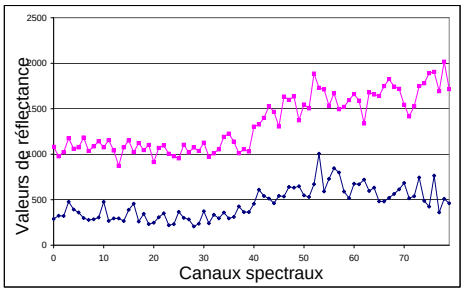
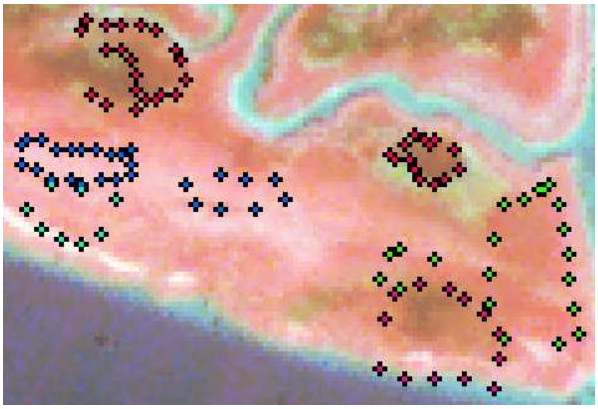


FIG. 7.5 – Comparaison entre la description textuelle d'une règle de **ICU**, le spectrogramme de la classe correspondante et le *cover-graph* de la règle (capteur ROSIS).

7.3 Études de cas

7.3.1 Classifications de type *hard* et *soft*

L'étude présentée dans cette partie a été menée en utilisant les données acquises durant la campagne de Septembre 2002, pour le projet TIDE [TIDE, 2005]. La figure 7.6(a) présente l'extrait choisi de San Felice (lagune de Venise), capturé par CASI (*Compact Airborn Spectrographic Imager*, 288 canaux). Il a une taille de 142x99 pixels, une résolution spectrale de 16 bits par pixels et une résolution spatiale de 1.3m. Ces données ont été choisies pour éprouver la capacité de généralisation des algorithmes, car les validations terrain ne sont pas nombreuses dans cette région. Les données contenaient beaucoup de *mixels*, nous les avons donc regroupées en six classes *virtuelles*, c'est-à-dire en compositions de spectres purs (voir la figure 7.6(b)). Nous avons retenu ce regroupement car ces classes ont une fréquence assez élevée sur le terrain. Les données expertes représentent 1540 points dont la moitié a été consacrée à l'apprentissage et l'autre moitié à la validation.



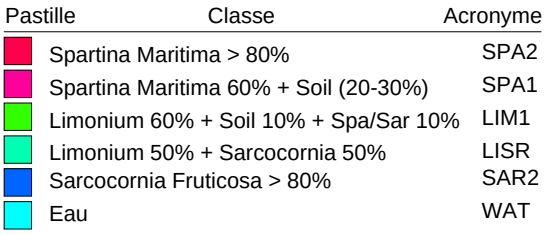


FIG. 7.6 – Sur la figure de gauche (a) : image hyperspectrale de CASI et les points utilisés pour le *ground-truthing*. Sur la figure de droite (b) : légende des 6 classes étudiées (San Felice).

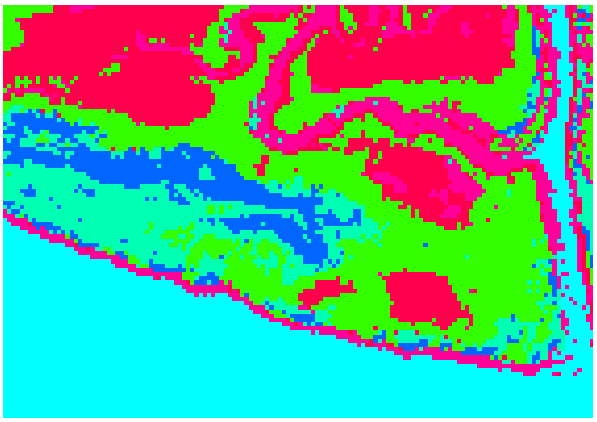
7.3.1.1 Classification avec ICU

Les résultats pour **ICU** sont présentés sur la figure 7.7. Les principaux paramètres que nous avons utilisés sont décrits dans le tableau 7.1. De nombreux tests ont été effectués pour optimiser les principaux paramètres génétiques (la taille du jeu de règles, le critère d'arrêt, les pourcentages de croisement et de mutation, les opérateurs de sélection et de remplacement ainsi que le paramètre d'élitisme). Dans certains cas, des boucles de tests imbriquées ont été mises en place pour sélectionner les meilleures

valeurs parmi des jeux comportant de trois à six valeurs considérées comme acceptables pour chacun des paramètres, en utilisant les mesures de qualité et de précision définies dans la section 4.6.1.1. Ces valeurs étant différentes pour chaque image analysée, nous pensons qu'une justification expérimentale plutôt que formelle du choix des paramètres peut être admise. Ces paramètres pourront être réutilisés sur d'autres images à caractéristiques semblables (même résolution spatiale et spectrale, contexte rural, ...) mais les boucles devront sans doute être ré-employées sur des images avec des caractéristiques différentes.

Paramètre	**Valeur**
Taille du jeu de règles	$\|P\| = 300$ individus
Critère d'arrêt	3000 générations
P_{cross}	0.70
P_{mut}	0.15
$P_{mut,band}$	0.35
$P_{mut,interval}$	0.20
$P_{mut,border}$	0.45
Opérateur de sélection	Ranking direct
Opérateur de remplacement	Ranking direct
Nombre d'enfants par génération	$\|P\|$
Elitisme	1% (fort)
Durée du traitement	8 min (CPU de 2.5 GHz)

TAB. 7.1 – Paramètres utilisés pour **ICU** sur CASI, voir la section 5.2.1.



		Nombre de pixels selon l'expert					
		SAR2	LISR	SPA2	SPA1	LIM1	Q_{ppa}
Nb de pix. selon les classifieurs	SAR2	222	0	0	0	5	98%
	LISR	59	149	0	0	21	65%
	SPA2	0	0	382	162	8	69%
	SPA1	0	0	0	28	0	100%
	LIM1	0	0	3	33	468	93%
	Q_{sens}	79%	100%	99%	13%	93%	

FIG. 7.7 – Image classifiée et matrice de confusion pour **ICU**.

Nous devons signaler que nous n'avons pas utilisé l'eau en apprentissage (la zone sombre au sud). Pour ces données, l'eau peut être discriminée rapidement en utilisant des outils statistiques ou même une approche non supervisée, et de toute façon n'intéressait pas l'expert sur cette image. *SPA2* est une classe pure de *Spartina Maritima*, une plante marine longue, souple et très fine, dont l'aspect laisse apercevoir le sol nu ou l'eau, ce qui en fait une classe difficile à traiter (voir la figure 7.8). *SPA1* est une classe contenant de la *Spartina* en quantité beaucoup plus faible. Globalement, l'algorithme **ICU** est complètement conforme à l'expertise. Le seul point pour lequel **ICU** n'est pas en accord est la tache située au nord-est de la berge, qu'il classe en *SPA1* plutôt qu'en *SPA2* (il s'agit de la tache à droite, celle de gauche ne faisant pas partie de l'expertise). Cela est dû au fait que les spectres sont très similaires. Les classifications obtenues sont correctes (le κ-index est de 0.81, la qualité Q_{accur} est de 81.1%).

La figure 7.9 montre la carte de recouvrement. Les pixels correspondants à des zones détectées comme étant mixtes par l'algorithme sont affichés en grisé, comme la tache de *SPA2* en bas de l'image. De plus,

FIG. 7.8 – *Spartina Maritima.*

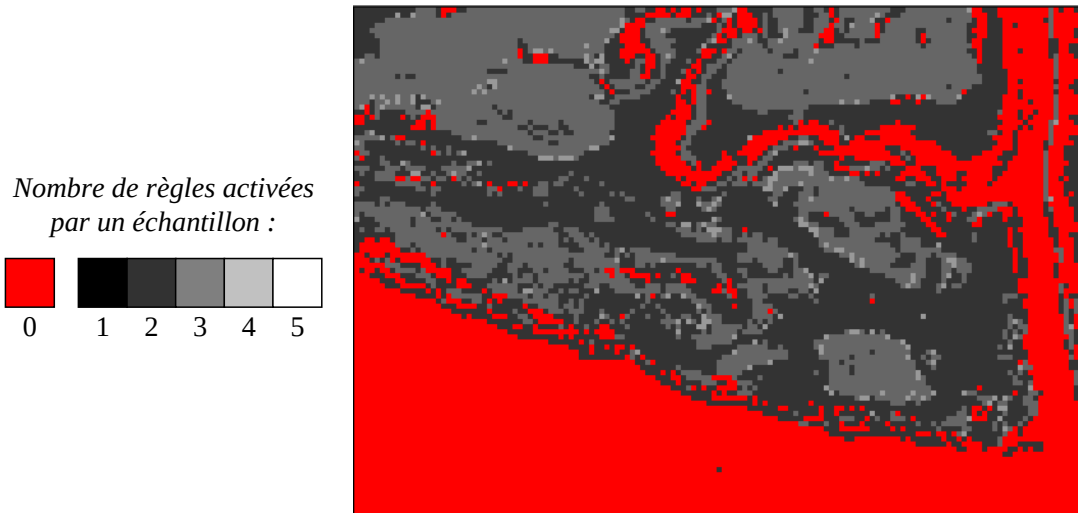


FIG. 7.9 – Carte de recouvrement pour la lagune de Venise.

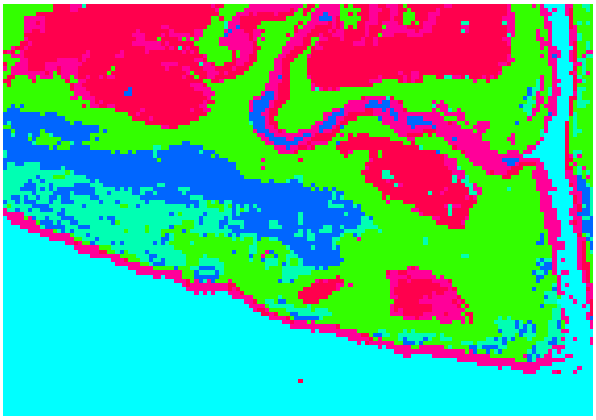
les pixels correspondants à des zones non apprises sont affichés en rouge (comme c'est le cas pour l'eau par exemple, ce qui permettrait à l'expert de les situer et de les inclure lors d'un apprentissage ultérieur.

7.3.1.2 Classification avec XCS-R

La classification obtenue avec **XCS-R** et la matrice de confusion correspondante sont présentées sur la figure 7.10, et les paramètres dans le tableau 7.2. Là encore, des boucles de tests imbriquées ont été utilisées pour sélectionner les meilleurs paramètres en fonction de mesures de qualité et de précision. Cette classification est très semblable à celle obtenue pour **ICU**, incluant le bord du marais, pourtant incrusté de nombreux *mixels*. On peut aussi noter la grande précision de **XCS-R** lors de la délimitation des zones de *ground-truthing* : globalement, les frontières des classes correspondent au parcours pédestre initial de l'expert sur le terrain, consistant à valider les zones frontalières. L'étude de la matrice de confusion montre que **XCS-R** est meilleur pour la séparation des classes *SPA1* et *SPA2*, sans doute grâce au jeu de règles plus important. Finalement, la qualité globale est très bonne (κ-index de 0.88, Q_{accur} de 87.7%).

Paramètre	**Valeur**
Taille du jeu de règles	$\|P\| = 2300$ individus
Critère d'arrêt	500000 générations
Modifications aléatoires (*cover-rand*)	10^5
Fréquence d'appel de l'AG (θ_{GA})	48
Fréquence de généralisation (*dontCareProb*)	0.60
Probabilité de croisement	$\chi = 1.0$
Probabilité de mutation	$\mu = 0.04$
Opérateur de sélection	Tournoi n-aire
Taille du tournoi	0.4 $\|P\|$
Taux d'apprentissage	$\beta = 0.2$
Paramètres de la fonction de qualité	$\alpha = 0.1$, $\varepsilon = 0.001$
Facteur pour la gratification	$\gamma = 0.9$
Durée du traitement	41 min (CPU de 2.5 GHz)

TAB. 7.2 – Paramètres utilisés pour **XCS-R** sur CASI [Butz et Wilson, 2002; Wilson, 1995].



		Nombre de pixels selon l'expert					
		SAR2	LISR	SPA2	SPA1	LIM1	Q_{ppa}
Nb de pix. selon les classifieurs	SAR2	266	12	0	0	3	95%
	LISR	11	133	0	0	2	91%
	SPA2	0	0	353	83	1	81%
	SPA1	0	0	27	106	3	78%
	LIM1	4	4	5	34	493	91%
	Q_{sens}	95%	89%	92%	48%	98%	

FIG. 7.10 – Image classifiée et matrice de confusion pour **XCS-R**.

7.3.2 Réduction par post-traitements

Dans l'étude précédente, nous avions précisé que le système **XCS-R** nécessite un jeu de règles assez important, afin d'avoir une qualité acceptable. Ainsi, entre 2000 et 3000 règles, toutes actions confondues, sont nécessaires pour avoir une qualité supérieure à 80%. Nous rappelons que cet algorithme est capable

de découvrir, par lui-même, le nombre de règles nécessaires à chacune des classes, ce qui lui permet de ne consacrer que quelques-unes des règles aux cas les plus triviaux et le reste du jeu pour les autres classes, plus complexes. Pour traiter chaque classe complexe d'un problème comprenant cinq classes, les besoins en règles de **XCS-R** pouvaient atteindre 30% du jeu, par rapport à une moyenne de 20%. Les règles n'étant pas pourvues de disjonctions, elles devaient modéliser les groupes d'échantillons presque individuellement, ce qui accroît la taille du jeu de règles (en effet, deux disjonctions spectrales nécessitent quatre règles distinctes). De plus, le fait que **XCS-R** tente de trouver une représentation complète du problème (ang., *complete mapping*) l'oblige à consommer encore plus de règles, même si elles ne sont pas nécessaires.

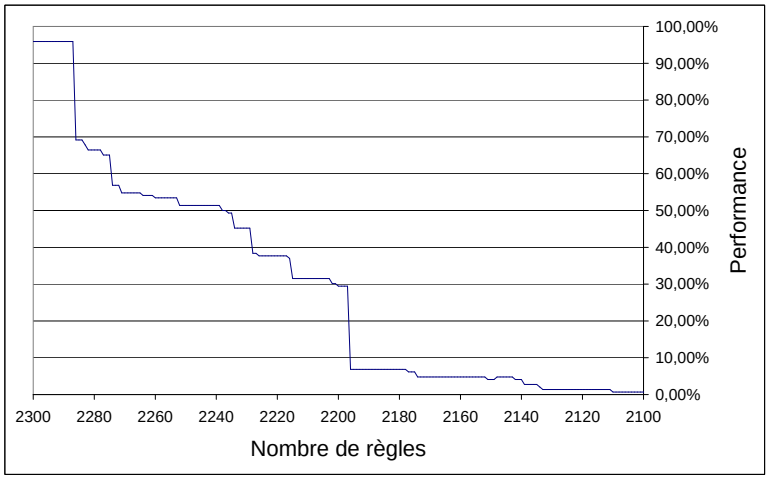


FIG. 7.11 – Perte de la performance par réduction aléatoire du jeu de règles de **XCS-R**.

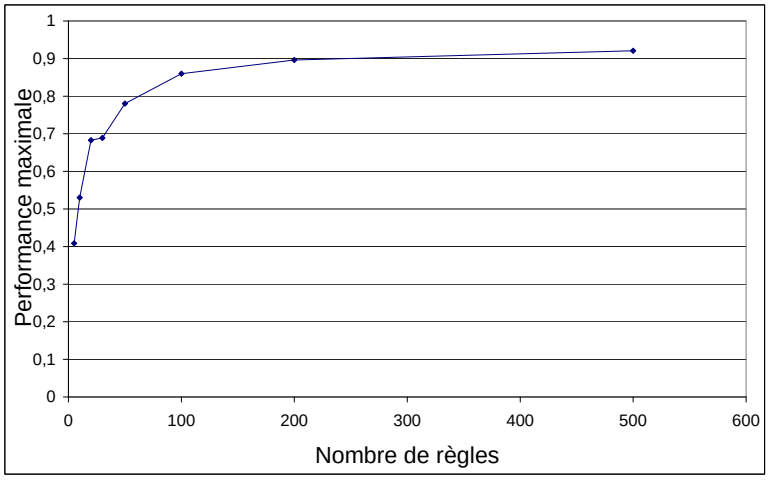


FIG. 7.12 – Performances des jeux de règles obtenus après une réduction optimisée par algorithme génétique des résultats initiaux de **XCS-R**. Comparaison de jeux de 5, 10, 20, 30, 50, 100, 200 et 500 règles.

Nous avons alors voulu savoir si le jeu de règles obtenu par **XCS-R** était robuste, c'est-à-dire si une qualité similaire en classification pouvait être atteinte avec un jeu réduit. Le test a consisté tout d'abord à analyser la perte de performance occasionnée par une suppression aléatoire des règles, réduisant le jeu

de 2300 à zéro règles. La mesure de performance est basée sur la mesure Q_{accur} définie dans la section 4.6.1.1, calculée sur des ensembles de test. 1000 expérimentations ont été effectuées et la figure 7.11 montre un exemple de perte typique qui a été observée entre 2300 et 2100 règles. On se rend compte que la perte est déjà de 50% après une suppression de seulement 60 règles, soit moins de 3% du jeu de règles initial. Ce graphique est à comparer avec la figure 7.12. Nous avons utilisé les principes de post-traitements par algorithme génétique édictés dans la section 6 : différentes expérimentations ont été conduites avec des génomes représentant 5, 10, 20, 30, 50, 100, 200 et 500 règles. Chaque extrait consiste en une sélection de règles, optimisée par algorithme génétique, formant une sous-population de la population initiale. On remarque que dans le meilleur des cas, on atteint une performance de 80% rien qu'en choisissant correctement une cinquantaine de règles, ce qui démontre l'intérêt de la sélection génétique par rapport à une sélection aléatoire. Pour obtenir des statistiques plus précises, 1000 expérimentations génétiques ont été conduites sur des jeux de 500 règles. Nous avons alors pu obtenir, en moyenne, 99% de jeux dont la performance se situe delà de 85%. Les résultats obtenus sur un sous-ensemble de règles sont équivalents, voir meilleurs, que sur l'ensemble complet pour plusieurs raisons. Tout d'abord, la présence de règles trop spécifiques perturbait la performance sur les jeux de test (sur-apprentissage). Ensuite, il est possible que certaines règles représentent du bruit : en les supprimant, on peut améliorer la capacité de discrimination générale de l'ensemble de règles.

Avant optimisation	
Classe	Nombre de règles
1	200
2	500
3	528
4	509
5	563
Total	2300
Perf. en test	0.890

Après optimisation	
Classe	Nombre de règles
1	4
2	4
3	4
4	4
5	4
Total	20 (-99%)
Perf. en apprentissage	0.836 (-6%)
Perf. en test	0.793 (-11%)

FIG. 7.13 – Comparaison entre un jeu de règles initial obtenu avec **XCS-R** sur des données hyperspectrales et le même jeu de règles post-traité en utilisant une technique à base d'AG.

En utilisant les autres types de génomes présentés dans la thèse, nous avons pu réaliser l'expérience présentée dans les tableaux de la figure 7.13. De nouvelles règles ont été construites par raffinement d'un jeu de règles appris de **XCS-R**, en utilisant la transformation et la recombinaison. Nous obtenons donc une perte de qualité limitée à 11% pour une réduction de plus de 99% du jeu de règles. Le post-traitement génétique est donc un atout très efficace pour contourner la gourmandise en règles des systèmes de classifieurs.

7.3.3 Classification de type floue

Nous nous intéressons, dans cette partie, à la validation du formalisme des règles permettant de résoudre des problèmes de classification floues. Dans ces problèmes, les classes à apprendre doivent être modélisées, par la représentation, sous forme de fonctions associant un échantillon à une note continue. La note est sensée indiquer la proportion d'une classe donnée dans cet échantillon. Le problème est donc vu comme un problème de régression plutôt que de classification. Les représentations adéquates pour ces règles sont des arbres dont l'opérateur situé au sommet est capable de produire une valeur réelle. Dans **ICUX**, il s'agit de l'opérateur Match() modélisant cette valeur comme une distance entre la règle et l'échantillon. Dans **ProgGen** et **GramGen**, la valeur est produite par un opérateur mathématique situé au sommet de l'arbre de calcul.

Nous avons cherché à retrouver deux indices concernant la composition d'un *mixel* en végétation. Le premier indice, très connu en télédétection, est l'indice *NDVI* (ang., *Normalized Difference Vegetation*

Index) :

$$NDVI = \frac{C_1 - C_2}{C_1 + C_2} \tag{7.1}$$

Cet indice permet, dans certains cas, l'estimation de la composition en végétation d'échantillons spectraux. Cette équation utilise le fait que le rayonnement solaire est réfléchi plus fortement par une végétation en pleine croissance dans le proche infrarouge que dans des longueurs d'onde plus courtes situées dans la partie visible du spectre [Mohr, 1999]. Généralement, avec l'instrument *VEGETATION* de SPOT, on utilise XS3 pour la variable C_1 et XS2 pour la variable C_2. Dans un contexte hyperspectral, les canaux sont à déterminer au cas par cas, en fonction des longueurs d'onde. Cet indice est étudié ici sur des données multispectrales (SPOT, 3 canaux, 1999) et hyperspectrales (CASI, 288 canaux, 2003).

Le second indice n'existe pas, à notre connaissance, sous une formulation standard. Nous le nommerons I_{Lim} et il consiste à déterminer la proportion dans le sol d'une plante connue sous le nom de *Limonium Narbonense*, plus communément appelée *Lavande de mer*, typique des marais et des sols argileux. Nous nous sommes intéressé à la découverte de cet indice car il s'agit d'une classe d'intérêt dans le cadre du projet TIDE. Pour cette étude, nous avons utilisé les données de MIVIS (20 canaux) car les relevés des compositions pour cette espèce de végétation et pour la date de survol du satellite (juillet 2003) comprennent des valeurs de pourcentages variées.

7.3.3.1 Indice *NDVI* multispectral

Nous utilisons dans cette étude **ProgGen**, l'algorithme à base de programmation génétique sans grammaire, puisqu'utiliser une grammaire dans ce cas serait trop trivial. Le but de l'algorithme est de déterminer la formulation de l'indice en fonction d'un jeu d'opérateurs de calcul et d'une contrainte souple sur le nombre de nœuds. L'algorithme choisira de respecter ou non cette contrainte en fonction de l'évaluation des arbres engendrés. Selon des expérimentations mettant en jeu des boucles de tests imbriquées, les paramètres sélectionnés en fonction de la qualité obtenue sur ces données sont présentés dans le tableau 7.3. L'opérateur de division utilisé, tout comme ceux des études qui vont suivre, est un opérateur de division protégée. Ce jeu d'opérateurs a été choisi car il correspond aux opérateurs déjà utilisés dans les formules d'indices habituels des géographes.

Paramètre	**Valeur**
Échantillonnage	100 exemples ($< 1\%$ des données)
Taille du jeu de règles	$\|P\| = 200$ individus
Taille des arbres	5 à 10 nœuds
Critère d'arrêt	200 générations
Opérateurs	{+, -, *, /, somme à arité N (opSOMM), valeur absolue (opABS)}
Terminaux	{constantes (opCST), canaux spectraux (opARG)}
P_{mut}	0.15
P_{cross}	0.70
Opérateur de sélection	Ranking direct
Opérateur de remplacement	Ranking direct
Nombre d'enfants par génération	$\|P\|$
Elitisme	1% (fort)
Durée du traitement	7 min (CPU de 2.5 GHz)

TAB. 7.3 – Paramètres utilisés pour **ProgGen** sur SPOT, voir la section 5.3.2.

La formule qui fut finalement trouvée est présentée dans l'équation 7.2. Cette formule se réduit à l'indice, après simplification :

$$f_{SPOT} = \frac{1.0654 + x_3 - |1.0654 + x_2|}{|x_3 + x_2|} \tag{7.2}$$

7.3.3.2 Indice *NDVI* hyperspectral

Le même algorithme a été appliqué sur une image de CASI. Cette image présente la particularité d'avoir un nombre de canaux plus élevé, dont beaucoup étaient corrélés, laissant à l'algorithme le soin de choisir les canaux C_1 et C_2 de l'équation 7.1. Les paramètres utilisés sont présentés dans le tableau 7.4.

Paramètre	**Valeur**
Échantillonnage	9000 exemples (1% des données)
Taille du jeu de règles	|P| = 200 individus
Taille des arbres	10 à 20 nœuds
Critère d'arrêt	700 générations
Opérateurs	{+, -, *, /, somme à arité N (opSOMM), valeur absolue (opABS)}
Terminaux	{constantes (opCST), canaux spectraux (opARG)}
P_{mut}	0.15
P_{cross}	0.70
Opérateur de sélection	Ranking direct
Opérateur de remplacement	Ranking direct
Nombre d'enfants par génération	|P|
Elitisme	1% (fort)
Durée du traitement	40 min (CPU de 2.5 GHz)

TAB. 7.4 – Paramètres utilisés pour **ProgGen** sur CASI, voir la section 5.3.2.

L'augmentation du nombre de canaux nous a conduits à augmenter le nombre de générations et la taille souhaitée des arbres. La formule qui a été créée par l'algorithme est la suivante :

$$f_{CASI} = 2.2526 \cdot \frac{2.2526 \cdot x_{11} - 2.2526 \cdot x_8 - x_4}{6.4977 \cdot x_{12} + 6.4977 \cdot x_8 - 6.2888} \tag{7.3}$$

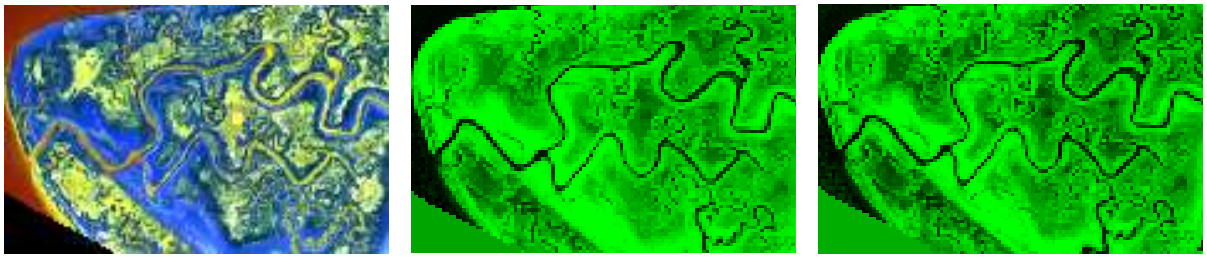


FIG. 7.14 – Sur la figure de gauche (a) : extrait de CASI. Sur la figure du milieu (b) : présence de végétation selon *NDVI*. Sur la figure de droite (c) : présence de végétation selon f_{CASI}.

Il est difficile d'estimer la valeur de cette formule par comparaison directe avec *NDVI*. Nous avons alors comparé les résultats des deux indices. La figure 7.14 montre un extrait d'une image de CASI avec le calcul de ces deux indices. Nous constatons que les images sont très proches : la corrélation entre l'indice *NDVI* et f_{CASI} est très bonne (C = 0.986). Pourtant, cette formule contient quatre variables différentes plutôt que deux. L'indice qui devait être imité est le suivant :

$$NDVI_{CASI} = \frac{x_{11} - x_3}{x_{11} + x_3} \tag{7.4}$$

Nous constatons donc une confusion entre x_{11} et x_{12} d'une part, et x_3 et x_8 d'autre part. Les corrélations entre ces canaux sont cependant très fortes (respectivement 0.868 et 0.997), ce qui explique la confusion. Finalement, les formules trouvées pour SPOT et CASI sont plutôt courtes et fiables à la réalité. Les formules exhibées par la programmation génétique, même si elles ne sont pas exactes, peuvent toujours être substituées à l'indice *NDVI*, par exemple dans le cas où nous chercherions d'autres canaux que ceux qui sont standards pour obtenir le même résultat (contournement de canaux bruités, ...).

7.3.3.3 Indice I_{Lim} multispectral

Les valeurs de composition en *Limonium* des échantillons (*mixels*) sont beaucoup plus variées sur les données de MIVIS que les données de CASI ou de SPOT. La figure 7.15 présente la répartition statistique des échantillons en fonction de leur composition en *Limonium* ou d'une autre classe pour les contre-exemples. Nous disposions de 975 exemples, dont 50% d'exemples contenaient une proportion dominante de *Limonium* et 60% en contenaient au moins 5%. Les contre-exemples sont principalement de l'eau et du sol mais proviennent de toutes les classes. L'image MIVIS comprend 20 canaux spectraux et chaque *mixel* a une résolution de 2.6 m^2.

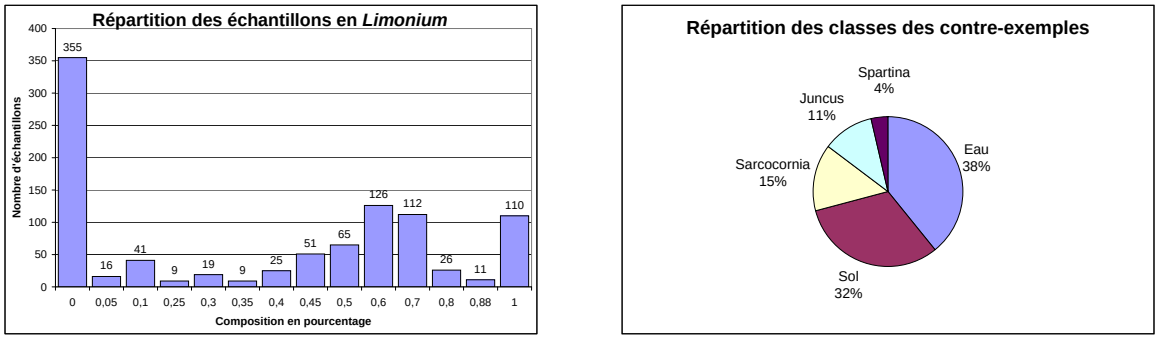


FIG. 7.15 – Jeu de données pour l'indice I_{Lim}. Figure de gauche : répartition des exemples en fonction de la quantité de *Limonium* dans les échantillons. Figure de droite : répartition des classes des contre-exemples.

Nous avions donc une base propice à l'apprentissage de l'indice I_{Lim} puisqu'il pourra être validé pour des proportions variées. Il est à noter qu'une telle base est plutôt rare puisque chaque échantillon a dû être validé à la main, en accord avec les données spectrométriques au sol et l'analyse visuelle des photographies des vérités terrain (*ground-truthing*). De plus, ce type de base ne faisait pas partie des objectifs principaux du projet TIDE.

La découverte d'un tel indice n'étant pas triviale, nous avons préféré contraindre l'espace de recherche par une grammaire. L'algorithme **GramGen** a donc été utilisé à cette fin. Nous avons mené plusieurs expérimentations avec plusieurs grammaires. Le paramétrage général est présenté dans le tableau 7.5.

Pour les expérimentations présentées ici, nous avons utilisé quatre grammaires engendrant des arbres de différentes tailles. Les trois premières produisent des arbres ayant des profondeurs respectives situées dans les plages $[2;3]$, $[2;4]$ et $[3;5]$. Elles présentent la particularité de n'utiliser que des opérateurs et des canaux spectraux et donc de ne pas utiliser de constantes. En effet, l'utilisation de constantes tend souvent à présenter l'algorithme en situation de sur-apprentissage et rend les formules découvertes plus dépendantes du jeu de données que celles sans constantes. Il serait par exemple envisageable de récupérer les longueurs d'onde des canaux pour appliquer ces formules sur une autre image. Bien entendu, leur qualité est souvent meilleure, d'où l'expérimentation avec la quatrième grammaire avec laquelle nous avons autorisé l'algorithme à créer des constantes aléatoires. Cette grammaire est présentée à titre d'illustration ci-dessous :

Paramètre	**Valeur**
Échantillonnage	100 exemples (10% des données)
Taille du jeu de règles	|P| = 250 individus
Taille des arbres	10 à 30 nœuds
Critère d'arrêt	500 générations
Opérateurs	{+, -, *, /, valeur absolue (opABS), opINV, opSQ, opINF, opSUP, *If ... Then ... Else*, opINTER, opRND, opEMPTY, opPOP, opPUSH, opSOMM}
Terminaux	{constantes (opCST), canaux spectraux (opARG)}
P_{mut}	0.15
P_{cross}	0.60
Opérateur de sélection	Ranking direct
Opérateur de remplacement	Ranking direct
Nombre d'enfants par génération	|P|
Elitisme	1% (fort)
Durée du traitement	10 à 50 min (CPU de 2.5 GHz)

TAB. 7.5 – Paramètres utilisés pour **GramGen** sur MIVIS, voir la section 5.3.2.

```
# Start symbol
S -> EQ24

# Définition d'une équation simple (profondeur 1 à 2, de 1 à 3 noeuds)
EQ12 -> Arg | OP Arg Arg

# Définition d'un arbre de profondeur 2 à 3 avec 3 à 7 noeuds
EQ23 -> OP EQ12 EQ12

# Définition d'un arbre de profondeur 2 à 4 avec 3 à 15 noeuds
EQ24 -> EQ23 | OP EQ23 EQ23

# Définition des opérateurs et des arguments
OP -> opADD | opSUB | opMUL | opDIV | opABS
Arg -> opARG | opCST
```

Les symboles à gauche des flèches sont les symboles non terminaux des arbres génotypiques, ceux à droite sont des symboles terminaux ou non qui sont utilisés pour la dérivation. Les symboles terminaux correspondant à des fonctions (opérateur de nœud) sont toujours suivis par les symboles qui représentent leurs arguments. Pour simplifier la notation, et sans perte de généralité, il ne peut y avoir qu'un seul opérateur de nœud par disjonction grammaticale. Par exemple, le terme non ambigu « `opMUL opARG opCST` » dénote une multiplication entre l'un des attributs d'un échantillon (opARG) et une constante instanciée par **GramGen** (opCST).

Les différentes formules produites sont présentées dans le tableau 7.6, avec le résultat de la fonction d'évaluation (comprenant l'évaluation sur le jeu d'apprentissage et le respect des contraintes de tailles, fixées par l'utilisateur), leur qualité sur le jeu de données de test (mesurée par la corrélation par rapport aux compositions attendues) et le nombre de générations au bout desquelles elles ont été découvertes.

On note l'importance des canaux b_{16} (740 nm) et b_{19} (800 nm) pour I_{Lim} (ils apparaissent huit fois dans les formules). Une très bonne corrélation est observée pour les deux premières formules, malgré leur simplicité. La première formule, par exemple, est obtenue très rapidement, en dépit du faible nombre

	Formule	Évaluation	Corrélation	Génération
1	$\frac{b_{19}-b_{14}}{b_{17}}$	0.859	0.802	14
2	$\frac{b_{15}\cdot(b_{19}-b_{16})}{b_1\cdot b_6}$	0.904	0.876	224
3	$\frac{b_{19}\cdot b_{15}}{(b_2-b_9)+(b_8\cdot b_{16})} - \frac{(b_2-b_9)+(b_8\cdot b_{16})}{b_{19}+(b_8\cdot b_{16})}$	0.668	0.814	463
4	$\frac{b_{17}-b_{15}}{b_4\cdot 0.9845} - \frac{0.938402}{b_{17}-b_{18}}$	0.694	0.858	94

TAB. 7.6 – Formules approximant l'indice I_{Lim}, avec quelques paramètres caractéristiques.

d'exemples dans l'ensemble d'apprentissage. L'algorithme **GramGen** est donc capable de découvrir des formules concises, relativement expressives pour l'expert (car elles ressemblent beaucoup à l'indice *NDVI*) et dans un temps relativement court. Surtout que l'espace des données est très vaste : par exemple, pour un arbre contenant cinq arguments au niveau des feuilles, comme la formule numéro 2, il faut tester 20^5 soit plus de 3 millions de combinaisons, sans compter celles des opérateurs. Cependant, nous n'avons testé que des opérateurs de base et des grammaires plutôt simples. L'utilisation d'opérateurs spécifiques au domaine d'étude encouragerait sans doute l'amélioration de ces résultats.

7.4 Complexité de l'apprentissage et tuning

Dans cette partie, nous nous intéressons essentiellement à l'étude de la complexité de l'apprentissage de **XCS-R**. Nous avons vu dans les sections précédentes que ce système de classifieurs était un algorithme plutôt gourmand, en temps, en mémoire, et pour la taille de la base de règles, par rapport à **ICU**. Nous avons donc cherché à connaître l'influence de certains paramètres sur le temps et la qualité de l'apprentissage. Cette section présente nos résultats sur les données de QuickBird (324 exemples et 4 attributs) sur une machine cadencée à 2.5GHz.

7.4.1 Temps d'apprentissage

La durée importante d'apprentissage est souvent considérée comme un défaut majeur en algorithmique évolutionnaire. Les deux plus importants paramètres de **XCS-R** concernant le temps d'apprentissage sont le nombre de générations (c'est-à-dire le nombre de cycles de l'algorithme génétique, N_G) et le nombre de classifieurs ($T_{[P]}$), qui représente aussi la taille du jeu de règles final. Compte tenu du rôle joué par les différents ensembles de classifieurs (*population set*, *action set*, ...) qui peuvent avoir une taille variable au cours de l'apprentissage, il n'était pas évident de calculer directement la complexité théorique pour N_G et $T_{[P]}$. Nous avons alors tenté de la déterminer de manière empirique. Nous avons obtenu un temps d'apprentissage linéaire pour la taille de la population, si l'on fixe le nombre de générations (voir la figure 7.16). Le temps d'apprentissage, en millisecondes, a été estimé à $26.2T_{[P]} + 3400$ avec un coefficient de détermination de $R^2 = 0.98$ (pour $N_G = 20000$).

Pour chaque valeur possible de N_G, nous avons calculé les lois linéaires dont les coefficients de détermination sont minimaux. Nous avons remarqué que leurs coefficients directeurs et leurs ordonnées à l'origine sont eux-mêmes linéaires en N_G. Nous en avons déduit une approximation (avec une erreur moyenne de 40%) du temps de calcul, linéaire en $N_G \cdot T_{[P]}$:

$$t = \frac{N_G \cdot T_{[P]}}{1934} + 79,52 \cdot T_{[P]} + 0,4055 \cdot N_G - 23155 \tag{7.5}$$

7.4.2 Qualité de l'évolution

L'étude plus précise de certaines mesures durant l'apprentissage de **XCS-R** est riche d'enseignements. La figure 7.17 présente l'évolution typique du pourcentage de règles correctes. Nous voyons qu'une qualité d'apprentissage supérieure à 0.9 est obtenue à partir de 200000 générations. Il est clair qu'un nombre de

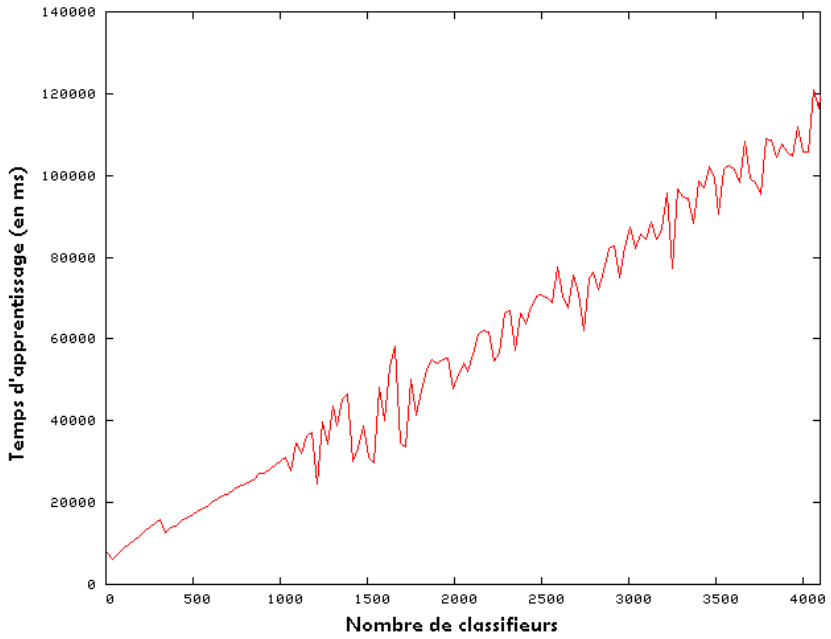


FIG. 7.16 – Temps d'apprentissage de **XCS-R** en fonction de la taille du jeu de règles $T_{[P]}$ (pour $N_G = 20000$).

générations trop faible est à proscrire, mais à l'inverse, un nombre de générations trop élevé peut conduire l'algorithme à apprendre par cœur. Ici nous voyons que la qualité n'atteint jamais 100%, car elle est bornée par l'introduction de nouvelles règles dans la population. Le renouvellement de cette population est donc constant et correctement contrôlé par **XCS-R**.

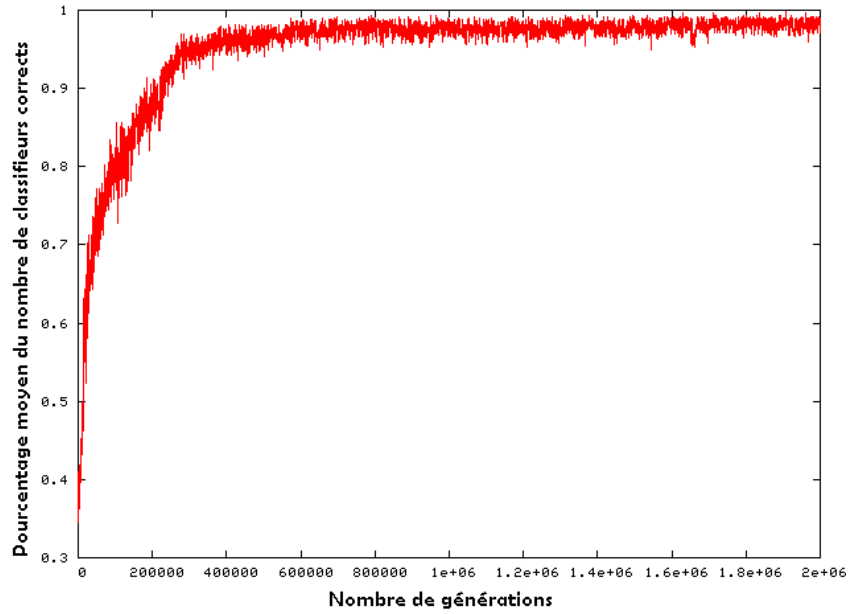


FIG. 7.17 – Pourcentage de classifieurs qui sont corrects en fonction du nombre de générations.

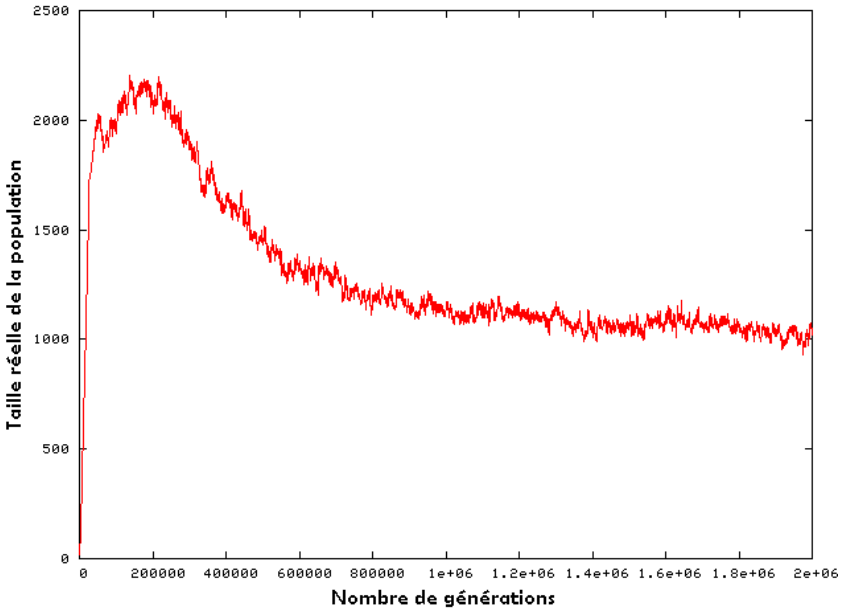


FIG. 7.18 – Taille réelle du jeu de règles $|T_{[P]}|$ en fonction du nombre de générations.

En fait, la taille du jeu de règles ($T_{[P]}$) n'est qu'une taille virtuelle. Les classifieurs sont dotés d'un numéro incrémenté à chaque fois qu'un de leur *clone* (*macro-classifieur*) est inséré dans la population. On peut se rappeler du paramètre de *numerosity* (ang.) à ce sujet (voir la section 5.2.2.2). En fait, la taille réelle ($|T_{[P]}|$) indique le nombre de classifieurs différents dans le jeu de règles, c'est-à-dire la complexité de la connaissance apprise. L'évolution de ce paramètre, montrée sur la figure 7.18, présente donc un intérêt. Nous y voyons qu'un nombre important de règles spécifiques sont créées au début de l'apprentissage, notamment par le *Covering Operator*. Chacune de ces règles ne s'active souvent que pour un ou deux exemples. Ensuite, des règles plus générales sont créées, et font converger la taille du jeu de règles vers une certaine proportion (ici la moitié) de la taille paramétrée par l'utilisateur.

Le dernier graphique de cette partie (figure 7.19) montre l'évolution de la *spécificité* des règles. Une règle spécifique est activée par un nombre d'exemples plus faible et participe à l'augmentation de la taille réelle du jeu de règles. Lors de l'utilisation de données de télédétection à valeurs continues, la *spécificité* est calculée en fonction de la surface de définition des conditions de la règle. En premier lieu, l'algorithme n'a aucune connaissance particulière et injecte des règles spécifiques dans la population, jusqu'à ce qu'un phénomène de sur-apprentissage survienne et soit apparemment éliminé. En réalité, **XCS-R** élimine un certain nombre de classifieurs spécifiques par le truchement de la pression génétique (notamment par l'opérateur de mutation qui produit de la généralisation et grâce au principe de partage de la récompense qui affaiblit les classifieurs spécifiques). Compte tenu des principes de fonctionnement de ce système de classifieurs, ce comportement est attendu et s'observe après quelques milliers de générations.

7.4.3 Tuning

Afin d'améliorer la qualité de la population apprise par **XCS-R**, différents paramètres ont été testés. Nous avons lancé de 100 à 1000 expérimentations en étudiant l'influence de la taille de la population, le nombre de générations et certains paramètres internes à l'algorithme comme le *cover-rand* et θ_{GA} (fréquence d'appel de l'AG sur la population [P]). La figure 7.20 présente la qualité observée de la base de classifieurs sur un jeu de test, en fonction de la taille réelle de la population ($T_{[P]}$), paramètre spécifié par l'utilisateur. Nous observons une augmentation rapide de la qualité jusqu'à un certain seuil, qui est dépendant à la fois des données (nombre d'échantillons et nombre d'attributs), et dans une moindre mesure

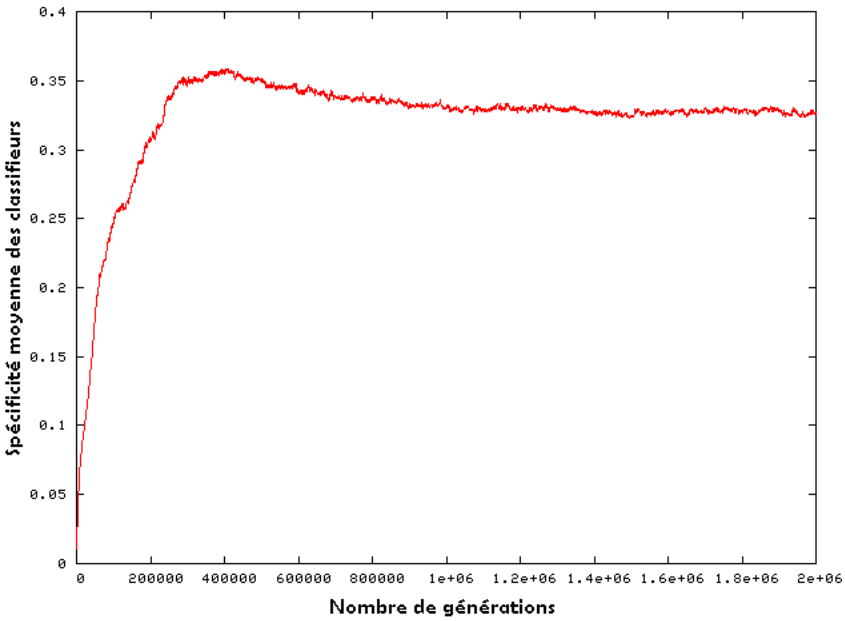


FIG. 7.19 – *Spécificité* moyenne des classifieurs en fonction du nombre de générations.

d'autres paramètres comme le nombre de générations. Comme nous le remarquons sur le graphique, ce seuil est d'environ 2000 règles. Parce que ces expérimentations sont très longues, nous n'avons pas réalisé de lissage, d'où l'aspect en pics de ce graphique[1]. Notons aussi que la qualité observée sur les matrices de confusion DCM (voir la section 4.6.2.1) est inférieure à celle obtenue sur les matrices classiques : un problème de classification de type *soft* étant toujours plus difficile qu'un problème de type *hard*.

Un autre test est présenté ici en utilisant les courbes ROC (ang., *Receiver Operating Characteristic*) introduite dans la section 4.6.2.4. Une question intéressante concerne l'étude de l'influence de la *spécificité* des classifieurs par rapport aux taux de faux positifs et faux négatifs. Par exemple, un classifieur trop générique couvrira un domaine de définition trop vaste et s'activera pour chaque exemple. À l'inverse, un classifieur trop spécifique ne s'activera jamais et sera inutile dans le jeu de règles. La *spécificité* d'un classifieur se mesure par la surface couverte par les intervalles des conditions, par rapport à une mesure dépendante de la base de classifieurs (par exemple, la moyenne des surfaces des règles de tout un jeu de règles) de telle sorte qu'elle soit indépendante de l'échelle des données. Les courbes ROC représentent un moyen efficace de tester l'influence de la *spécificité* des classifieurs en modifiant leur taille. Cette modification correspond au paramètre p de la courbe.

Après l'application de ce paramètre, pris dans l'intervalle $[0;1]$, la partie $<condition>$ $[m_i^R; M_i^R]$ d'une règle R pour l'attribut i est remplacée par :

$$[\mu - \varepsilon; \mu + \varepsilon] \quad \text{où} \quad \mu = \frac{m_i^R + M_i^R}{2} \quad \text{et} \quad \varepsilon = \mu \cdot p \tag{7.6}$$

Le jeu de données est ensuite classé avec la nouvelle population et les mesures de qualité $1 - Q_{spe}$ et Q_{sens} sont reportées sur la courbe. La figure 7.21 montre le résultat obtenu.

Nous y voyons que les classes pures « *sar2* » (*Sarcocornia Fruticosa*) et « *spa2* » (*Spartina Maritima*) sont moins sensibles à la *spécificité* des règles de **XCS-R**. À l'inverse, les classes mixtes nécessitent des classifieurs plus efficaces. On peut noter que les meilleurs classifieurs, correspondant à la distance AC_d la plus courte, sont obtenus pour $p = 1$, c'est-à-dire correspondant aux classifieurs originaux, sans modification. Le système de classifieurs apprend donc déjà les meilleurs règles possibles.

[1] D'autres expérimentations pourraient améliorer la qualité visuelle de ce graphique, mais nous avons décidé de conserver celui présenté ici car il met en évidence la variabilité des valeurs observées.

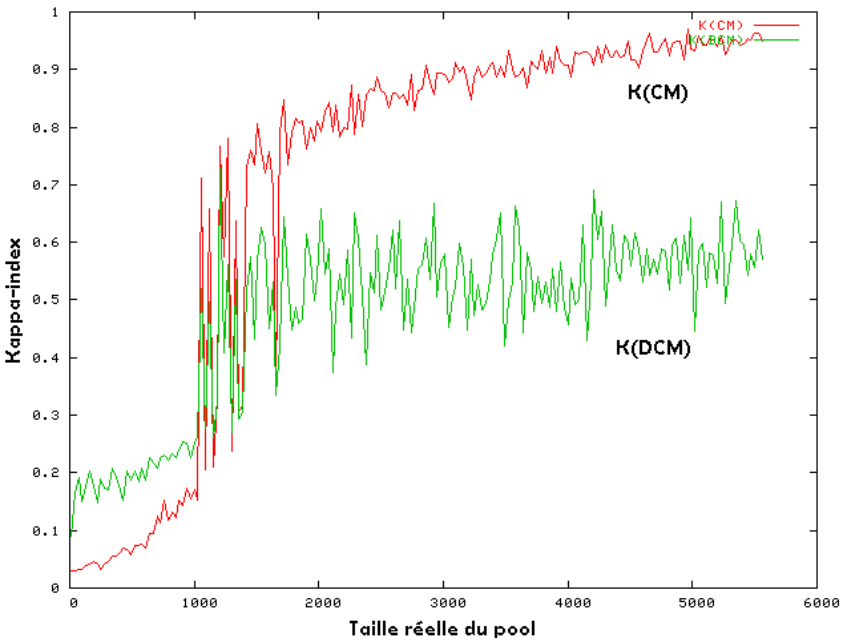


FIG. 7.20 – Qualité observée de la base de classifieurs, mesurée à l'aide du κ-index sur la matrice de confusion habituelle K(CM) et sur la matrice de confusion directe K(DCM), en fonction de la taille réelle de la population.

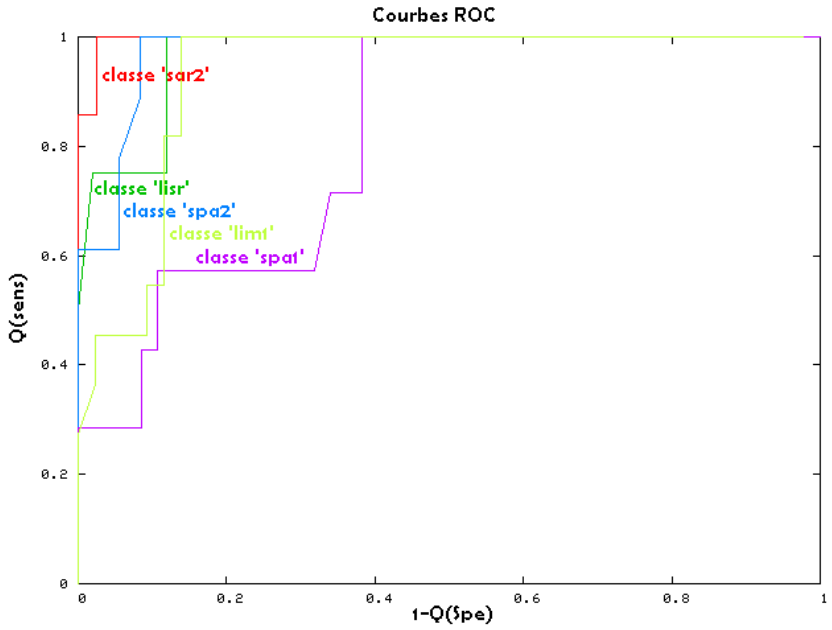


FIG. 7.21 – Courbes ROC pour chaque classe du jeu de données de CASI.

De toutes les études de tuning que nous avons menées, nous en avons conclu qu'un post-traitement de type génétique, comme présenté dans la section 7.3.2, reste plus efficace que la modification directe des paramètres internes de **XCS-R**. Cependant, cette solution peut s'avérer la plus rapide pour obtenir

un jeu de règles de qualité, d'autant plus qu'un post-traitement génétique nécessite une base de règles d'une taille suffisamment élevée.

7.5 Comparaisons avec d'autres systèmes

7.5.1 Comparaison avec une méthode inductive

Dans cette première partie, nous souhaitons comparer trois des méthodes développées dans cette thèse concernant la classification *hard* (**ICU**, **XCS-R** et **XCS5**) à **C4.5** [Quinlan, 1993], une méthode inductive dont nous nous sommes inspirés pour construire l'algorithme **XCS5**. L'algorithme **C4.5** est toujours considéré comme une référence pour le traitement des images satellitaires au moment où nous écrivons ces lignes [Debeir et al., 2001]. Plutôt que de discuter des résultats à l'aide d'images et de mesures de qualité, nous allons ici nous intéresser à la comparaison des représentations.

Nous rappelons brièvement les caractéristiques de ces algorithmes.

ICU est un algorithme évolutif produisant une règle de classification pour chaque classe à traiter. La partie <*condition*> des règles est formée de conjonctions de disjonctions d'intervalles, ce qui autorise l'algorithme à regrouper plusieurs groupes de spectres dans la même classe par une seule règle.

XCS-R est un système de classifieurs produisant une base de règles pour toutes les classes à traiter. La partie <*condition*> des règles est formée de conjonctions d'intervalles.

XCS5 est un algorithme de post-traitement d'une base générée par **XCS-R** et y extrait un arbre de décision classifiant toutes les classes à traiter.

C4.5 est un algorithme inductif créant un arbre de décision pour toutes les classes à traiter.

Pour la comparaison, les paramètres principaux de ces algorithmes ont été choisis afin d'obtenir la base de règles ou les arbres les plus légers possible, tout en ayant la meilleure qualité sur l'ensemble d'apprentissage. Nous avons travaillé sur l'image de QuickBird, comprenant 4 canaux spectraux, 324 exemples et 5 classes.

	C4.5	**ICU**	**XCS-R**	**XCS5**
Nombre de nœuds	37	–	–	35
Nombre de feuilles	19	–	–	5
Profondeur moyenne	4.6	–	–	6.5
Nombre de règles	–	5	2300	–
Qualité en apprentissage	0.90	0.77	0.81	1.00 ◇
Qualité en test	0.75	0.67	0.69	0.78 ◇
Classe « jun » (qualité)	0.76 ◇	0.70	0.61	0.69
Classe « lim »	0.83	0.63	0.93	1.00 ◇
Classe « sar »	0.56	0.65 ◇	0.35	0.40
Classe « spa »	0.75 ◇	0.65	0.45	0.51
Classe « wat »	0.77	0.73	1.00 ◇	1.00 ◇

TAB. 7.7 – Comparaison de certains paramètres associés aux résultats des méthodes **C4.5**, **ICU**, **XCS-R** et **XCS5**. Les meilleures qualités sont représentées par le symbole ◇.

Par leur nature, ces algorithmes produisent des objets de types différents. Une synthèse est proposée dans le tableau 7.7. Nous considérons que parmi les critères définis dans la section 1.2, la qualité de généralisation correspond à la mesure « qualité en test » définie dans le tableau, et que les critères de la simplicité de la représentation et celui de la compréhensibilité sont représentés par les mesures des tailles des arbres ou des bases de règles produites.

D'après le tableau 7.7, la représentation la plus simple est obtenue avec l'algorithme **ICU**, suivi des algorithmes **C4.5** et **XCS5**. **XCS5** produit des arbres légèrement plus petits que **C4.5**, bien que la

profondeur moyenne des arbres, correspondant à la durée de son évaluation, soit légèrement supérieure. Cependant, **XCS5** permet d'obtenir la meilleure qualité d'apprentissage et de généralisation par rapport à l'ensemble des algorithmes testés.

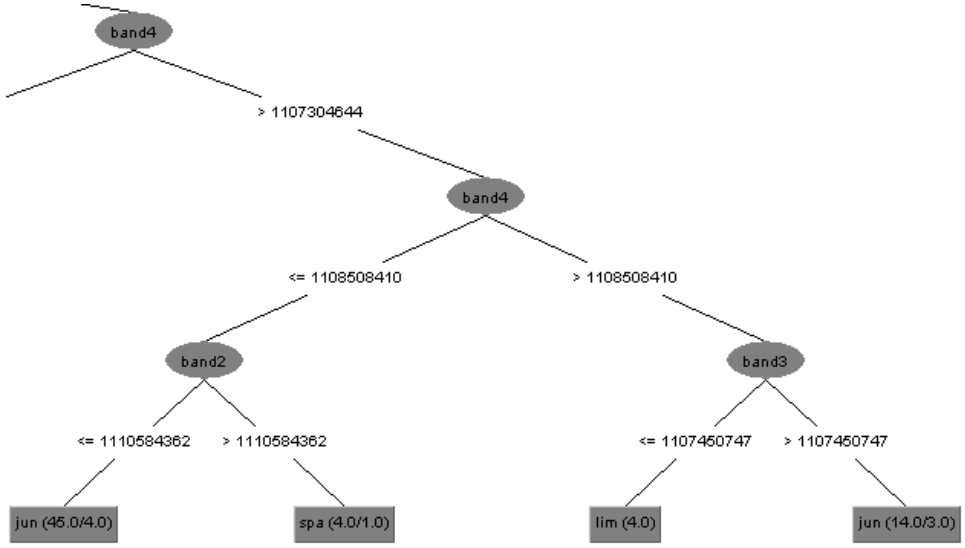


FIG. 7.22 – Extrait de l'arbre de décision obtenu avec **C4.5** pour l'image de Quick Bird.

Classe	Condition 1	Condition 2	Condition 3	Condition 4
water	[184254095; 1697585823]	[1111095941; 1112633176]	[1101700664; 1107911210]	[65713000; 1689640133]
spa	*aucune*	*aucune*	[812607812; 3995608073]	[1101328817; 1107529419]
sar	[1111197921; 1111900618]	*aucune*	[1103982725; 1106351939]	[1104847294; 1108970791]
jun	[1111831774; 1113828805]	*aucune*	[1105993125; 1110841190]	[1553060173; 4126604578] ∨ [1107331890; 1109909944]
lim	*aucune*	*aucune*	[1100513714; 1105572097]	[1107829371; 1116219594]

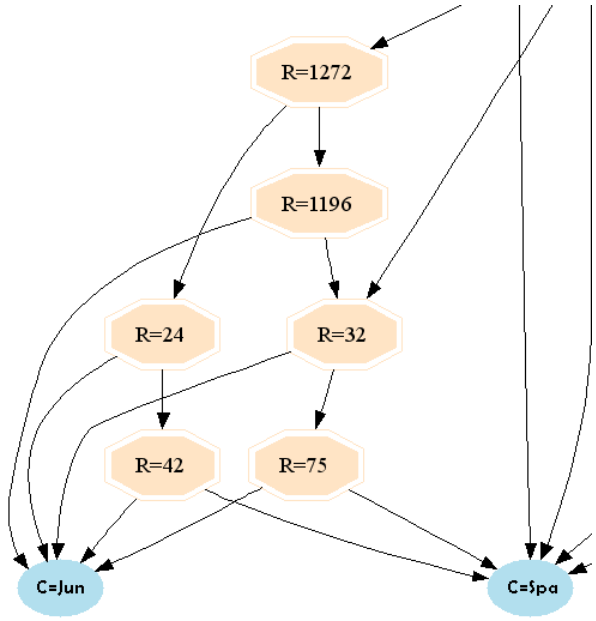
FIG. 7.23 – Base de règles obtenue avec **ICU** pour l'image de QuickBird.

Bien que les représentations obtenues soient assez simples, il nous a été difficile de présenter un arbre à 30 nœuds ici. Nous avons sélectionné, dans les figures suivantes, quelques extraits pour les algorithmes **C4.5** (figure 7.22), **ICU** (figure 7.23) et **XCS5** (figure 7.24). Dans les arbres, la branche de gauche fait référence au label « *oui* », tandis que celle de droite fait référence au label « *non* ».

À titre de comparaison, examinons par exemple les classes « *jun* » (*Juncus Maritimus*) et « *spa* » (*Spartina Maritima*). Pour **XCS5**, l'arbre de la figure 7.24 nous présente deux règles permettant de classifier des exemples de ces classes, selon des pré-conditions situées plus haut qui ne sont pas montrées dans l'extrait. Il s'agit des règles numéro 42 et 75, détaillées dans le tableau de la figure 7.24. Sur la figure 7.22, **C4.5** discrimine les deux classes en utilisant la bande numéro 2 (partie inférieure gauche de l'arbre). Quant à **ICU**, il discrimine les deux classes en utilisant essentiellement les bandes 1, 3 et 4 (voir la figure 7.23).

Mathématiquement, pour la classe « *jun* », une partie des conditions d'activation des règles sont les suivantes, selon les extraits des différentes règles :

$$\begin{array}{rll} f_{C4.5} := & \textbf{si} & ((b_4 \in \ ]1107304644; 1108508410]) \\ & & \wedge (b_2 \leq 1110584362)) \\ & \vee & ((b_4 > 1108508410) \\ & & \wedge (b_3 > 1107450747)) \\ & & \textbf{alors} \ \text{« } jun \text{ »} \end{array} \tag{7.7}$$



Règle	Condition 1	Condition 2	Condition 3	Condition 4
R=24	[1111474566; 1112976266]	[1109878107; 1111125153]	[1106714425; 1107233966]	[1106443267; 1107968839]
R=32	[1112282247; 1113473514]	[1110481959; 1112065376]	[1106895645; 1109565375]	[1107556909; 1108920696]
R=42	[1111628106; 1113521542]	[1109963290; 1111335420]	[1106428437; 1107909976]	[1107821093; 1109097650]
R=75	[1110643798; 1114002012]	[1108118897; 1111982040]	[1102530129; 1109565375]	[1108803992; 1111062741]
R=1196	[1112589385; 1113670011]	[1110625446; 1112526849]	[1108626370; 1109026719]	[1106776090; 1111062741]
R=1272	[1111499901; 1113670011]	[1109669653; 1110073050]	[1106719643; 1108320872]	[1106982143; 1108222551]

FIG. 7.24 – Figure du haut : extrait de l'arbre de décision obtenu avec **XCS5** pour l'image de QuickBird. Tableau du bas : extrait de la base de règles de **XCS-R** correspondant à l'arbre de décision.

$$\begin{aligned} f_{ICU} := \quad & \textbf{si} \quad b_1 \in [1111831774; 1113828805] \\ & \wedge \quad b_3 \in [1105993125; 1110841190] \\ & \wedge \quad ((b_4 \in [1553060173; 4126604578]) \\ & \quad \vee (b_4 \in [1107331890; 1109909944])) \\ & \quad \textbf{alors} \ \text{« } jun \text{ »} \end{aligned} \tag{7.8}$$

$$\begin{aligned} f_{XCS5} := \quad & \textbf{si} \quad b_1 \in [1111628106; 1113521542] \\ & \wedge \quad b_2 \in [1109963290; 1111335420] \\ & \wedge \quad b_3 \in [1106428437; 1107909976] \\ & \wedge \quad b_4 \in [1107821093; 1109097650] \\ & \quad \textbf{alors} \ \text{« } jun \text{ »} \end{aligned} \tag{7.9}$$

où b_i représente la valeur de radiance contenue dans l'échantillon pour la bande spectrale i.

Même s'il est vrai que ces équations ne donnent qu'une vue partielle des règles réellement trouvées par les algorithmes, il y a un certain nombre de caractéristiques communes concernant la connaissance apprise, qui ont été dégagées par ces extraits :

- la valeur b_2 est écartée en tant qu'attribut inutile ou du moins non discriminant par les trois équations (l'attribut n'apparaît pas pour **ICU**, et une plage très large lui est réservée dans les autres équations),
- la valeur b_3 est ciblée très précisément par les trois équations et doit se trouver dans la plage [1107450747; 1107909976],

– la valeur b_4 est, elle aussi, précisée de façon étroite par les trois équations qui la situent aux environs de la plage [1107821093; 1109097650].

Lorsque l'on regarde les différentes qualités obtenues sur les jeux de test pour les différentes classes, détaillées dans le tableau 7.7, on observe qu'aucun des algorithmes n'est performant pour toutes les classes à la fois. En fait, chacun des quatre algorithmes est *spécialisé* pour une ou deux classes différentes. L'algorithme **C4.5** traite au mieux les deux classes « jun » et « spa », tandis que **XCS5** classifie correctement tous les exemples de « lim » (*Limonium Narbonense*) et « wat » (*Eau*). Les répartitions statistiques des valeurs des exemples attribués à chaque classe sont très différentes, donc chaque classe tire profit d'une représentation particulière.

Bien entendu, cette étude ne tient pas compte du reste de l'arbre, elle montre simplement qu'il est possible d'établir une interprétation pour une partie des règles par rapport aux spectres réels des échantillons. Nous avons vu, qu'au moins pour cet exemple, ces interprétations sont cohérentes entre elles, ce qui paraît logique en regard des qualités élevées présentées par les différentes méthodes. On notera cependant des divergences pour chacune, par exemple pour **XCS-R** qui a tendance à présenter des contraintes précises, alors qu'elles sont plus relâchées pour **ICU** (la condition *joker* s'est exprimée plusieurs fois puisque des canaux ont été annulés et les intervalles sont plus larges en général), sans doute parce que le nombre de règles et donc de conjonctions avec lesquelles **ICU** peut exprimer la connaissance découverte est plus limité.

7.5.2 Comparaison en classification floue

Dans cette seconde partie, nous nous proposons de comparer l'approche floue de **ICUX** par rapport à des modèles connus pour donner de très bons résultats dans le domaine de la télédétection. Dans la littérature, la découverte de la composition spectrale des *mixels* est généralement traitée par deux méthodes : l'analyse SMA (ang., *Spectral Mixture Analysis*) et la classification floue [Lucieer et Kraak, 2004; Kriebel et Koepke, 1987].

L'analyse SMA consiste à exprimer la fonction de distribution de réflectance bidirectionnelle[2] d'une surface non homogène comme la somme pondérée des fonctions de distribution des constituants homogènes de cette surface [Kriebel et Koepke, 1987; Meerkoetter, 1990; Qin et al., 1996]. Malgré sa simplicité, de nombreux problèmes ont été rencontrés avec cette approche, notamment le fait que les modèles ne sont pas suffisamment robustes pour séparer les fonctions de distribution avec une précision acceptable. De plus, elle nécessite une intervention humaine pour le choix des pixels purs, ce qui ne la prive pas de toute erreur [Barnsley et al., 1998].

La seconde méthode est surtout dominée par les modèles probabilistes comme *Maximum Likelihood* [Harris et Stocker, 1998] ou les réseaux de neurones, et les modèles géométriques, tenant compte de la forme des arbres, de la direction et de la distribution de l'illumination du soleil [Ichku et Karnieli, 1996]. Ces dernières sont beaucoup plus complexes et très récentes [Lucieer et Kraak, 2004; Ichku et Karnieli, 1996]. La méthode *Maximum Likelihood* ne donnant pas de bons résultats avec nos données, nous avons choisi de comparer notre approche à des réseaux de neurones. Nous avons aussi complété cette étude par une comparaison avec SVM-R, la version *régression* des machines à vecteur support (ang., *Support Vector Machines - Regression*). Réseaux de neurones et machines à vecteurs supports sont connus pour être très robustes dans ce domaine [Benediktsson et al., 1990; Gualtieri et Cromp, 1998].

Pour avoir une idée sur les performances de l'algorithme **ICUX**, nous l'avons testé sur trois jeux de données. Les deux premiers jeux sont deux images de San Felice à des résolutions différentes : le capteur CASI (image de 754x293 pixels, résolution de 1.3 m^2 et 6 classes) et le capteur MIVIS (image de 396x170 pixels, résolution de 2.6 m^2 et 7 classes). Dans ces jeux, la composition de chaque classe experte est représentée en pourcentage avec une précision de 5%. Le troisième jeu est une base de 846 exemples de véhicules décrits par 18 attributs spatiaux continus (surface, périmètre, compacité, élongation, ...) qui est disponible sur le site de l'UCI KDD [Blake et Merz, 1998]. Ce dernier jeu a été choisi pour

[2]Notée BRDF (ang., *Bidirectional Reflectance Distribution Function*), il s'agit de considérer la fonction de réflectance d'un objet comme un modèle d'illumination géométrique [Barnsley et al., 1998; Nielsen, 2001].

ses caractéristiques intéressantes, c'est-à-dire le nombre élevé d'exemples et d'attributs, et la présence d'attributs continus.

Paramètre	**Valeur**
Taille du jeu de règles	300 individus
Initialisation	$P_e = 20$ parties égales
Critère d'arrêt	2000 à 5000 générations
P_{cross}	0.70
P_{mut}	0.15
$P_{mut,cond}$	0.30
$P_{mut,int}$	0.20
$P_{mut,term}$	0.40
ϵ	0.50
T_{up}	0.10
Opérateur de sélection	Ranking direct
Opérateur de remplacement	Ranking direct
Nombre d'enfants par génération	|P|
Elitisme	1% (fort)
Durée du traitement	20 à 40 min (CPU de 2.5 GHz)

TAB. 7.8 – Paramètres utilisés pour **ICUX**, voir la section 5.3.1.

Le tableau 7.8 présente les paramètres utilisés lors de l'apprentissage. Certains paramètres, relativement nouveaux par rapport à ceux des algorithmes évolutionnaires classiques, ont été choisis dans les plages de valeurs qui ont données les meilleurs résultats sur divers jeux de données. Les résultats sur les ensembles de test (représentant 50% de toutes les données) sont présentés dans les tableaux 7.9, 7.10 et 7.11, en utilisant les mesures de qualité développées dans la section 4.6.1.1.

	Q_{accur}	Q_{ppa}	Q_{spe}
Neural network	0.997 $\diamond$	0.981 $\diamond$	0.931
SVM-R	0.974	0.737	0.499
ICUX	0.972	0.974	0.933 $\diamond$

TAB. 7.9 – Résultats obtenus sur l'image de CASI. Les meilleures notes sont représentées par le symbole $\diamond$.

	Q_{accur}	Q_{ppa}	Q_{spe}
Neural network	0.776	0.645	0.597
SVM-R	0.978 $\diamond$	0.782	0.777 $\diamond$
ICUX	0.863	0.855 $\diamond$	0.765

TAB. 7.10 – Résultats obtenus sur l'image de MIVIS.

Pour l'algorithme à base de réseau de neurones, nous avons choisi un taux d'apprentissage de 0.1, 100000 itérations, 1 couche cachée de 7 à 15 neurones, une méthode d'apprentissage incrémentale et une fonction d'activation sigmoïdale symétrique. La couche de sortie représente les valeurs continues attendues par l'expert et il y a autant de neurones de sortie que de classes à apprendre. Nous avons utilisé la librairie gratuite en C qui se nomme *Fast Artificial Neural Network Library* [FANN, 2005]. La topologie choisie est simple mais efficace (voir le tableau 7.9 qui présente des qualités très élevées).

Pour l'algorithme à base de machines à vecteur support (*SVM-R*), nous avons utilisé le noyau *Radial Basis Function* dont les paramètres C et γ ont été découverts pour chaque jeu de données à l'aide d'une

	Q_{accur}	Q_{ppa}	Q_{spe}
Neural network	0.722	0.682	0.704
SVM-R	0.898 ◇	0.869 ◇	0.869 ◇
ICUX	0.619	0.570	0.609

TAB. 7.11 – Résultats obtenus sur la base de données de véhicules.

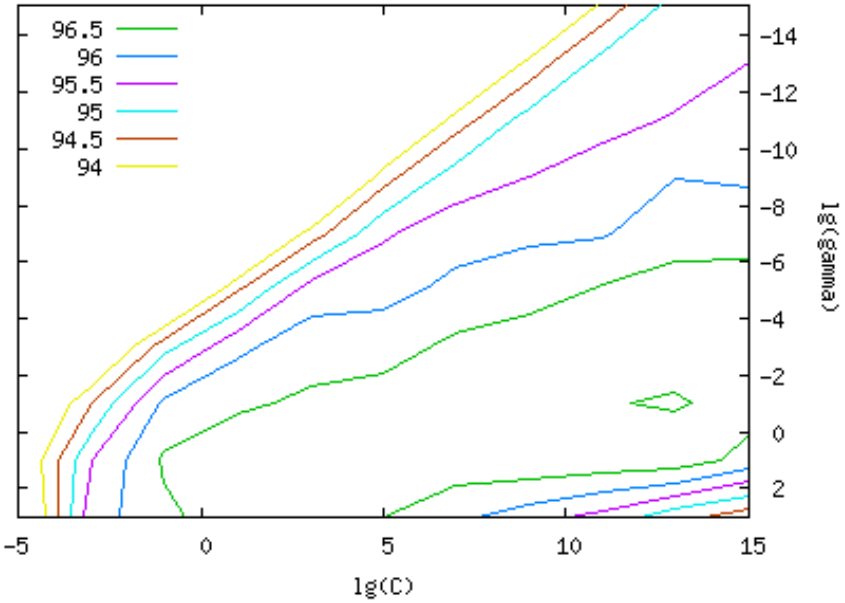


FIG. 7.25 – Optimisation des paramètres du noyau de SVM-R pour le jeu de données de CASI.

librairie gratuite en Python et d'un algorithme d'optimisation pas à pas présenté sur le site de la LIBSVM [Chang et Lin, 2001]. La figure 7.25 présente la stratégie d'optimisation utilisée par le script sur le jeu d'apprentissage : les paramètres optimaux se trouvent dans le polygone central.

L'image du capteur MIVIS était la plus difficile à traiter pour différentes raisons : quelques bandes supplémentaires, moins de pixels dans l'image donc moins d'exemples à classifier (à chaque fois l'ensemble d'apprentissage était constitué de moins de 2% de la totalité de l'image) et enfin les premières bandes de MIVIS sont légèrement plus bruitées que pour CASI. On note aussi une performance plus faible sur la base des véhicules car les valeurs des données se présentent sous une forme groupée, sans relation particulière entre les groupes de valeurs (cela est notamment dû à l'orientation des différents objets de la base). L'algorithme **ICUX** a une qualité moyenne de $Q_\mu = 0.818$ sur les trois jeux, ce qui est comparable à celle des réseaux de neurones, égale à 0.832. Pour l'image de CASI, la qualité de l'algorithme évolutionnaire est comparable à celle de SVM-R et s'en éloigne d'un peu plus de 10% pour l'image de MIVIS. Les performances sont donc très bonnes, surtout que les paramètres de SVM-R ont été optimisés pour chaque problème. Lorsque les valeurs sont regroupées, un système à base de disjonctions, comme **ICUX**, se révèle être intéressant. Les résultats de **ICUX**, même s'ils sont légèrement moins bons que ceux des autres classifieurs[3] (en fait, les notes Q_{accur} sont seulement 1.5% inférieures aux autres notes sur les images de télédétection), présentent l'intérêt d'être directement interprétables et d'exposer la connaissance apportée par les règles à un expert humain, ce que ne peuvent pas faire les réseaux de neurones ou l'algorithme SVM-R. Parce qu'il illustre bien notre propos, nous présentons le jeu de règles découvertes pour la classe « *bus* » de la base de véhicules dans le tableau 7.12. Le nombre de conditions par règle passait de 1 à 4

[3]Issus, eux aussi, de plusieurs années de développement. La version 1.0 de FANN est disponible depuis novembre 2003 et celle de LIBSVM date d'avril 2000.

intervalles selon la complexité des données, ce qui est très faible, et les 18 attributs sont décrits par des contraintes simples et lisibles.

Attribut	Nombre de conditions	Conditions
1	1	$[81.0982; 82.5794]$
2	2	$[40.5105; 53.1107] \vee [36.5423; 50.0423]$
3	1	$[68.0462; 70.7166]$
4	1	$[252.035; 263.525]$
⋮	⋮	...
18	1	$[180.865; 181.952]$

TAB. 7.12 – Exemple de règle produite par **ICUX** pour la classe « bus » de la base véhicule.

7.6 Élection au niveau des résultats

Nous sommes arrivés à un point où nous avions de nombreuses méthodes, associées à un nombre tout aussi important de paramètres et donc inévitablement de nombreux résultats. Nous cherchions alors un moyen de :

- pouvoir obtenir un résultat unique, fondé si possible sur une implication homogène de chaque méthode dans le résultat final,
- sans masquer les défaillances individuelles, c'est-à-dire, si possible, avoir un retour sur ces méthodes afin de proposer éventuellement des corrections.

Effectuer une simple moyenne des résultats est donc proscrite. De plus, des approches hybrides existent [Gançarski et Wemmert, 2005], mais nous souhaitions pouvoir appliquer les méthodes existantes, et même éventuellement des classifications déjà prêtes, détachées des méthodes, supervisées ou non, sans avoir à retoucher les algorithmes pour les emboîter entre eux (*hybridation*). Pour résoudre le problème des pixels qui sont classifiés différemment par les multiples méthodes de classification existantes, nous avons développé une solution à base d'élection consensuelle. À titre de synthèse pour ce chapitre, nous la présentons ici avec quelques résultats qui nous semblent intéressants. Dans la partie suivante, nous commençons par décrire de manière abstraite la notion de *mesure consensuelle*.

7.6.1 Mesure consensuelle

Une décision de classification d pour un échantillon P est la dénomination de la classe de cet échantillon par une méthode donnée. D'un point de vue statistique, d peut être modélisée comme la probabilité p_c qu'a une classe donnée c d'être affectée à P.

Quand l'étude concerne n classes, d est alors modélisée comme la distribution de probabilité comprenant l'ensemble des probabilités $(p_0, \ldots, p_i, \ldots, p_n)$ pour toutes les n classes. Pour les méthodes correspondant aux problèmes de type *hard*, où chaque échantillon est classé uniquement dans une classe, la valeur d'un élément p_i est 1 si l'échantillon est affecté à la classe i et 0 sinon. Pour les méthodes correspondant aux problèmes de type *soft* ou flous, la probabilité p_i dépend de l'estimation de la proportion de la classe i pour cet échantillon.

Lorsque différentes méthodes $(M_1, \ldots, M_n)$ s'affrontent sur le même échantillon, la distribution de probabilité peut être vue comme un modèle probabiliste indiquant la probabilité qu'a une classe donnée d'être sélectionnée pour un échantillon donné. Dans ce cas, chaque élément p_i est une moyenne des probabilités p_i^M assignées par chaque méthode dans leurs distributions respectives.

Un échantillon est dit avoir une bonne qualité de classification quand les décisions concernant sa classe sont consensuelles, c'est-à-dire qu'une forte homogénéité est observée dans les distributions de probabilité.

Nous allons maintenant introduire les concepts théoriques nécessaires pour mesurer l'homogénéité d'une telle distribution. Cette homogénéité repose sur le concept d'entropie.

Si $P = (p_1, p_2, \ldots, p_n)$ est une distribution de probabilité, alors l'information exprimée par cette distribution, appelée l'*entropie* de P, est définie par l'équation :

$$I(P) = -\sum_{i=0}^{n} p_i \log(p_i) \tag{7.10}$$

Moins la distribution est uniforme, moins elle délivre d'information. Si nous considérons que $P = (p_1, p_2, \ldots, p_n)$ est la probabilité de distribution de la classification d'un échantillon dans n classes par tous les classifieurs, alors la mesure de l'entropie peut être utilisée comme une mesure de l'homogénéité de la décision. Trivialement, d'après l'équation :

$$\lim_{p \to 0} p \log(p) = 0 \tag{7.11}$$

$I(1, 0, \ldots, 0) = 0$ et l'entropie est comprise entre 0 et $\log(n)$, où n est le nombre de classes. Nous définissions l'entropie normalisée d'une distribution, aussi appelée son homogénéité, par l'équation :

$$H(P) = \frac{I(P)}{\log(n)} \tag{7.12}$$

Dans ce cas, la consensualité peut être définie par l'équation :

$$C(P) = 1 - H(P) \tag{7.13}$$

La consensualité est maximale quand tous les classifieurs classifient un échantillon donné dans la même classe, et minimale quand tous les classifieurs le classifie dans des classes différentes. Cette mesure est illustrée sur la figure 7.26.

7.6.2 Cartes de consensualité

Dans cette étude nous avons utilisé la mesure $C(P)$ pour produire deux types de cartes distinctes :

La carte de consensualité est une carte dans laquelle chaque pixel P représente la consensualité $C(P)$, calculée à partir des décisions de classification de chaque classifieur. Les valeurs sont affichées en niveaux de gris. Par exemple, si tous les classifieurs sont d'accord pour classifier un échantillon dans la même classe, le pixel correspondant de la carte sera affiché en blanc ($C(P) = 1$). Si la classification de cet échantillon est homogène (par exemple, deux classifieurs indiquent la classe 'A', deux autres la classe 'B' et les deux derniers la classe 'C'), le pixel correspondant aura une consensualité faible et sera imprimé en noir ($C(P) = 0$).

La carte du vote est une classification dans laquelle chaque pixel est affecté à la classe dominante, d'après tous les classifieurs. Chaque pixel est imprimé dans la couleur correspondant à la classe en fonction d'une légende proposée par l'utilisateur. Si la consensualité d'un échantillon est inférieure à un certain seuil C_{min}, alors le pixel est imprimé en noir, signifiant « non classifié ». Un seuil de 0.5 a été choisi arbitrairement pour les figures qui vont suivre.

Une autre statistique a été proposée à partir de la carte de consensualité. Il s'agit d'un histogramme qui indique pour chaque catégorie de valeurs dans la carte, le pourcentage correspondant de pixels. Les valeurs en abscisse de cet histogramme sont généralement prises par pas de 0.1 points, c'est-à-dire $[0; 0.1[; [0.1; 0.2[; \ldots; [0.9; 1.0]$, mais d'autres échantillonnages peuvent être employés. Cet histogramme peut être utilisé comme post-validation de la carte de consensualité et constitue une preuve importante de sa qualité. Le nombre de pixels le plus élevé doit se situer dans le dernier intervalle. L'analyse de cet histogramme peut permettre de faire des conclusions intéressantes. Par exemple, si l'intervalle $[0.9; 1.0]$ est maximal et qu'un maximal local est observé dans l'intervalle $[0.5; 0.6[$, nous pouvons conclure qu'il

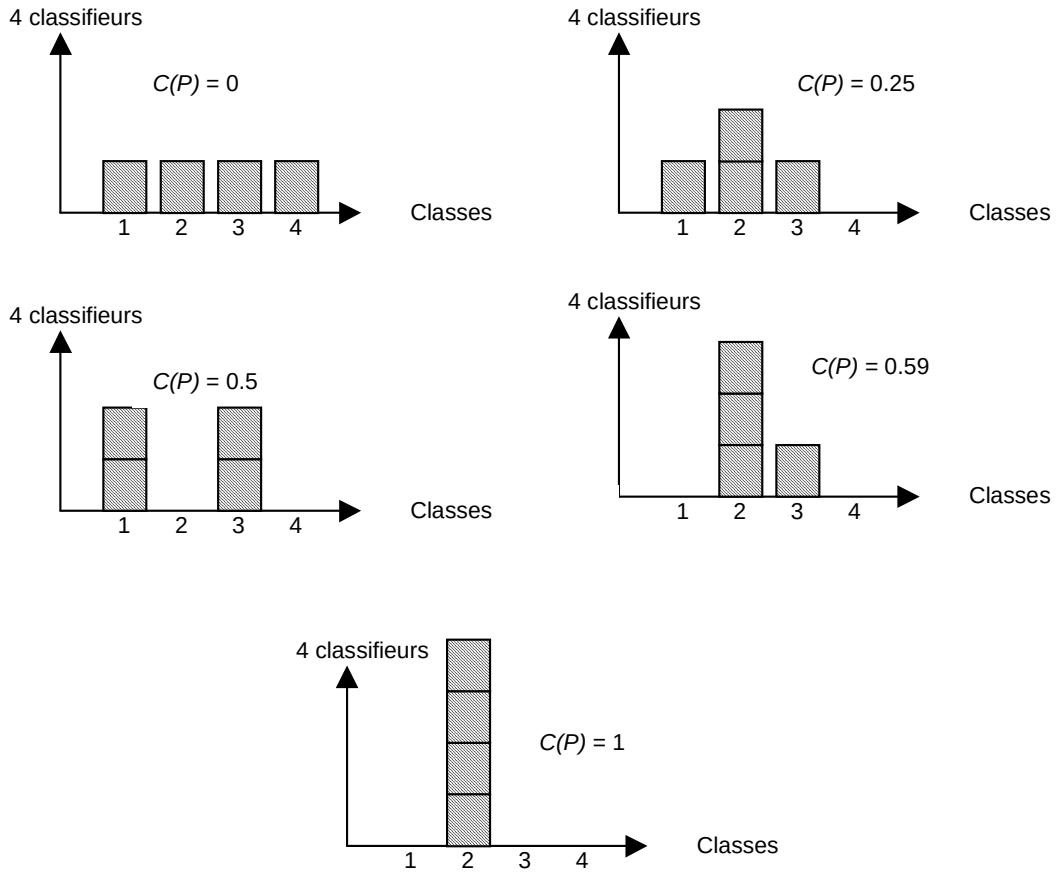


FIG. 7.26 – Comparaison de différentes valeurs de consensualité.

y a principalement deux types d'échantillons : ceux qui sont consensuels, et ceux qui ne le sont systématiquement pas à raison d'un choix ambigu entre deux classes. L'identification de ces classes peuvent conduire à l'exclusion de certains classifieurs ou au rajout de suffisamment d'informations expertes pour améliorer la qualité de l'histogramme. L'analyse des résultats à la lumière d'un seul classifieur, dans une démarche non collaborative, ne permettrait pas de telles identifications.

7.6.3 Illustration

En plus de la carte consensuelle, nous présentons ici un comparatif des différents résultats que nous avons pu obtenir, issus d'un travail collaboratif induit par le projet TIDE. Le tableau 7.13 indique la liste des classifieurs sélectionnés pour cette étude.

Les résultats des classifieurs sont présentés de manière indépendante sur la figure 7.27. Nous pouvons y voir que de nombreuses zones ne sont pas toutes classifiées à l'identique, notamment pour les zones de végétation situées au nord-est. La figure suivante (figure 7.28) montre la carte du vote : de nombreux pixels ont pu être votés, le nombre de pixels noirs étant très faible. Ces pixels non consensuels se situent surtout en bord de mer, en effet, la végétation y est extrêmement mélangée. La figure 7.29 montre la carte de consensualité. Les pixels grisés sont les plus difficiles à classifier. On remarque que la région du nord-est, surtout constituée de plantes fines, couvrant un sol très humide voire inondé, est la moins

Nom	Supervisée ?	Floue ?	Paramètres	Forme des résultats	Durée CPU (3 GHz)
C4.5	Oui	Non	Confiance (0.25), feuilles (3)	Arbre de décision	Courte (5 min)
ICU	Oui	Non	Population (50 à 200), générations (100 à 500)	Règles de classification	Moyenne (30 min)
ICUX	Oui	Oui	Population (100 à 300), générations (200 à 1000)	Règles de classification	Longue (1h)
XCS-R	Oui	Non	Population (2000 à 3000), générations (200000)	Règles de classification	Longue (1h)
MLP[a]	Oui	Oui	Topologie (5 à 15 neurones cachés), itérations	Réseau de neurones	Moyenne (30 min)
SAM[b]	Oui	Non	Seuil (0.1 *rad*)	Angles	Courte (2 min)
K-Means	Non		Noyaux (15 à 20)	Classif. réaffectée	Courte (10 min)
SOM[c]	Non		Taille de la fenêtre (3x3 à 5x5)	Classif. réaffectée	Courte (10 min)

[a] *Multi-Layer Perceptron*
[b] *Spectral Angle Mapper* [Yuhas et al., 1992]
[c] *Self-Organizing Map* [Kohonen, 1982]

TAB. 7.13 – Classifieurs utilisés pour l'élection d'un résultat consensuel.

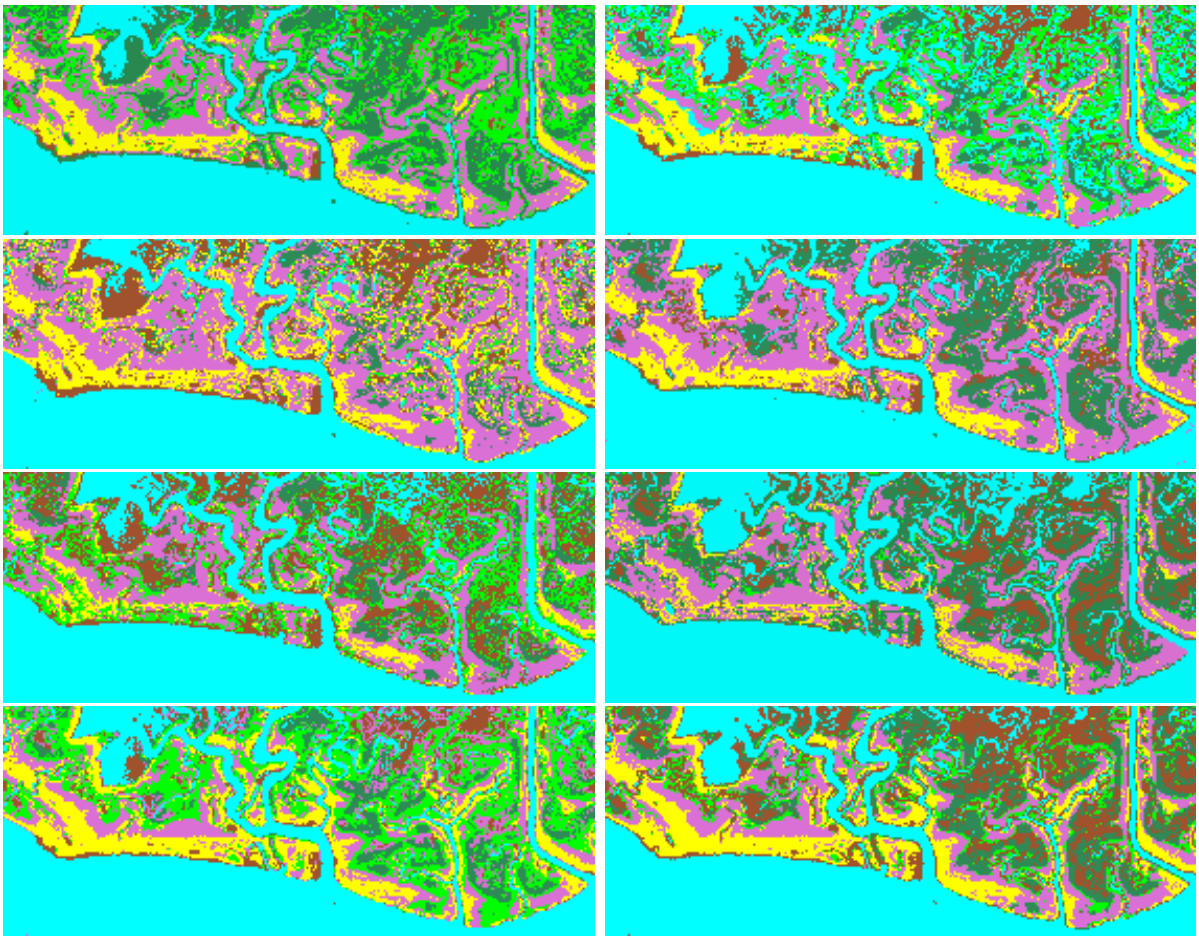


FIG. 7.27 – Classifications de l'image de CASI par différents classifieurs. De gauche à droite, puis de haut en bas : (a) C4.5, (b) ICU, (c) ICUX, (d) XCS-R, (e) MLP, (f) SAM, (g) K-Means (15 classes), (h) SOM (fenêtre 4x4).

consensuelle. Heureusement, le sol et l'eau ne représentent pas des classes intéressantes pour le projet TIDE. L'histogramme de la figure 7.30 nous apprend qu'une grande majorité de pixels (plus de 40%) sont pleinement consensuels et les dix classifieurs ont pu fournir une expertise efficace sur plus de 70%

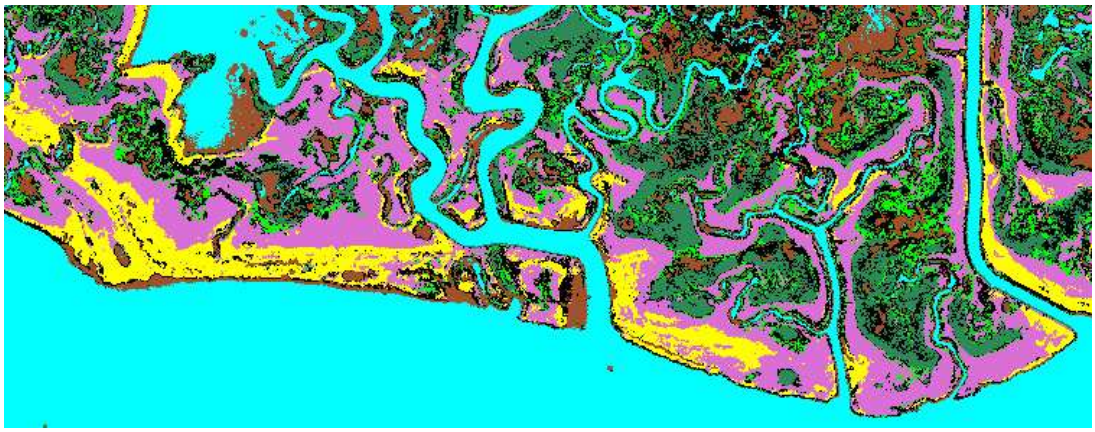


FIG. 7.28 – Carte de vote pour CASI.

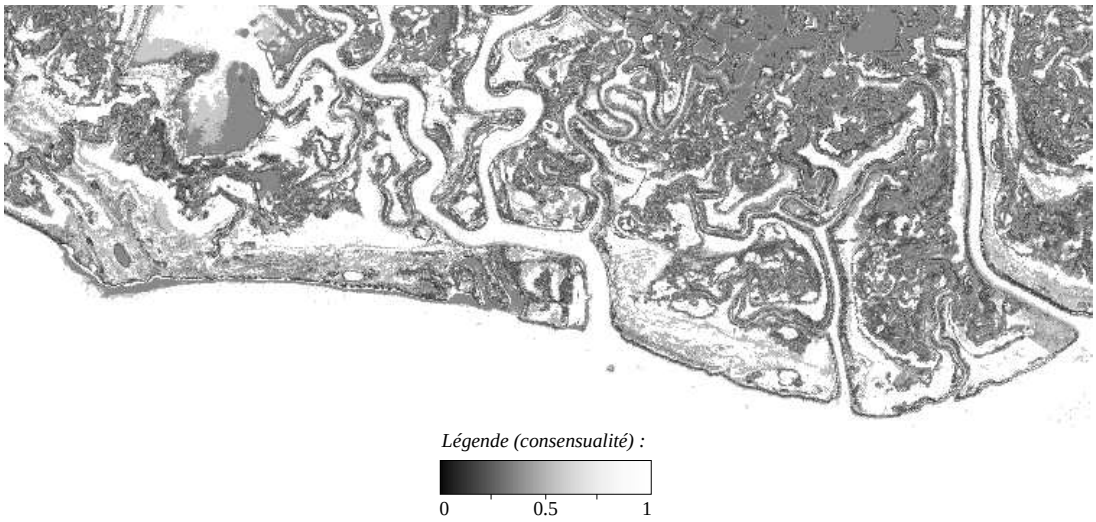


FIG. 7.29 – Carte de consensualité pour CASI.

de l'image ($C(P) > 0.5$). Il faut cependant noter que la majorité des pixels ayant une consensualité maximale correspondent à l'eau, plus facile à discriminer que la végétation.

Ces résultats sont très encourageants et surtout surprenants car ils ont été générés par des méthodes très diverses, avec des paramètres et des jeux d'apprentissages différents (par exemple, certaines méthodes nécessitaient seulement quatre pixels pour l'apprentissage, alors que pour d'autres, il en fallait quelques centaines). La technique que nous avons développée, basée sur l'entropie, permet donc d'unifier efficacement une dizaine de classifieurs différents, en présentant un résultat final plutôt homogène.

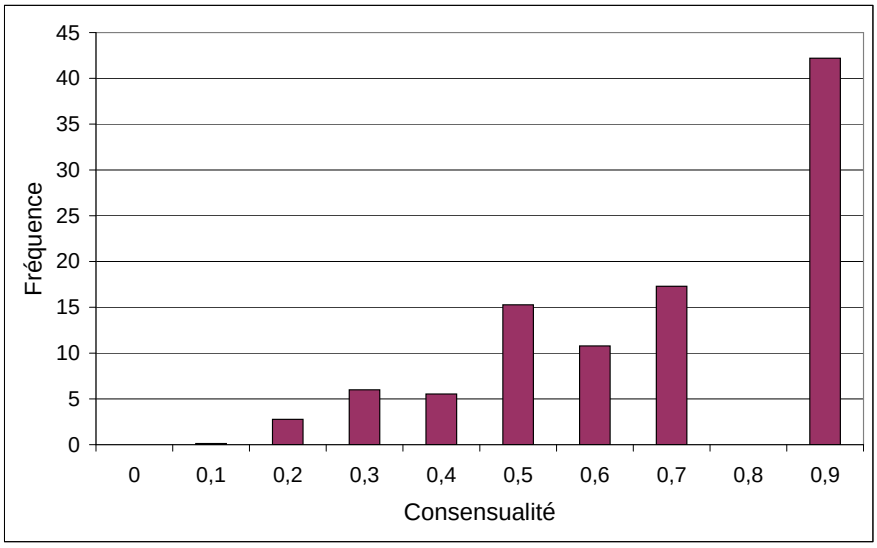


FIG. 7.30 – Histogramme de la consensualité pour CASI.

Conclusions et perspectives

Travaux et contributions en évolution artificielle

Nous avons mené à bien une étude sur la découverte de règles de classification des concepts thématiques contenus dans une image de télédétection. Cette étude a constitué d'une part en la définition de représentations adaptées à ce processus de découverte et d'autre part, en l'intégration de ces représentations au sein de plusieurs systèmes d'apprentissage, coiffés par une architecture évolutive générique. Les connaissances découvertes sont stockées dans des bases de règles indépendantes et réutilisables. Les objectifs qui ont été définis dans la problématique ont été atteints. Les algorithmes mis au point sont robustes et efficaces, comme nous l'avons montré dans les différentes études de cas. Nous avons montré que les représentations choisies, plates ou arborescentes, sont à la fois pratiques et simples pour l'utilisateur, et suffisamment expressives pour assurer une qualité de généralisation assez élevée. À cette intention, nous avons validé certaines étapes clés.

La première étape consistait en l'étude théorique de représentations de classifieurs spécialisés dans le traitement d'images multispectrales ou hyperspectrales, permettant d'intégrer toutes les informations nécessaires au traitement de données d'une complexité aussi importante. Pour obtenir des classifieurs adaptés à ce domaine d'application, il a fallu imaginer des structures capables d'apporter une solution efficace au problème posé, tout en gardant une certaine simplicité (ergonomie, lisibilité, présentation du résultat, etc.) vis-à-vis de l'utilisateur. Après une étude de l'existant qui a orienté la composition d'une liste de critères intéressants pour l'expert, nous avons considéré différentes représentations, utilisant des opérateurs booléens ou des opérateurs mathématiques qui ont respectivement donné accès à la résolution de problèmes de classification de type *soft* ou flous. Nous avons donc créé des règles symboliques à base de conjonctions de disjonctions d'intervalles, d'arbres de décision ou d'arbres de calcul, garantissant du même coup leur applicabilité à des problèmes de classification variés : *hard*, *soft*, flous ou même par intervalles flous. Puis nous avons montré que le comportement de base de ces règles était à même de saisir les particularités de l'environnement étudié, par exemple le cas des *interstices spectraux* observés à des degrés d'humidification divers de la végétation (section 5.3.1.5).

Ces classifieurs, nouveaux dans leur genre, nécessitent l'emploi d'un système permettant de les manipuler de manière adéquate. La deuxième étape nous amena donc à étendre la stratégie proposée par les systèmes de classification habituels pour réaliser deux objectifs. D'une part, il fallait intégrer, dans notre architecture, de nouveaux opérateurs génétiques spécialisés pour chacune des représentations. D'autre part, il fallait traiter les échantillons d'apprentissage avec une fonction d'évaluation offrant une pression de sélection efficace, combinant la qualité de décision des règles, la brièveté et la généricité de leur représentation avec les contraintes de l'utilisateur. C'est ainsi que nous avons étudié puis réalisé des opérateurs adaptés à l'initialisation de classifieurs évolutifs (procédures GenMinMax et GenSpectro de **ICU**, à base de *mixels* pour **ICUX**, à base de grammaires pour **GramGen**, etc.), à leur croisement et leur mutation, à la détection de convergence (par exemple, par stabilisation de la fonction *fitness*) ou à leur sélection. En conséquence, nos travaux ont permis l'élaboration d'un modèle possédant une grande adaptabilité et ce à plusieurs niveaux. Tout d'abord il permet d'aborder des images possédant un nombre quelconque de dimensions spatiales, spectrales et radiométriques, et de qualité inégale (bruit, expertise succincte, etc.). Ensuite, il autorise l'intégration future de règles plus puissantes de façon cohérente avec le reste du système. Enfin, l'application dans d'autres domaines, tout aussi hostiles aux classifications traditionnelles, est envisageable malgré la présence d'opérateurs spécialisés.

La troisième et dernière étape était de proposer un processus de validation de confiance, permettant aussi bien de sélectionner les règles lors de l'apprentissage, que de les évaluer à la sortie des algorithmes. Ce processus est concrétisé par différentes mesures de qualité (*précision*, *sensibilité*, *spécificité*, etc.) spécifiques aux données ou à la représentation (mesures de taille ou de diversité d'une population d'arbres), ainsi que par des protocoles de validation (*holding-out*, *cross-validation*, *bootstrapping* et *jackknifing*). Nous les avons complétés par des mesures de validation graphique (carte de recouvrement, spectrogramme, méthode *RuleView*, *cover-graph* et courbes ROC). Cette validation a donc pu se poursuivre aussi bien tout au long de l'apprentissage (évolution de la *spécificité*, *tuning*) qu'en post-validation, permettant d'apprécier la pertinence et la facilité d'interprétation des règles trouvées [Quirin et al., 2004].

Apports pratiques en classification d'images de télédétection

Les résultats présentés dans le dernier chapitre de ce mémoire sont très encourageants, par rapport aux difficultés posées par le contenu très riche de ces images et par la qualité variable de l'expertise. Cette dernière peut en effet souffrir, par son caractère humain ou systématique, de problèmes de bruit, de pixels mixtes ou de bordures mélangeant toutes les combinaisons de classes possibles. Cependant nous avons prouvé, par les expérimentations, que notre système se révélait être très robuste, par exemple, en retrouvant le contour des régions d'intérêt fixé par l'expert (section 7.3.1) et même en découvrant des indices génériques, comme *NDVI* ou I_{Lim}, ce qui apporte une valeur ajoutée à la classification.

L'étude de cas a permis de dégager plusieurs points intéressants. Les structures proposées par **ICU** et **XCS-R**, malgré leur simplicité, reproduisent fidèlement la réalité [Quirin et al., 2005]. Nous avons vu qu'un pool appris en situation de classification *soft* peut être converti et utilisé en situation de classification *hard* par l'addition, à la base de règle, d'une stratégie de sélection des règles en cas d'activation multiple (recouvrements) ou de classes non apprises, permettant ainsi l'attribution sans équivoque de chaque élément de l'image à la classe qui lui est la plus proche. Ainsi, les stratégies à base de cartes de recouvrement (pour **ICU**) et les méthodes *MaxConfident* et *ScoreConfident* (pour **XCS-R**) ont été créées et testées dans ce but [Quirin et Korczak, 2005a]. Nous avons aussi vu comment nous pouvions répondre aux paradigmes émergents, comme celui de l'*unmixing* (classification floue), en proposant une fonction de correspondance (ang., *matching*) règle-échantillon particulière (**ICUX**, [Quirin et Korczak, 2005b]) ou deux systèmes à base de programmation génétique (**ProgGen** et **GramGen**), contraints par probabilités ou grammaires pour ne pas faire exploser l'espace de recherche. Enfin, nous avons proposé plusieurs méthodes de raffinement de la base de règles, par post-traitement génétique (réduction, transformation ou recombinaison de la base), extraction de fonctions d'appartenance permettant de préciser la formation de *niches* dans **XCS-R** et extraction d'un arbre de décision (**XCS5**). Nous avons notamment observé, lors d'une étude comparative entre **XCS5** et **ICU**, que ces algorithmes délaissent les canaux inexploitables ou inutiles tout en présentant des résultats cohérents.

Finalement, ces travaux ont donné lieu à un prototype, nommé **VPlat** (voir l'annexe F), exploitant tous les algorithmes étudiés, dans le but de tester leur validité sur des exemples concrets, mais aussi de proposer un outil convivial, diffusant la connaissance acquise à des équipes non spécialisées dans l'informatique, qui seraient intéressées par nos résultats. Cet outil accomplit la chaîne complète du traitement des données, depuis la visualisation des images hyperspectrales jusqu'à la production de descriptions de classes compréhensibles en passant par l'obtention d'images classifiées. L'architecture modulaire basée sur le *paradigme objet* permet l'intégration rapide de nouvelles méthodes, actuelles ou à venir, simplement en ajoutant une classe de fonctions spécifiques au nouvel objet. Elle facilite aussi la compréhension du système, l'utilisateur y gagne donc lors de son paramétrage. Ce prototype a été présenté avec intérêt à l'équipe Télédétection, Radiométrie et Imagerie Optique du LSIIT et au laboratoire Image et Ville de Strasbourg (dans le cadre de l'ACI *Masse de données*), avec lesquels nous avons tissé des liens depuis longtemps. Les méthodes d'apprentissage évolutives **ICU**, **XCS-R**, **ICUX** et **GramGen**, ainsi que le système de sélection des résultats à base de carte de consensualité et de carte de vote sont utilisés actuellement pour le défrichage des données de l'année 2004, avant la conclusion du projet européen TIDE [TIDE, 2005], qui par son enjeu et la nature des sommes investies, requerrait un prototype fonctionnel et applicable rapidement.

Perspectives

Les algorithmes évolutifs, et donc les algorithmes évolutionnaires et les systèmes de classifieurs que

nous avons employés, sont connus pour être coûteux en temps et en mémoire car ils nécessitent des populations de grande taille. Une première perspective pour nos travaux pourrait être l'optimisation de leur temps de calcul. Par exemple, nous pouvons optimiser la découverte des arbres de calcul dans **ProgGen** et **GramGen** en optimisant les valeurs des constantes des arbres (ang., *tuning*) toutes les n générations, ou en utilisant les fonctions automatiquement définies (ADF). Outre l'optimisation des algorithmes eux-mêmes, la parallélisation est une piste qui peut se révéler fructueuse pour le gain de temps. Les algorithmes à base de populations de règles se prêtent naturellement à la parallélisation et celle-ci a été étudiée largement dans la littérature. Le transfert de la charge de travail sur un ou plusieurs postes serveurs représente un autre intérêt des systèmes distribués car ils peuvent dès lors être présentés sous forme de service à l'utilisateur (réseau local, réseau Internet).

Nous sommes aussi conscients que la représentation des règles évoquée ici n'exprime, par des expressions réduites à leurs formes normales, qu'une connaissance spectrale. Afin d'inclure d'autres types de connaissances existantes et pertinentes dans le domaine d'étude (relations spatiales ou temporelles) il faudrait étendre notre formalisme de représentation pour y inclure des concepts sémantiquement plus riches comme la forme des objets, leur texture, etc. et de dépasser le niveau du pixel en proposant des connaissances de niveau région. Mais bien au-delà, les classifications contextuelles permettraient la formation de concepts thématiques puissants, à l'image de prédicats du premier ordre (« tout échantillon de végétation est entouré d'échantillons de non-eau ») ou de prédicats flous (« un arbre isolé est souvent de forme ronde et de petite taille », « un pavillon est rarement au bord de l'autoroute », etc.). D'autre part, l'utilisation de grammaires temporelles permettant l'exploitation de séquences d'images augure, elle aussi, de perspectives intéressantes. Ce formalisme très ouvert exigera des opérateurs génétiques et une représentation appropriés, capables de regrouper des régions de pixels ou de découvrir des relations entre les séquences de valeurs, autorisant la mise à jour de connaissances de la base de concepts, qu'elle soit radiométrique, spatiale ou temporelle. L'inclusion au préalable de toutes ces méta-connaissances améliorera les performances actuelles de notre système et renforcera notre savoir et notre expérience en télédétection.

Notons que cette mise à jour ne pourra sans doute pas s'effectuer depuis une unique source de données. L'absence d'un traitement direct des données multi-sources et multi-échelle devra être comblée : de nombreux moyens sont d'ores et déjà disponibles et ce type de traitement présente un intérêt certain dans la communauté des géographes. Cette problématique, ainsi que celle de l'utilisation d'opérateurs thématiques plus poussés (morphologiques, contextuels, etc.), s'inscrira immanquablement dans la continuité de nos travaux, d'autant plus qu'elle fait partie du cadre du projet FoDoMust de l'ACI *Masse de données* qui se poursuit actuellement, et qui promet une collaboration étroite entre informaticiens, géographes et spécialistes du traitement d'image, caution de la mise en place harmonieuse de toutes ces briques.

Enfin, d'autres perspectives pourraient être envisagées, parmi lesquelles une interactivité plus poussée entre l'utilisateur et le système **VPlat** (visualisation de la population en cours d'apprentissage, marquage d'exemples ou de contre-exemples en dessinant directement sur l'image pendant une pause de l'algorithme, manipulation de *points d'arrêt* en posant des contraintes sur différents paramètres internes de l'algorithme, etc.) ou bien une hybridation des algorithmes d'apprentissages (un arbre obtenu par **GramGen** sélectionne les résultats de plusieurs méthodes évolutives ou non en fonction de la classe étudiée ou des données observées). Nous le voyons, de nombreux problèmes restent ouverts, tant dans l'amélioration des représentations que dans celle des algorithmes qui les manipulent.

Annexe A

Rappels sur les algorithmes connexionnistes

De part la simplicité de leur fonctionnement, leur disponibilité très tôt dans la communauté et leurs performances souvent correctes, les réseaux de neurones ont maintes fois été appliqués en télédétection. Un réseau neuronal est un ensemble de cellules de calcul appelées *neurones*, connectées par des liens nommés *connexions* portant chacun un *poids* et modifiant la valeur des données transitant entre les entrées du réseau et ses sorties [Rosenblatt, 1962]. Les attributs explicatifs (en entrée) sont souvent appelés variables exogènes, tandis que ceux à prédire sont dénommés variables endogènes (en sortie). Ces réseaux font partie de la grande famille des mémoires hétéro-associatives [Williams et Zipser, 1989], c'est-à-dire qu'ils peuvent apprendre à associer certains concepts à d'autres. Généralement en télédétection, on voudrait pouvoir associer une couleur, une forme, la densité d'une zone de points à une représentation conceptuelle de cette zone (végétation, ...). On donne donc à ces algorithmes un ensemble de paires de valeurs (entrée et sortie souhaitée) et l'apprentissage découvre une fonction déterministe qui fait correspondre les entrées sur les sorties, en minimisant les conflits qu'il peut y avoir sur de nouvelles présentations de paires. Voici un parcours de quelques types d'algorithmes connexionnistes existant.

A.1 Perceptrons MultiCouches

Les Perceptrons Multi-Couches (PMC) [Rumelhart et al., 1986] permettent de séparer les données par des hyperplans. Dans ce cas, le réseau détermine la classe d'un objet présenté en entrée en fonction de son appartenance à la région supérieure ou inférieure situées de part et d'autre de l'hyperplan. Plusieurs algorithmes d'apprentissage ont été développés calculant l'ajustement du poids des connexions. Ils utilisent dans la plupart des cas l'une des deux stratégies évoquées ci-dessous :

- La règle de Hebb [Hebb, 1949] basée sur l'interprétation suivante : si deux neurones de chaque côté d'une connexion sont activés simultanément, le poids de la connexion va augmenter. La règle est la suivante :

$$w_{ij}^{new} = w_{ij}^{old} + e_i s_j \tag{A.1}$$

  où w_{ij}^{new} est le nouveau poids de la connexion du neurone i au neurone j, w_{ij}^{old} est son ancien poids, e_i est la valeur de l'entrée du neurone et s_j la valeur de sa sortie.
- D'autres variantes existent comme par exemple la règle de Hebb avec taux d'apprentissage (la forme classique), avec terme d'oubli [Gluck et Thompson, 1987] ou la règle de Widrow-Hoff, [Widrow et Hoff, 1960]. Ces règles ajoutent des termes supplémentaires qui adaptent les poids des connexions de manière à obtenir la convergence la plus efficace possible.

L'algorithme le plus couramment utilisé qui manipule de telles règles est celui de la rétro-propagation du gradient (ang., *back-propagation*) [Rumelhart et al., 1986], décrit aussi par [Fodor et Pylyshyn, 1988].

À partir d'une base d'exemples (paires de valeurs pour les entrées et les sorties désirées), l'algorithme calcule l'erreur du réseau à partir de la corrélation entre les exemples donnés et les sorties attendues. Ensuite, les poids des connexions sont modifiés en fonction de la part qu'occupe chaque connexion dans l'erreur globale. Cet algorithme est généralement utilisé pour les réseaux de type multi-couches (ayant au moins une couche cachée) : dans ce cas, on propage en arrière (depuis les neurones de sortie vers les neurones d'entrées) l'erreur calculée sur les neurones de sortie. Les PMC ont été très tôt utilisés en classification d'images et donnent de bons résultats [Manry et al., 1994; Jiang et al., 1994]. On peut aussi citer les travaux de [Liu et al., 2001] concernant la récupération de canaux hydrauliques urbains sur une image DOQQ (ortho-photo) de 6200x7800 pixels.

A.2 Réseaux HyperConvexes

Les réseaux HyperConvexes, introduits par [Novak, 2000] définissent un nouveau type de neurone utilisant le principe des *polygones de Thiessen* [Blekas et al., 1997], pour délimiter l'espace en régions géométriques rectangulaires ou sphériques. Cette famille de réseaux a été améliorée, toujours en vue de les appliquer à la télédétection, par [Grzelak, 2001] en ajoutant la détection de zones ellipsoïdales (orientées ou non). Grâce à ces réseaux, on peut mieux délimiter la région de l'espace concernée par l'activation des neurones, ce qui en font des systèmes plus précis. Ces réseaux ont de bonnes capacités en télédétection et, de surcroît, l'apprentissage est rapide.

A.3 Réseaux RBF

Les réseaux RBF (*Radial Basis Function*) [Powell, 1985; Boser et al., 1992] sont des réseaux qui permettent de modifier les poids des connexions uniquement de manière locale. Ceci est intéressant dans le cadre de la reconnaissance de forme ou dans la télédétection où il faut inculquer au réseau que les pixels adjacents à un pixel donné jouent un rôle très important durant l'apprentissage, tandis que les pixels trop éloignés ne doivent pas intervenir. Une telle méthode d'apprentissage est développée dans [Blanzieri, 1998]. Elle est par exemple utilisée pour découvrir une douzaine de classes sur une images multispectrale [Luo et al., 2001] avec une meilleure précision comparée aux Perceptrons classiques.

A.4 L'algorithme Cascade Correlation

L'algorithme *Cascade Correlation* (*CasCor*) caractérise plutôt une architecture de traitement qu'un type réseau [Fahlman et Lebiere, 1990]. L'architecture est de type matricielle : les signaux ne passent pas obligatoirement par la couche cachée depuis les neurones d'entrée vers les neurones de sortie, car les neurones sont tous reliés entre eux. Principalement deux idées y sont implémentées : la première est d'ajouter des neurones dans la couche cachée (neurones *candidats*) en les entraînant un par un. La seconde concerne l'entraînement lui-même : il s'agit de maximiser la covariance entre la sortie du candidat et l'erreur résiduelle du réseau en modifiant les poids entre le neurone inséré et les autres neurones. Une fois entraînés ils ne changeront plus, ce qui évite le problème d'instabilité des poids (ang., *moving target problem*) rencontré dans les PMC.

Outre le principe d'apprentissage incrémental, le principal avantage réside dans le fait que ces réseaux sont capable d'apprendre eux-même leur topologie. Leur rapidité leur permet de traiter de grandes bases de données, par exemple l'imagerie d'un tera-octets de la Nasa pour la mission MTPE[1] dans laquelle il faut détecter et quantifier la présence de nuages [Blonda et al., 1993]. Sur un jeu de données, la performance de l'algorithme *CasCor* a dépassé les 99.8%, même si les auteurs reconnaissent avoir utilisé une image simple (nuages ou océan). Une étude plus récente [Diverio et al., 2002] fait état, quant à elle, de performance moindre (69%) pour une image Landsat de 5 canaux lors de la discrimination de 6 classes.

[1] Mission To Planet Earth.

Annexe B

Machines à Vecteur Support

On attribue à Vapnik [Boser et al., 1992; Vapnik, 1995; Cortes et Vapnik, 1995] le fait d'avoir dès 1992 rassemblé différentes notions mathématiques (*maximal margin hyperplanes*, la théorie des noyaux, ...) pour la fondation de la théorie des machines à vecteur support (ang., *Support Vector Machines* ou SVM). Elles sont utilisées dans la découverte de fonctions à partir d'un ensemble de données labellisées, que ce soit pour de la classification ou de la régression. Il s'agit de séparateurs linéaires : leur principe est de trouver un hyperplan séparant un hyperespace E de façon à maximiser la marge entre les données et E. Ils nécessitent la définition de plusieurs paramètres, et malgré le fait que certaines études se soient concentrées sur des méthodes de choix du noyau en fonction de la connaissance du domaine [Brailovsky et al., 1999], ils restent relativement difficiles à comparer. Ces paramètres sont notamment : $C > 0$, le paramètre de pénalité pour le terme d'erreur, le noyau ainsi que tous les paramètres qui lui sont associés. Par exemple, l'un des noyaux les plus performants est la gaussienne RBF (ang., *Radial Basis Function*) : $K(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2}$ avec le paramètre $\gamma > 0$.

En télédétection, la robustesse des SVM a surtout été utilisée pour classifier des images souvent bruitées et complexes (parfois plusieurs centaines de canaux représentant une dimensionalité importante pour les données). Dans certains cas, les SVM n'utilisent que 5% des données d'apprentissage [Brown et al., 1999], ce qui présente l'avantage de réduire fortement l'effort de validation sur ces points. Par exemple, cette méthode a donné de meilleurs résultats que ML[1] (ang., *Maximum Likelihood*) sur des données hyperspectrales DAIS, tout en réduisant le *phénomène de Hugues*[2] [Pal et Mather, 2004]. Dans une autre étude [Gualtieri et Cromp, 1998], des performances moyennes à hauteur de 96% pour un problème de 4 classes et de 87% pour un problème de 16 classes ont été obtenues pour des images hyperspectrales de plusieurs centaines de canaux.

Enfin, nous devons citer une étude publiée dans [Brown et al., 1999] utilisant la capacité de régression des machines à vecteur support (SVM-R). Il s'agissait de retrouver l'information au niveau du sous-pixel, lorsque les spectres de deux ou plusieurs classes[3] se recomposent dans le même pixel, c'est-à-dire de prédire les proportions de différentes classes pures à partir de la composition spectrale connue des pixels d'apprentissage. Les auteurs montrent que les performances de SVM-R sont au-dessus celles obtenus par les modèles linéaires existant jusqu'alors (ang., *Linear Spectral Mixture Model* ou LSMM).

[1] L'algorithme ML est un algorithme supervisé classique en télédétection implémenté dans de nombreux logiciels commerciaux, comme *ENVI*

[2] D'une manière générale, la qualité de classification augmente rapidement avec le nombre de canaux utilisés parmi ceux de l'image initiale, puis baisse progressivement à mesure que l'on s'éloigne d'un certain seuil, quel que soit la taille du jeu d'apprentissage. Ce phénomène a été décrit par Hugues en 1968 [Hughes, 1968].

[3] Un ensemble d'exemples (ici de pixels) étiquetés de la même façon.

Annexe C

Fonctions génétiques

Dans cette section, nous présentons succinctement le rôle et les choix qui ont été effectués pour les opérateurs génétiques principaux. À vocation théorique, elle ne développe que les aspects communs qui concernent l'implémentation de ces opérateurs, qui seront plutôt détaillés dans les sections suivantes. Le lecteur intéressé consultera à profit Goldberg [Goldberg et Sastry, 2001], Michalewicz [Michalewicz, 1996] et Eiben [Eiben et Smith, 2003].

C.1 Fonction d'évaluation

La fonction d'évaluation calcule une note pour chaque individu et représente le noyau d'un algorithme évolutionnaire [Mitchell et al., 1991]. Dépendante fortement du domaine d'application et du type de problème, elle est à déterminer avec soin. Par exemple, en classification *hard* ou *soft*, la fonction d'évaluation tient compte à la fois du nombre d'exemples pour lesquels la règle s'active lorsque la classe correspond et le nombre d'exemples pour lesquels la règle ne s'active pas lorsque la classe ne correspond pas. En classification floue ou par intervalles flous, la fonction d'évaluation tient compte d'une corrélation entre le pourcentage attendu par l'expert et celui calculé par la règle.

La fonction *fitness* F_i d'un individu i mesure l'adaptation ou la performance de cet individu par rapport aux autres. Elle est calculée à partir de l'évaluation de l'individu donnée par la fonction d'évaluation f_i :

$$F_i = \frac{f_i}{\sum f_k} \tag{C.1}$$

Dans un algorithme évolutionnaire, la fonction *fitness* est utilisée comme moyen de pression sélective sur les individus et guide le parcours de l'espace de recherche.

C.2 Opérateur d'initialisation

La génération de la population initiale est l'une des étapes les plus importantes dans un algorithme évolutionnaire. Quelque soit le pouvoir de découverte des opérateurs génétiques, il n'est jamais bon de leur proposer un pool initial trop proche de la solution à trouver. Il y a principalement deux critères à prendre en compte lors de cette initialisation :

- la proximité du pool de départ avec une bonne solution et
- l'obtention d'un pool diversifié, sans quoi la population serait homogène : notons que si la distribution des individus n'est pas assez uniforme l'algorithme risque de converger prématurément sans avoir trouvé de bonne solution.

La création d'un individu initial I_0 à partir d'exemples d'apprentissage est la première phase de la création de la population initiale. L'individu I_0 est ensuite *dupliqué* jusqu'à former une population

complète. Dans le cas d'une initialisation standard, la création des individus est aléatoire, pour tenter d'ensemencer l'espace de recherche de manière uniforme.Enfin, la population est *évaluée* puis triée avant le premier cycle generationnel. L'opérateur de création d'un individu initial étant spécifique à chaque algorithme, nous en avons donné le détail dans le chapitre 5.

C.3 Création de la nouvelle population

D'une manière brève, cette population initiale évolue en d'autres populations, qui se substituent les unes aux autres, à travers une succession de générations, jusqu'à un critère d'arrêt. La procédure standard pour créer la population de la génération suivante à partir d'une population de parents est la suivante :

1. Déterminer `NbEnfantsParGeneration`, le nombre d'enfants que l'on souhaite créer (potentiellement différent de la taille de la population).
2. `NbEnfants` = 0
3. Tant que `NbEnfants` < `NbEnfantsParGeneration`, faire :
   (a) Sélectionner le type de l'opérateur à appliquer, soit le croisement, soit la recopie, avec une probabilité de 80 à 100% pour l'opérateur de croisement.
   (b) Si le croisement est choisi, piocher deux (ou n dans le cas d'un opérateur n-aire[1]) individus dans la population des parents en utilisant une méthode de sélection avec remise (un individu avec une *fitness* élevée peut donc être sélectionné plusieurs fois). Effectuer le croisement avec les parents sélectionnés, pour obtenir n enfants.
   (c) Si la recopie est choisie, piocher un individu en utilisant une méthode de sélection avec remise, le cloner pour obtenir un enfant.
   (d) L'opérateur de mutation est ensuite appelé sur chacun des enfants créés par les opérateurs que nous venons de citer. Chaque gène de ces individus est soumis à la probabilité de mutation. Par exemple, un individu représenté par un vecteur de 100 bits et soumis à une probabilité de mutation de 1%, subit en moyenne une mutation. Si l'enfant provient d'une recopie et qu'aucune mutation n'a été effectuée, récupérer la valeur d'évaluation du *parent* dont l'enfant est le clone, pour gagner du temps.
   (e) Les opérateurs de variation (croisement et mutation) peuvent être suivis d'un opérateur de validation des enfants, dont le rôle est de réduire au mieux le nombre d'évaluations d'individus incohérents (perte de temps). Il y a plusieurs stratégies possibles :
      - Corriger directement les individus incohérents. Il n'y a souvent pas lieu d'en faire un opérateur a part. Si cette correction est directement intégrée à l'opérateur de variation, on parle de macro-mutation (ou de macro-croisement). Dans certains cas, qui nécessitent une fonction d'évaluation complexe, il n'est pas possible d'effectuer de telles corrections sans coût de calcul prohibitif et redondant. De plus, de telles corrections sont parfois difficiles, voire impossibles à réaliser.
      - Supprimer les individus incohérents. Cette stratégie s'apparente à la suivante, car comme l'algorithme boucle tant que le nombre d'enfants `NbEnfantsParGeneration` n'a pas été atteint, à l'issue de la génération de la population d'enfants, tous les enfants sont valides, quitte à ce que la création de la génération d'enfants prenne du temps.
      - Donner une très mauvaise évaluation aux individus incohérents. Contrairement aux deux premières, cette stratégie est très rapide, et fait confiance à l'algorithme de remplacement pour séparer le bon grain de l'ivraie.
   (f) `NbEnfants` est incrémenté du nombre d'individus effectivement insérés dans la population des enfants.

[1] n individus sont exploités lors de la création d'un enfant, avec $n \geq 2$.

4. La fonction d'évaluation est appelée sur les individus dont l'évaluation est à mettre à jour (c'est-à-dire ceux qui ne sont ni clonés, ni pénalisés par l'opérateur de validation).
5. Nous disposons à présent d'une population de parents P, et d'une population d'enfants P', tous évalués. La création de la population de la génération suivante peut être précédée d'une étape de sélection élitiste (forte ou faible), au choix de l'utilisateur. Cette méthode de sélection est particulière car elle peut s'employer quelque que soit la stratégie de remplacement retenue (étape suivante).
   - Dans le cas de l'élitisme fort, on déplace autoritairement le meilleur parent de P vers la nouvelle population sans se poser de question.
   - Dans le cas de l'élitisme faible, on fait les choses de manière plus intelligente : on déplace le meilleur individu de la population $P \cup P'$ vers la nouvelle population.
6. Tant que la taille de la nouvelle population est inférieure à la taille souhaitée, choisir un individu parmi la population $P \cup P'$ par un opérateur de sélection choisi pour cette procédure (nommée remplacement) et le déplacer dans la nouvelle population (remplacement = sélection sans remise).
7. Renvoyer la nouvelle population.

On distingue donc deux occasions lors desquelles un opérateur de sélection s'applique : la sélection des parents et la sélection de la génération suivante (remplacement des survivants). La sélection des parents pioche le matériel génétique nécessaire à l'application des opérateurs de variation que nous verrons dans les deux sections suivantes, c'est-à-dire l'opérateur de croisement et l'opérateur de mutation.

Les stratégies de sélection peuvent être similaires ou différentes pour les opérateurs de sélection des parents et de remplacement, mais le tirage se fait avec remise dans le premier cas, et sans remise dans le second cas. Ces stratégies seront présentées dans la section C.6.

C.4 Opérateur de croisement

L'opérateur de croisement (ang., *crossover*) est utilisé afin d'exploiter la connaissance acquise par les individus de la population courante. Fondé sur le fonctionnement biologique du croisement, il fonctionne par l'échange du matériel génétique entre les deux parents. Si les deux reproducteurs sont meilleurs que la moyenne, on fait l'hypothèse [Vose, 1999] que le matériel génétique résultant contiendra une copie de gènes performants et sera transmis aux enfants. A noter que si l'on choisit toujours les meilleurs parents, on risque une perte de diversité pouvant amener à une convergence prématurée (d'où le choix d'un algorithme de sélection avec une pression de sélection modérée). De plus, le croisement peut découvrir des combinaisons non présentes dans la population des parents, ce qui n'est évidemment pas le cas lorsque l'on se borne à recopier les meilleurs individus dans la population des enfants. L'appel à la fonction de croisement pour créer un nouvel individu se fait avec une probabilité de 80 à 100%. La figure C.1 illustre de manière graphique cet opérateur.

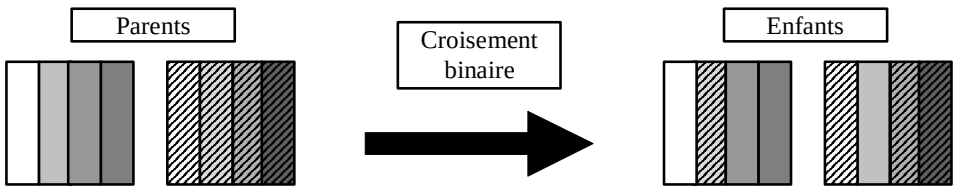


FIG. C.1 – Exemple de l'application d'un opérateur de croisement uniforme binaire.

L'opérateur de croisement ne suffit pas pour l'exploration efficace de l'espace de recherche : notamment l'opérateur est incapable de créer un matériel génétique nouveau. Cet objectif est réalisé par l'opérateur de mutation.

C.5 Opérateur de mutation

La mutation provoque un petit nombre de modifications mineures et aléatoires au sein des gènes d'un individu. Idéalement, l'opérateur de mutation doit être fabriqué de telle sorte qu'un nombre fini d'opérations permette d'atteindre tout point de l'espace de recherche (notion d'ergodicité). C'est donc un opérateur d'exploration, qui sert aussi à maintenir la diversité dans la population d'individus. Les règles sont en fait un ensemble relativement important de variables, surtout lorsque l'on traite des images hyperspectrales. L'opérateur de mutation se déclenche au moins une fois à chaque itération et il est progressif : il convient d'étudier les modifications du génome étape par étape de façon à pouvoir éliminer rapidement une mutation létale. La figure C.2 illustre de manière graphique cet opérateur.

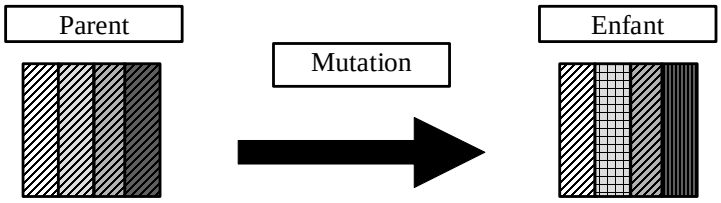


FIG. C.2 – Exemple de l'application d'un opérateur de mutation.

Dans la littérature, nous pouvons observer que certains perfectionnements ont été apportés à cet opérateur. Lorsque le pool semble converger vers un optimum local, la probabilité de mutation est progressivement remontée. Nous pouvons aussi citer le principe de mutation auto-adaptative des stratégies d'évolution [Back et al., 1991], consistant à optimiser la probabilité de déclenchement en codant directement cette dernière dans la structure du chromosome, qui sera soumis aux opérateurs génétiques que nous venons de voir. Au cours du déroulement de l'algorithme, les gènes et les individus ayant des probabilités de mutation élevées auront tendance à disparaître à mesure que la population converge. De même, les gènes ayant des probabilités de mutation trop faibles ne peuvent évoluer favorablement et tendent à être supprimés.

C.6 Opérateurs de sélection

Un algorithme évolutionnaire est généralement constitué de cinq étapes (sélection des parents, croisement, mutation, évaluation et remplacement) et chacune demande la définition d'un opérateur de sélection spécifique par l'utilisateur. Plus précisément, chaque couche réalise une opération précise et retransmet le résultat (le pool génétique obtenu) à la couche suivante qui le traite à son tour. Il est donc possible de sauvegarder à certaines étapes le contenu actuel du pool, afin de le réutiliser plus tard, comme dans le cas du remplacement des stratégies d'évolution qui reprend certains individus de la population des parents et d'autres de la dernière population générée [Beyer et Schwefel, 2002]. Malheureusement, la mise à jour des individus (calcul de la *fitness*, paramètres de certains individus complexes comme les arbres en programmation génétique, comparaisons deux à deux si l'on souhaite éliminer les doublons, ...) doit être effectuée à chaque fois pour permettre à l'opérateur génétique suivant d'utiliser ces critères pour la sélection, ce qui est très coûteux en temps.

Chaque couche a besoin de matériel (les individus) qu'elle va piocher dans la couche précédente. Il est important d'avoir des fonctions de sélection d'individus différentes et adaptées selon le type de la couche sollicitant ces individus. La sélection est donc vue comme un problème à part entier, et l'utilisateur peut en pratique soit choisir les relations couche/fonction-de-sélection qui lui semblent les mieux adaptées à chacun des cas dans un catalogue de fonctions déjà définies, soit définir lui-même une telle fonction. Les

travaux de Blickle [Blickle et Thiele, 1995] ont permis d'étudier et de justifier formellement de nombreuses méthodes de sélection, dont nous nous inspirons ici.

Il faut distinguer trois étapes de l'algorithme, où il est nécessaire de sélectionner un individu, ce qui nous donne autant de fonctions de sélection à définir :

- La sélection pour le croisement (avec remise) est utilisée dans le choix des deux parents à croiser.
- La sélection pour la recopie (avec remise) est utilisée dans le choix d'un parent à cloner.
- La sélection pour le remplacement choisit dans la population des parents et des enfants les individus à conserver pour la génération suivante. La stratégie retenue influence grandement la possibilité pour la population de converger à long ou à court terme. La stratégie du tournoi est à conseiller à défaut d'une autre, lorsque ce paysage n'est pas connu.

Certains opérateurs de sélection (pas celui du tournoi, par exemple) prennent en entrée un pool trié selon l'un des critères caractéristiques de la structure des règles. Par exemple, pour une structure arborescente, la population pourra être triée selon la *fitness*, la taille des arbres en nombre de nœuds ou la profondeur maximale des arbres. L'opérateur renvoie en sortie l'individu sélectionné. Pour les opérateurs génétiques nécessitant le choix de deux individus (croisement), l'opérateur de sélection correspondant est appelé deux fois. Par sa conception, ce protocole autorise plusieurs individus à être sélectionnés plusieurs fois (par exemple, le meilleur individu), ce qui leur permet de pérenniser leurs gènes. Les opérateurs sont représentés graphiquement sur la figure C.3.

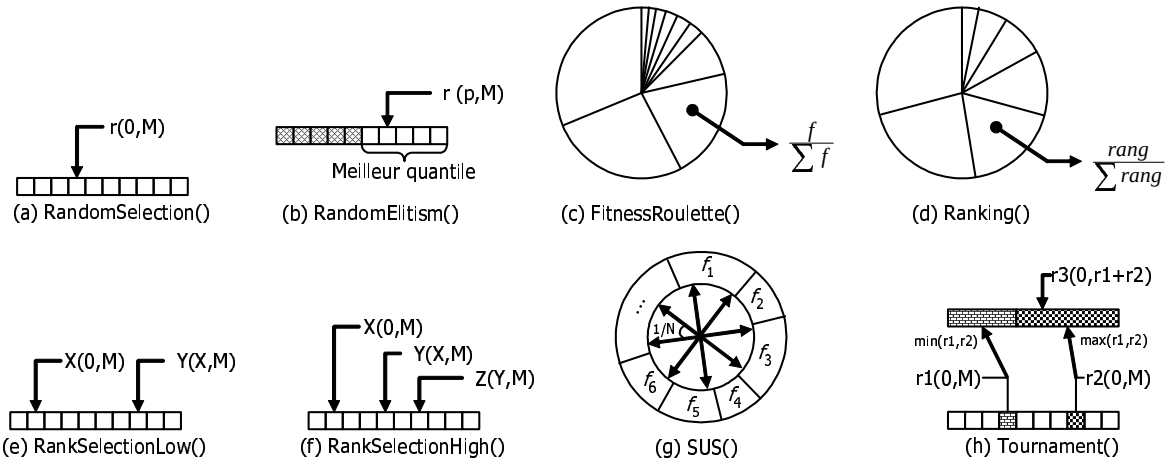


FIG. C.3 – Représentations graphiques d'opérateurs standards et non standards de sélection. X(a,b) renvoie un nombre aléatoire X dans l'intervalle $[a;b]$ et M est la taille de la population.

C.6.1 Sélections fondées sur la roulette

La roulette correspond au tirage aléatoire des individus dans une sorte de camembert dont chaque part a une taille proportionnelle à la *fitness* de l'individu concerné et donc chaque probabilité de sélection est directement liée à sa qualité. La figure C.3(c) présente l'explication de cette stratégie.

Cet opérateur est considéré par certains spécialistes comme un opérateur que l'on ne doit pas utiliser si l'on ne connaît pas assez bien le paysage de la *fitness* [Collet et al., 2000]. En effet, il n'offre pas ou peu de contrôle sur la pression de sélection : notamment, un petit sous-ensemble d'individus de force élevée peut absorber le reste de la population lors des générations suivantes et peut conduire à une convergence prématurée à cause d'un manque de diversification (la *fitness* se stabilisant alors loin du maximum souhaité).

C.6.2 Sélections fondées sur le rang

La sélection selon le rang (ang., *ranking*) est une méthode fondée sur l'ordre des individus dans un pool ordonné par leurs notes. Les notes proposées par la fonction *fitness* ne sont qu'une représentation relative de la qualité d'un individu par rapport à ses congénères. Ici, la représentation est relative par rapport à leur rang. Il existe plusieurs méthodes pour définir la probabilité de sélection d'un individu à partir de son rang. Cette probabilité peut être définie directement comme illustré sur la figure C.3(d) ou indirectement en utilisant par exemple des pointeurs intermédiaires (figures C.3(e) et C.3(f)).

Dans le cas direct, chaque part à une taille proportionnelle au rang de l'individu. D'autres stratégies existent, par exemple les variantes polynomiales utilisant R^α, avec $\alpha > 0$ à la place de R (le rang de l'individu). Un individu très faible n'est donc pas éliminé. De plus, la fonction de qualité n'a pas pour but de produire une note absolue (il faudrait qu'un individu réellement deux fois moins performant dans la première génération que dans la dernière obtienne une note deux fois plus faible pour que ses chances de sélection soient comparables, ce qui est difficile à garantir dans la pratique), mais simplement de trier les individus entre eux. Ce relativisme permet de ne pas perturber les opérateurs génétiques lorsque les notes sont extrêmes. Seules comptent donc les positions relatives des individus et non la valeur de la fonction *fitness* dont l'échelle peut être arbitraire.

La sélection par le rang présentée sur la figure C.3(e), qui n'est pas une méthode de sélection standard, applique le principe suivant : disposant d'une liste de M individus triés (le premier étant le meilleur), on tire un nombre aléatoire X entre 1 et M inclus. Puis on tire un second nombre aléatoire Y entre 1 et X. L'individu sélectionné est alors Y. Étudions la probabilité de sélection d'un individu donné dans ce cas.

Soit $P_M(\{X = i\})$ la probabilité qu'en choisissant X entre 1 et M on obtienne i. Puisque l'on dispose d'un espace de valeurs discrètes, cette probabilité est facile à calculer :

$$P_M(\{X = i\}) = \frac{1}{M} \tag{C.2}$$

Soit $P'_M(\{Y = i\})$ la probabilité qu'en choisissant Y selon la procédure décrite ci-dessus, on obtienne i. On a :

$$P'_M(\{Y = i\}) = \sum_{j=i}^{M} [P_M(\{X = j\}).P_j(\{Y = i\})] \tag{C.3}$$

En effet, il faut que $1 \leq i \leq X \leq M$. Pour tous les choix de j possibles pour X, la probabilité que i soit tiré entre 1 et j est $P_j(\{Y = i\})$. Par équivalence, on obtient une équation ne dépendant que de i et de M :

$$P'_M(\{Y = i\}) = \sum_{j=i}^{M} \left[\frac{1}{M}.\frac{1}{j}\right] = \frac{1}{M}.\sum_{j=i}^{M} \frac{1}{j} \tag{C.4}$$

Pour l'exemple, le choix du meilleur individu dans une population de 100 individus se calcule ainsi :

$$P'_{100}(\{Y = 1\}) = \frac{1}{100}.\sum_{j=1}^{100} \frac{1}{j} \simeq 0.0519 \tag{C.5}$$

Il y a donc environ 5% de très bons individus qui passent dans la génération suivante grâce à cette méthode. S'il n'y a que 5 individus, la probabilité qu'a le meilleur de passer est de 45%. Cette méthode donne de bons résultat pour les populations d'au moins 100 à 200 individus.

Contrairement à la stratégie dite de la *roue de la roulette* distinguant les individus très forts des individus forts et moyens, la sélection selon le rang permet de privilégier les plus forts sans oublier les faibles ni ceux qui n'ont pas une note suffisante pour s'imposer. Par exemple, si cinq individus ont une excellente évaluation par rapport aux autres parmi une population d'une centaine d'individus, il y a de fortes chances que le sixième individu puisse être sélectionné, même s'il a une *fitness* faible, ce qui est important pour éviter le phénomène d'absorption élitiste, évoqué dans la section précédente. Les sélections

par le rang sont en pratique meilleures que les sélections par la *roue de la roulette*, mais elles nécessitent un tri préalable de la population. Cette opération, coûteuse en temps, n'est pas nécessaire dans d'autres types de sélection, comme celle par tournoi (voir la section C.6.4). De plus, la pression de sélection est plus facilement contrôlable que pour la *roue de la roulette*, mais moins que pour le tournoi.

C.6.3 Sélection par échantillonnage stochastique universel

Cette méthode, aussi connue sous le nom de *roue de la roulette généralisée* ou *Stochastic Universal Sampling* (SUS), a été proposée par Bäker [Bäker, 1987]. Elle est illustrée sur la figure C.3(g). Comme dans le cas de la *roue de la roulette*, les parts ont une taille proportionnelle à la *fitness* de chaque individu. Ensuite, un sélecteur comprenant autant de pointeurs que d'individus à choisir, et espacés régulièrement, est utilisé de manière aléatoire pour sélectionner en une seule étape les individus à conserver.

Cette méthode permet aux meilleurs individus d'être sélectionnés plus souvent, avec une pression de sélection moins forte comparée à celle de la roulette. Les individus qui ont des valeurs de *fitness* plus faibles auront donc plus de chance de participer à la génération suivante. De plus, contrairement aux méthodes par rang, il ne faut pas trier la population au préalable.

C.6.4 Sélection par tournoi

L'utilisateur non expérimenté avec les algorithmes évolutionnaires pourra généralement considérer le tournoi comme l'une des meilleures stratégies. En effet, on peut contrôler la pression de sélection finement, contrairement à la *roue de la roulette* ou au *ranking*, grâce à la multitude de stratégies ou de paramètres disponibles. On emploiera un tournoi n-aire pour une pression de sélection forte, dont la force décroît en fonction du nombre d'individus participant au tournoi (généralement on considère une taille de 2 à 5 individus) ou un tournoi stochastique pour une pression de sélection plus faible. Le tournoi n-aire est une méthode déterministe qui consiste à choisir directement le meilleur individu parmi ceux participant au tournoi. Le tournoi stochastique (illustré sur la figure C.3(h)) sélectionne le meilleur individu avec une probabilité comprise entre 50% (pur hasard) et 100% (équivalent à un tournoi binaire). Un autre avantage est sa rapidité, comparé à d'autres stratégies comme la *roue de la roulette*. Puisque seule importe la *fitness* pour les individus participant au tournoi, un tri préalable de la population complète n'est pas nécessaire. Enfin, elle fait partie des stratégies de sélection, avec les sélections par rang, à ne pas dépendre directement du paysage de la *fitness*. Les sélections qui en dépendent trop fortement présentent le problème que leur pression de sélection est incontrôlable, pouvant mener à une convergence prématurée ou une convergence trop lente, suivant le paysage de la *fitness*. Sauf à savoir exactement à quoi ressemble le paysage de la *fitness*, et à condition que ce paysage fournisse une pression de sélection adéquate, il vaut mieux utiliser un algorithme de sélection à pression de sélection contrôlée par l'utilisateur. La sélection par tournoi présente donc une bonne efficacité en plus de sa simplicité algorithmique.

C.7 Critères d'arrêt

Les critères d'arrêt permettent de savoir s'il faut arrêter l'algorithme : on s'assure que la population a convergé selon des critères statistiques définis de telle sorte qu'il soit probable que le meilleur individu est proche d'un maximum global ou que la population n'évoluera plus [Schmitt, 2004]. Nous allons détailler plusieurs critères utilisés couramment. Ces critères sont inclusifs, c'est-à-dire qu'il suffit qu'il y en ait un seul qui s'active pour que l'algorithme soit stoppé.

Nombre de générations. Un contrôle direct sur la durée de l'exécution s'effectue par le choix du nombre maximum de générations à calculer ($N_{\max}$). Cette limite nous donne l'assurance que l'algorithme se terminera dans tous les cas.

Seuil sur la *fitness*. La qualité des individus est bornée par 1. Si la *fitness* d'un individu dépasse un seuil fixé, par exemple pris dans l'intervalle $[0.8; 1[$, nous pouvons considérer qu'il y a peu de chance d'avoir un individu encore plus performant : nous demandons donc à l'algorithme de s'arrêter.

Convergence de la *fitness*. Cependant, la note se stabilise souvent bien loin du seuil fixé. Il faut donc un moyen de détecter le ralentissement de l'apprentissage, afin d'éviter d'attendre indéfiniment la fin des itérations. On utilise donc comme critère d'arrêt l'évolution des notes des meilleurs individus à chaque génération : si celles-ci se stabilisent et forment des paliers trop longs, on met fin à l'apprentissage. Plus formellement, soit Q_k la qualité du meilleur individu du pool génétique obtenu au cours de la k-ième dernière génération, avec $k \geq 1$, et Q_0 la qualité du meilleur individu de la génération courante. L'algorithme est interrompu si l'inégalité suivante est vérifiée :

$$\left| \frac{\sum_{k=1}^{P} Q_k}{P} - Q_0 \right| \leq \epsilon \tag{C.6}$$

où P représente la longueur maximale d'un palier et ϵ la variation maximale de ce palier par rapport à la note courante : si on choisit une petite valeur pour ϵ, la variation de la note devra être quasiment constante pour que l'algorithme s'arrête.

Convergence de la population. La *fitness* moyenne Q_μ de la population est comparée à celle (Q_0) du meilleur individu. Lorsque $Q_\mu \geq \tau Q_0$, avec τ un paramètre défini par l'utilisateur, l'algorithme est déclaré comme ayant convergé.

L'utilisateur possède le contrôle de la totalité de cette paramétrisation. Des valeurs par défaut lui sont proposées, en accord avec nos expérimentations par rapport à la taille des règles, la complexité de la représentation et la performance de l'apprentissage : le palier P doit varier entre 10 et 20 générations, selon la fiabilité que l'on désire pour le test de stabilité de la solution. La variation ϵ, quand à elle, peut se satisfaire d'une valeur très faible (10^{-4}) voire nulle pour les génomes à valeurs entières, car les quantités discrètes rendent aisément l'évolution des notes constante au bout d'un certain point. Le nombre de générations $N_{\max}$ ne devrait pas excéder 200 à 500, même dans les problèmes les plus complexes. Enfin, le taux τ peut être choisi dans l'intervalle $[0.6; 0.9]$.

Annexe D

Le projet TIDE

Le projet TIDE (*Tidal Inlets Dynamics and Environment*) [TIDE, 2005] est un projet de recherche européen d'une durée de quatre ans (2002-2005), dirigé par le professeur Marco Marani de l'université de Padova (UNPADU). Les zones à marées, telles que les lagunes et les estuaires, sont des environnements complexes et sensibles à des changements morphologiques et écologiques rapides, souvent en réponse à la forte présence humaine [Marani et al., 2004]. Les problèmes liés à ces zones ont été identifiés par la convention internationale de Ramsar sur les zones humides : 445 des 844 zones humides protégées d'importance internationale énumérées par la convention sont situées en Europe. Le projet TIDE vise à développer des modèles complets concernant les zones à marées incorporant des descriptions des processus physiques et écologiques [Marani et al., 2003]. Les modèles en question ne peuvent pas décrire séparément les phénomènes biologiques et physiques sans prévoir leurs conséquences dans le système global. Pour progresser, la recherche dans ce domaine nécessite une nouvelle génération de modèles économiques, adaptés aux environnements de grande envergure. Les conséquences socio-économiques de l'altération de ces systèmes font partie intégrante du programme du projet TIDE [Marani et al., 2005; Silvestri et Marani, 2004; Silvestri et al., 2005].

Ce projet implique plusieurs laboratoires, en Italie, l'université de Padoue (UNPADU), l'université de Trente (UNITN), l'université de Venise (UNIVE) et l'Institut Vénitien des Arts et des Lettres (IVSLA), le groupe industriel allemand Toposys, au Royaume-Uni, le Centre Scientifique sur les Systèmes Environnementaux (ESSC) et l'université St. Andrews (USTAN) et en France, l'université Louis Pasteur de Strasbourg.

La figure D.1 présente les relations entre les groupes de travaux du projet TIDE. Nos recherches se situent dans le groupe de travail numéro 6, dirigé par Massimo Menenti, de l'université Louis Pasteur de Strasbourg.

L'objectif de ce groupe de travail, nommé *classification de végétations microphytobenthiques à partir de données de télédétection*, est de développer une recherche scientifique visant à connaître l'utilisation optimale des données radiométriques hyperspectrales pour la classification de végétations halophytiques et comprendre les relations microbiennes benthiques dans les zones à marées [Silvestri et al., 2002; Silvestri et al., 2003]. Quatre objectifs interdépendants ont été identifiés :

1. Traiter les données de télédétection pour en assurer l'assemblage géométrique optimal et rassembler les données de terrain nécessaires pour l'extraction d'informations quantitatives sur les classes d'intérêt.
2. Développer des méthodes génériques pour sélectionner les données radiométriques les plus utiles et les plus précises sur la réflectance spectrale des objets du sol.
3. Développer des classifications efficaces, précises et robustes, ainsi que des algorithmes de démixtion (ang., *unmixing*) pour gérer la complexité des données produites par les instruments hyperspectraux.
4. Superposer les classifications de végétations benthiques dans les zones à marées pour comprendre leurs relations et leurs propriétés géomorphiques.

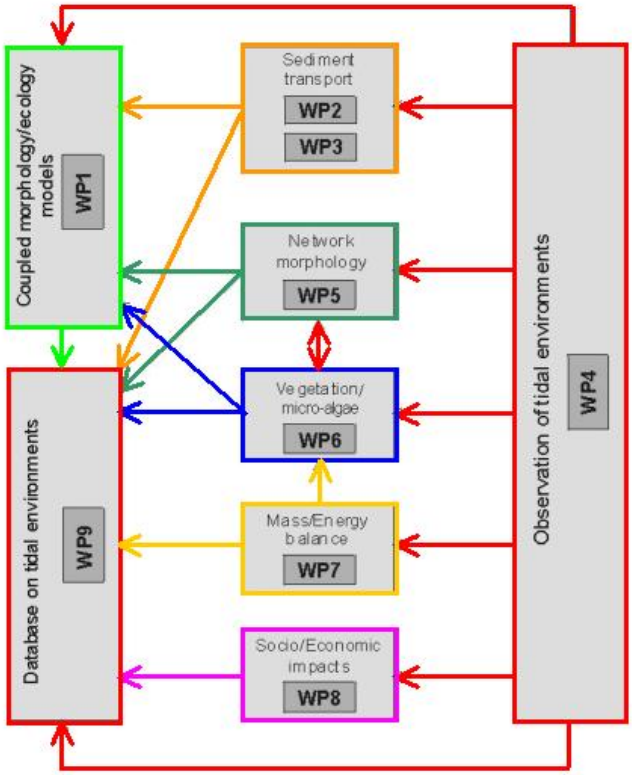


FIG. D.1 – Les groupes de travail du projet TIDE.

Cette thèse a été financée par le projet TIDE et les recherches scientifiques développées durant nos travaux ont permis d'atteindre certains des objectifs définis par le groupe de travail numéro 6. Cela concerne notamment le développement d'algorithmes efficaces et robustes pour traiter les jeux d'apprentissage *hard*, *soft* et flous, et la découverte et l'extraction de connaissances structurées sous la forme de règles de classification, stockées dans des bases de règles réutilisables, visant à permettre une compréhension plus aisée de la complexité des données.

Annexe E

Le projet FoDoMust

Le projet FoDoMust, dont nos travaux font partie, est inscrit dans l'ACI *Masse de Données*. L'amélioration des dispositifs expérimentaux et la part grandissante de l'informatique et de la simulation dans la plupart des champs disciplinaires conduit à la production de données en quantité de plus en plus importante. L'objectif de cette ACI est de fortement dynamiser la recherche sur l'ensemble des aspects relatifs à ces grandes masses de données : acquisition, stockage, transmission, traitement, modélisation, représentation, structuration, indexation, interrogation, comparaison, manipulation, classification, fusion, extraction de sens, apprentissage, visualisation [MDD, 2005].

Le but du projet FoDoMust (*Fouille de Données Multi-Stratégies*), coordonné par Pierre Gançarski de l'université Louis Pasteur de Strasbourg, est d'extraire et qualifier la végétation urbaine à partir de bases de données images [FoDoMuST, 2005]. La profusion des informations géographiques (photographies aériennes, images satellites, etc) permet un suivi et une gestion durable des territoires (végétation urbaine). L'augmentation des sources de données, leur hétérogénéité et leur complexité rendent leur utilisation difficile par des utilisateurs non-spécialistes de l'interprétation et du traitement d'image. Les objectifs du projet, associés à l'imagerie spatiale, sont d'une part, de proposer une méthode d'aide à l'interprétation à partir d'une masse de données images et d'autre part, de définir un processus complet de fouilles de données (structuration, construction des « objets », classification et interprétation de l'information) permettant une utilisation conjointe et complémentaire des différentes sources. Ce dernier aspect est rarement pris en compte dans les méthodes actuelles d'extraction. Le verrou principal réside dans la nécessité d'utiliser une multi-formalisation à plusieurs niveaux d'abstraction.

La solution envisagée par le projet consiste en une approche multi-stratégie dans le processus de fouille de données. L'objectif premier de ces travaux est d'étudier et de définir des méthodes et outils permettant une utilisation conjointe de plusieurs sources de connaissances et d'images lors de l'identification, la localisation et la formalisation des éléments du tissu urbain (surfaces minéralisées, végétation, eau). De fait, les données manipulées ne sont plus sous la forme classique attributs-valeurs simple et immédiate car elles peuvent être issues de sources diverses comme des capteurs ou sources physiques différentes (capteurs radiométriques, etc), représenter la scène à des dates différentes (différents passages d'un satellite) ou enfin, regrouper des informations de natures totalement différentes (texte, taxonomie, référentiel, etc). Dans le cadre de cette ACI, l'objectif global est de proposer un processus complet de sélection, d'extraction et d'interprétation de connaissances à partir de bases de données d'images et de connaissances du domaine considéré en quatre phases :

1. Structuration et organisation des données.
2. Construction des « objets ».
3. Classification multi-stratégie.
4. Aide à l'interprétation.

Ce projet, d'une durée de trois ans (2004-2006), implique plusieurs laboratoires français : à Strasbourg, le laboratoire des Sciences de l'Image, de l'Informatique et de la Télédétection (LSIIT) et le labo-

ratoire Image et Ville (LIV), et à Lyon, l'équipe de Recherche en Ingénierie des Connaissances (ERIC).

Les travaux publiés dans ce mémoire s'inscrivent dans certains objectifs du projet FoDoMust. À travers la découverte et l'extraction de règles de classification, ils proposent une explication compréhensible des données et constituent dès lors l'une des stratégies d'extraction et d'interprétation des connaissances envisagées par le projet. La collaboration initiée par le projet FoDoMust avec ces laboratoires vise aussi à expérimenter et à valider les différents prototypes que nous avons développés. De nombreuses perspectives offertes par cette thèse seront étudiées durant la suite du projet, par exemple la fouille de données multi-sources ainsi que la construction de règles à représentation contextuelle.

Annexe F

La plate-forme VPlat

Nous avons conçu une plate-forme, nommée **VPlat**, qui rassemble la totalité des algorithmes, étudiés dans le cadre de cette thèse, sous la forme d'un ensemble de librairies. Ces librairies, utilisées pour réaliser les différentes expérimentations, ont été implantées en partie dans un logiciel possédant une interface utilisateur conviviale, le logiciel ICU [ICU, 2005]. Les trois objectifs principaux de la plate-forme sont les suivants :

1. Abstraire le format des données d'entrée (images, textes, bases de données, ...) et de sortie (règles, classifications). Cette abstraction facilite l'importation, l'exportation et la gestion de nouveaux formats de données.
2. Abstraire le paramétrage et le lancement des différentes méthodes d'apprentissage. Le paramétrage peut être ainsi lu et écrit dans divers fichiers de configuration ce qui est utile pour exécuter les algorithmes plusieurs fois de la même façon ou pour proposer à l'utilisateur plusieurs fichiers de configuration automatique par défaut (apprentissage rapide, apprentissage haute qualité, ...).
3. Proposer des modèles de validation génériques manipulant des données et des méthodes d'apprentissage abstraites. L'abstraction des données et des modèles permet la mise en place de procédures de validation automatiques (validation croisée, ...) ou des apprentissages *batchs* (extraction de statistiques d'apprentissage) qui seront indépendants des algorithmes étudiés.

La figure F.1 détaille l'architecture de la plate-forme et montre les relations entre les différentes classes. Le bloc « MODEL » [B1] contient toutes les fonctions d'apprentissage ainsi que les fonctions qui permettent d'évaluer les règles sur les échantillons (classifications *hard*, *soft*, floue ou par intervalles flous). Il renferme aussi les fonctions qui gèrent les bases de règles (lecture, écriture, affichage, statistiques). Le bloc « DATA » [B2] rassemble les filtres pour les divers formats de fichier et comprend la gestion de la lecture et de l'écriture pour ces formats. Les données en question doivent obligatoirement être sous la forme d'un ensemble d'échantillons, où chaque échantillon est une séquence de valeurs, chacune étant associée à un attribut particulier.

Cette plate-forme est capable de lire ou écrire les données dans les formats suivants : les données tabulaires (à séparateurs divers) TXT et SEQ, le format ARFF, les descriptions des régions d'intérêt en ASCII (*Region of Interest*), les formats d'image classiques BMP, PGM, PPM et TIFF, les formats d'image spectrale BIL, BIP, BSQ et TOR, ainsi que les descriptions de légendes de classes LEG. Selon leurs représentations, les bases de règles peuvent être exportées dans les formats TXT et HTML (représentations plates) ou dans les formats DOT et PNG (représentations arborescentes). Les formats HTML et PNG permettent une visualisation aisée des règles par l'utilisateur.

Les méthodes d'apprentissage incluses dans **VPlat** sont **ICU**, **XCS-R**, **ICUX**, **ProgGen**, **GramGen**, **XCS5** et **FANN** (*Fast Artificial Neural Network Library*, [FANN, 2005]). Un module est aussi dédié à l'interprétation et à la traduction en C des arbres de décision sous format texte produits par le logiciel Weka (notamment pour l'algorithme J48, c'est-à-dire **C4.5**), ce qui permet de les tester et les valider sur des échantillons spécifiques de données de télédétection. La plate-forme comprend aussi la

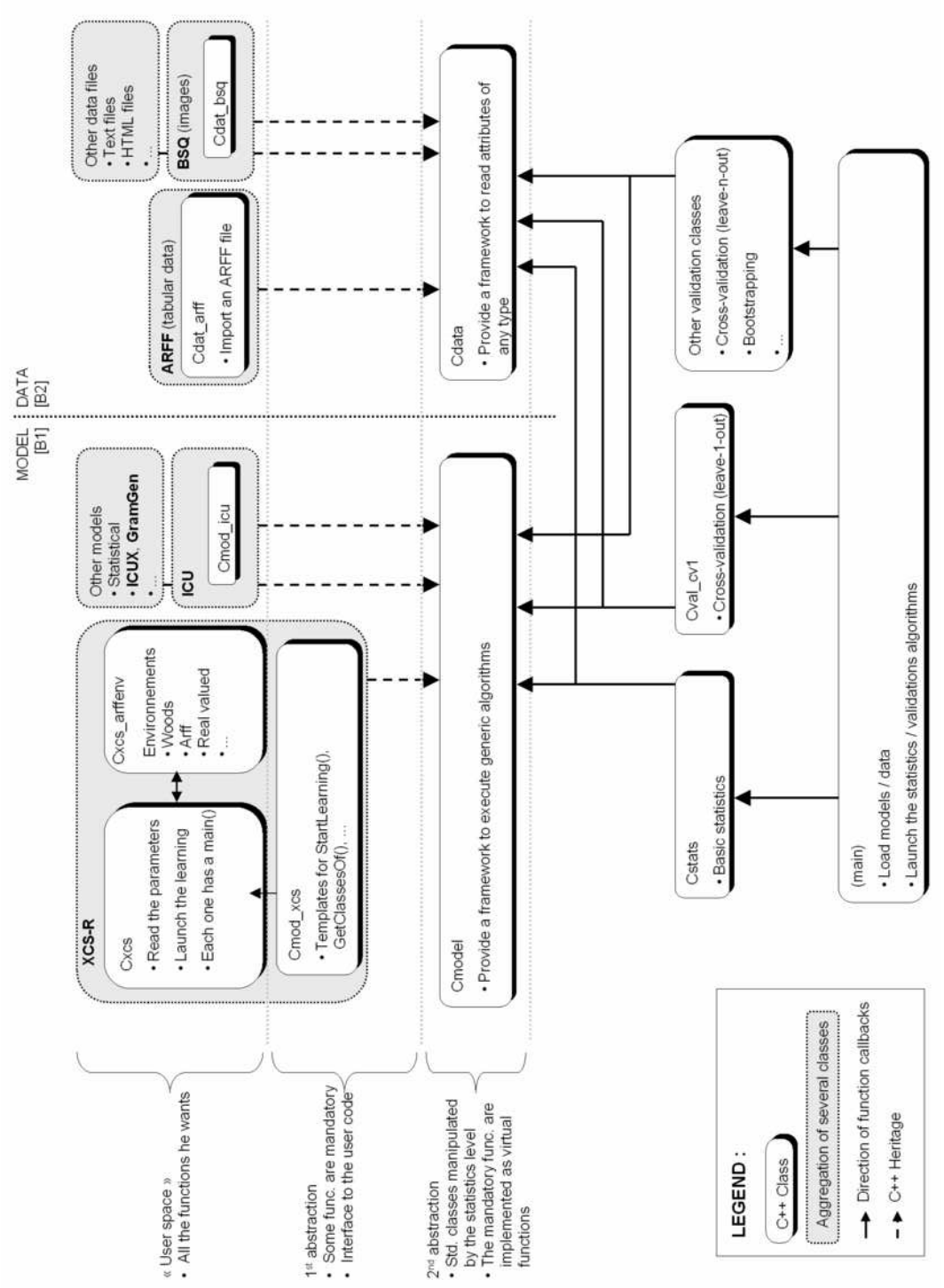


FIG. F.1 – Diagramme de classes pour la plate-forme **VPlat**.

librairie GALib [Corcoran, 1993] ainsi que des corps de fonctions personnalisés qui ont permis de réaliser les expérimentations liées à l'utilisation de génomes dans des algorithmes génétiques simples.

Les stratégies de validation qui sont implémentées dans **VPlat** sont les suivantes : le *holding-out*, la validation croisée à k partitions, le *bootstrapping*, le *jackknifing* et la génération de courbes ROC. D'autres fonctions diverses sont présentes pour le calcul des mesures de qualité, la génération des visualisations (spectrogrammes, ...), l'échantillonnage des exemples ou le calcul de certains pré-traitements, comme les enveloppes convexes (voir la section 3.4.2).

Cette plate-forme a été programmée en C++, pour des raisons de rapidité et de portabilité (nous avons réalisé nos expérimentations sur GNU/Linux et sur Windows). De plus amples informations sont disponible sur le site suivant : http://lsiit.u-strasbg.fr/afd.

Liste des tableaux

Table des figures

Table des algorithmes

Bibliographie

A525G (2005). Les algorithmes génétiques, document en ligne disponible sur http://www.a525g.com/intelligence-artificielle/algorithme-genetique.php.

Andrews, R., Diederich, J., et Tickle, A. (1995). A survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge Based Systems*, 8(6) :373–389.

Andrews, R. et Geva, S. (1995). *Hybrid problems, hybrid solutions*, chapter RULEX and CEBP Networks as the Basis for a Rule Refinement System, pages 1–12. IOS. Press.

Andrews, R. et Geva, S. (1996). Rules and local function networks. In *Proceedings of the Rule Extraction From Trained Artificial Neural Networks Workshop, Artificial Intelligence and Simulation of Behaviour*, Brighton.

Back, T., Hoffmeister, F., et Schwefel, H. (1991). A survey of evolution strategies. In *Proceedings of the 4th International Conference on Genetic Algorithms*, pages 2–9, San Diego.

Bagnall, A. J. et Cawley, G. C. (2003). Learning classifier systems for data mining : A comparison of XCS with other classifiers for the forest cover dataset. In *Proceedings of the IEEE/INNS International Joint Conference on Artificial Neural Networks (IJCNN-2003)*, Portland.

Barnsley, M. J., Hobson, P., Hesley, Z., Evans-Jones, K., Quaife, T., Lewis, P., Disney, M., Muller, J.-P., Strahler, A., Lucht, W., et Hyman, A. (1998). Determination and validation of land-surface biophysical properties using SPOT-4 vegetation and HRVIR : Interim report. Technical report, University of Wales Swansea.

Barto, A. G. et Anandan, P. (1985). Pattern-recognizing stochastic learning automata. *IEEE Transactions on Systems, Man and Cybernetics*, 15 :360–375.

Barto, A. G., Anandan, P., et Anderson, C. (1985). Cooperativity in networks of pattern recognizing stochastic learning automata. In *Proceedings of the Fourth Yale Workshop on Applications of Adaptive Systems Theory*, New Haven.

Baum, E. B. et Durdanovic, I. (2000). An evolutionary post production system. Technical report, NEC Research Institute, Princeton.

Benediktsson, J. A., Swain, P. H., et Erase, O. K. (1990). Neural network approaches versus statistical methods in classification of multisource remote sensing data. *IEEE Transactions on Geoscience and Remote Sensing*, 28 :540–551.

Bernadó-Mansilla, E. et Garrell-Guiu, J. M. (2003). Accuracy-based learning classifier systems : Models, analysis and applications to classification tasks. *Evolutionary Computation*, 11(3) :209–238.

Beyer, H.-G. et Schwefel, H.-P. (2002). Evolution strategies. a comprehensive introduction. *Natural Computing : an international journal*, 1(1) :3–52.

Bijaoui, A., Starck, J.-L., et Murtagh, F. (1994). Restauration des images multi-échelles par l'algorithme à trous. *Traitement du Signal*, 3(11).

Bäker, J. E. (1987). Reducing bias and inefficiency in the selection algorithm. In *Proceedings of second International Conference on Genetic Algorithms (ICGA2)*, pages 14–21.

Blake, C. L. et Merz, C. J. (1998). UCI Repository of machine learning databases, disponible en ligne sur http://www.ics.uci.edu/~mlearn/MLRepository.html, University of California, Irvine, Dept. of Information and Computer Sciences.

Blanzieri, E. (1998). *Learning Algorithms for Radial Basis Function Networks : Synthesis, Experiments and Cognitive Modelling*. PhD thesis, University and Polytechnic, Turin.

Blekas, K., Likas, A., et Stafylopatis, A. (1997). A fuzzy neural network approach to classification based on proximity characteristics of patterns. In *9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97)*, California.

Blickle, T. et Thiele, L. (1995). A comparison of selection schemes used in genetic algorithms. Technical Report 11 Version 2, Computer Engineering and Communication Network Lab, Swiss Federal Institute of Technology, Zurich.

Blonda, P., Pasquariello, G., et Smid, J. (1993). Comparison of backpropagation, cascade-correlation and kokonen algorithms for cloud retrieval. In *Proceedings of 1993 International Joint Conference on Neural Networks (IJCNN'93)*, Nagoya.

Bock, H. et Diday, E. (2000). *Analysis of Symbolic Data. Exploratory methods for extracting statistical information from complex data*. Springer Verlag, Heidelberg, 425p., ISBN 3-540-66619-2 (seconde édition).

Bolis, E., Zerbi, C., Collet, P., Louchet, J., et Lutton, E. (2001). A GP artificial ant for image processing : Preliminary experiments with EASEA. In *EUROGP 2001*, Como.

Booker, L. B., Goldberg, D. E., et Holland, J. H. (1989). Classifier systems and genetic algorithms. *Artificial Intelligence*, 40 :235–282.

Boser, B. E., Guyon, I., et Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152.

Brailovsky, V. L., Barzilay, O., et Shahave, R. (1999). On global, local, mixed and neighborhood kernels for support vector machines. *Pattern Recognition Letters*, 20(11-13) :1183–1190.

Brown, M., Lewis, H. G., et Gunn, S. R. (1999). Support vector machines for optimal classification and spectral unmixing. *Ecological Modelling*, 120 :167–179.

Brumby, S. P., Harvey, N. R., Perkins, S., Porter, R. B., Szymanski, J. J., Theiler, J., et Bloch, J. J. (2000). A genetic algorithm for combining new and existing image processing tools for multispectral imagery. In *Proceedings of SPIE 4049 Algorithms for Multispectral, Hyperspectral, and Ultraspectral Imagery VI*, pages 480–490.

Butz, M. V., Goldberg, D. E., Lanzi, P. L., et Sastry, K. (2004). Bounding the population size to ensure niche support in XCS. Technical Report 2004033, Department of General Engineering The University of Illinois, Urbana-Champaign.

Butz, M. V., Sastry, K., et Goldberg, D. E. (2003). Tournament selection in XCS. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1857–1869.

Butz, M. V. et Wilson, S. W. (2002). An algorithmic description of XCS. *Soft Computing*, 6 :144–153.

Cao, H. (2001). IMPUTE : A SAS application system for missing value imputations. Technical report, Institute for Social Research, University of Michigan.

Carse, B. et Pipe, A. G. (2001). X-FCS : A fuzzy classifier system using accuracy based fitness - first results. In *Proceedings of the International Conference on Fuzzy Logic and Technology (EUSFLAT)*, pages 195–198.

CASI (2005). Satellite d'observation de la Terre (SPOT, CASI), document en ligne disponible sur http://www.ccrs.nrcan.gc.ca/ccrs/eduref/tutorial/chap2/c2p12f.html.

Ceroni, A., Pelikan, M., et Goldberg, D. E. (2001). Convergence-time models for the simple genetic algorithm with finite population. Technical Report 2001028, Illinois Genetic Algorithms Laboratory, University of Illinois, Urbana-Champaign.

Chang, C.-C. et Lin, C.-J. (2001). LIBSVM : a library for support vector machines, disponible en ligne sur http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Chavent, M. (1998). A monothetic clustering method. *Pattern Recognition Letters*, 19(11) :989–996.

Chemin, Y., Honda, K., et Ines, A. V. M. (2005). Genetic algorithm for assimilating remotely sensed evapotranspiration data using a soil-water-atmosphere-plant model - implementation issues. *Technical letter, International Journal of Geoinformatics*, 1(1) :87–90.

Clark, P. et Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning*, 3 :261–283.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20 :37–46.

Collet, P., Lutton, E., Schoenauer, M., et Louchet, J. (2000). Take it EASEA. In *Proceedings of PPSN VI, Springer, LNCS 1917*, pages 891–901, Paris.

Corcoran, A. L. (1993). GALib : Library of GA routines written in C, librairie disponible sur http://lancet.mit.edu/ga/.

Cordon, O., Herrera, F., Gomide, F., Hoffmann, F., et Magdalena, L. (2001). Ten years of genetic fuzzy systems : Current framework and new trends. In *IFSA/NAFIPS*, Vancouver.

Cortes, C. et Vapnik, V. N. (1995). Support-vector network. *Machine Learning*, 20 :273–297.

Craven, M. W. et Shavlik, J. W. (1994). Using sampling and queries to extract rules fom trained neural networks. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 37–45, San Mateo.

Daida, J. M., Bersano-Begey, T. F., Ross, S. J., et Vesecky, J. F. (1996). Evolving feature-extraction algorithms : Adapting genetic programming for image analysis in geoscience and remote sensing. In *Proceedings of the International Geoscience and Remote Sensing Symposium : Remote Sensing for a Sustainable Future*, Washington.

Daida, J. M., Lund, D. E., Wolf, C., Meadows, G. A., Schroeder, K., Vesecky, J. F., Lyzenga, D. R., et Bertram, R. (1995). Measuring topography of small-scale waves. In Stein, T., editor, *Proceedings of IEEE International Geoscience and Remote Sensing*, pages 1881–1883, Florence. IEEE Press.

DAISEX (2001). *Proceedings of the Final Results Workshop on DAISEX (Digital AIrborne Spectrometer EXperiment)*. ESTEC, Noordwijk.

Debeir, O., Latinne, P., et den Steen, I. V. (2001). Remote sensing classification of spectral, spatial and contextual data using multiple classifier systems. In *Proceedings of the 8th European Congress of Stereology and Image Analysis, ECS-IA'2001*, pages 584–589.

DeJong, K. A. (1975). *Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan.

DeJong, K. A. (1988). Using genetic algorithms to learn task programs : the Pitt Approach. *Machine Learning*, 2(2-3).

Dietterich, T. G., Hild, H., et Bakiri, G. (1990). A comparative study of ID3 and backpropagation for english text-to-speech mapping. In *In Proceedings of the 7th International Conference on Machine Learning*, pages 24–31, Austin.

Diverio, V. T., Gomez, A. T., Osório, F. S., Hoffmann, L. T., et Rodrigues, A. G. (2002). A neural approach to classification of satellite images. In *Proceedings of SIBGRAPI 2002*, page 413.

Eiben, A. E. et Schoenauer, M. (2002). Evolutionary computing. *Information Processing Letters*, 82(1) :1–6.

Eiben, A. E. et Smith, J. E. (2003). *Introduction to Evolutionary Computing*. Springer, ISBN 3-540-40184-9.

Fahlman, S. E. et Lebiere, C. (1990). The cascade-correlation learning architecture. Technical Report CMU-CS-90-100, School of Computer Science, Carnegie Mellon University, Pittsburgh.

FANN (2005). Fast Artificial Neural Network Library, disponible en ligne sur http://fann.sourceforge.net.

FoDoMuST (2005). FoDoMuST, un projet de fouille de données multi-stratégie réalisé dans le cadre de l'ACI *Masse de données*, document en ligne disponible sur http://lsiit.u-strasbg.fr/afd/fodomust.

Fodor, J. A. et Pylyshyn, Z. W. (1988). Connectionism and cognitive architecture : A critical analysis. *Cognition*, 28 :3–71.

Fong, C. K. et Yuen, S. Y. (2000). A genetic algorithm with coverage for object localization. In *Proceedings of the International Symposium on Intelligent Multimedia, Video & Speech Processing*, University of Hong Kong.

Fonlupt, C. et Robilliard, D. (2000). Genetic programming with dynamic fitness for a remote sensing application. In *Proceedings of Parallel Problem Solving from Nature (PPSN'2000)*, Paris.

Foody, G. M., Lucas, R. M., Curran, P. J., et Honzak, M. (1997). Non-linear mixture modelling without end-members using an artificial neural network. *International Journal of Remote Sensing*, 18(4) :937–953.

Freeman, J. J. (1998). A linear representation for GP using context free grammars. In *Genetic Programming 1998 : Proceedings of the Third Annual Conference*, pages 72–77, University of Wisconsin, Madison.

Furuhashi, T., Kakaoka, K., Morikawa, K., et Uchikawa, Y. (1993). Controlling excessive fuzziness in a fuzzy classifier system. In *Proceeedings of the Fourth International Conference on Genetic Algorithms*.

Gançarski, P. et Wemmert, C. (2005). Collaborative multi-strategy classification : Application to per-pixel analysis of images. In *Sixth International Workshop on Multimedia Data Mining (MDM/KDD2005)*, Chicago.

Ganascia, J. G. (1987). *AGAPE et CHARADE : deux techniques d'apprentissage symbolique appliquées à la construction de bases de connaissances*. PhD thesis, Université de Paris-Sud.

Gluck, M. A. et Thompson, R. F. (1987). Modeling the neural substrate of associative learning and memory : a computational approach. *Psychological Review*, 94 :176.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, Mass.

Goldberg, D. E. (1991). *Applications of Artificial Intelligence in Engineering VI*, chapter Genetic algorithms as a computational theory of conceptual design. Boston and New York : Computational Mechanics Publications and Elsevier Applied Science.

Goldberg, D. E. (1994). *Algorithmes génétiques, exploration, optimisation et apprentissage automatique*. Addison-Wesley, Paris.

Goldberg, D. E. et Deb, K. (1991). *Foundations of Genetic Algorithms*, chapter A comparative analysis of selection schemes used in genetic algorithms, pages 69–93.

Goldberg, D. E., Deb, K., et Clark, J. H. (1992). Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, 6(19) :333–362.

Goldberg, D. E., Korb, B., et Deb, K. (1989). Messy genetic algorithms : Motivation, analysis, and first results. *Complex Systems*, 3 :493–530.

Goldberg, D. E. et Sastry, K. (2001). A practical schema theorem for genetic algorithm design and tuning. In *Proceedings of the Third Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 328–335, San Francisco.

Gonzalez, A. et Perez, R. (1999). Slave : a genetic learning system based on an iterative approach. *IEEE Transactions on Fuzzy Systems*, 7(2) :176–191.

Graham, R. L. (1972). An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1(4) :132–133.

Greene, D. P. et Smith, S. F. (1994). Using coverage as a model building constraint in learning classifier systems. *Evolutionary Computation*, 2(1) :67–91.

Grzelak, O. (2001). Apprentissage supervise et extraction de connaissances à partir des réseaux de neurones de type hyperconvexe. Master's thesis, Université Louis Pasteur, Strasbourg.

Gualtieri, J. A. et Cromp, R. F. (1998). Support vector machines for hyperspectral remote sensing classification. In *Proceedings of the 27th AIPR Workshop : Advances in Computer Assisted Recognition, Vol. 3584*, pages 221–232, Washington.

Halkidi, M., Batistakis, Y., et Vazirgiannis, M. (2001). On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3) :107–145.

Harris, C. et Buxton, B. (1996). Evolving edge detectors with genetic programming. In Koza, J. R., Goldberg, D. E., Fogel, D. B., et Riolo, R. L., editors, *Genetic Programming 1996 : Proceedings of the First Annual Conference*, pages 309–315, Stanford University. MIT Press.

Harris, J. W. et Stocker, H. (1998). *Handbook of Mathematics and Computational Science*, chapter Maximum Likelihood Method. New York, Springer-Verlag.

Haykin, S. (1999). *Neural Networks*. Prentice Hall (seconde édition).

Hebb, D. O. (1949). *The Organization of Behaviour : A neuropsychological theory*. New York, Wiley.

Hjorth, J. S. U. (1994). *Computer Intensive Statistical Methods Validation, Model Selection, and Bootstrap*. London : Chapman & Hall.

Hoffmann, F. et Pfister, G. (1997). Evolutionary design of a fuzzy knowledge base for a mobile robot. *International Journal of Approximate Reasoning*, 17(4) :447–69.

Holland, J. H. (1971). Processing and processors for schemata. In *E. L. Jacks (Ed.), Associative information processing*, pages 127–146, New York : American Elsevier.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Cambridge : MIT Press.

Holland, J. H. (1985). Properties of the bucket brigade. In *Proceedings of the International Conference on Genetic Algorithms*, Hillsdale.

Holland, J. H. (1986). *Machine learning : An artificial intelligence approach, vol. 2*, chapter Escaping brittleness : The possibilities of general purpose learning algorithms applied to parallel rule-based systems. Morgan Kaufmann, San Mateo.

Holland, J. H., Holyoak, K. J., Nisbett, R. E., et Thagard, P. R. (1986). *Induction : Processes of Inference, Learning, and Discovery*. MIT Press.

Holland, J. H. et Reitman, J. S. (1978). *Pattern-directed inference systems*, chapter Cognitive systems based on adaptive algorithms. New York : Academic Press.

Holmes, J. H., Durbin, D. R., et Winston, F. K. (2000). A New Bootstrapping Method to Improve Classification Performance in Learning Classifier Systems. In *Proceedings of Parallel Problem Solving from Nature (PPSN VI)*.

Horn, J., Goldberg, D. E., et Deb, K. (1994). Implicit niching in a learning classifier system : Nature's way. *Evolutionary Computation*, 2(1) :37–66.

Hughes, G. F. (1968). On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory*, 14(1) :55–63.

Huguenin, R. L., Karaska, M. A., Blaricom, D. V., et Jensen, J. R. (1997). Subpixel classification of bald cypress and tupelo gum trees in thematic mapper imagery. *Photogrammetric Engineering and Remote Sensing*, 63(6) :717–725.

Ichku, C. et Karnieli, A. (1996). A review of mixture modelling techniques for sub-pixel land cover estimation. *Remote Sensing Reviews*, 13 :161–186.

ICU (2005). Discovery of classification rules : evolutionnary classifiers, logiciel disponible en ligne sur http://lsiit.u-strasbg.fr/afd/logiciels/icu.

Ishibuchi, H., Nakashima, T., et Murata, T. (1999). Performance evaluation of fuzzy classifier systems formultidimensional pattern classification problems. *IEEE Transactions on Systems, Man and Cybernetics - Part B : Cybernetics*, 29 :601–618.

Javed, F., Bryant, B., Crepinsek, M., Mernik, M., et Sprague, A. (2004). Context-free grammar induction using genetic programming. In *Proceedings of the 42nd Annual ACM Southeast Conference*, pages 404–405, Huntsville.

Jiang, X., Chen, M.-S., Manry, M. T., Dawson, M. S., et Fung, A. K. (1994). Analysis and optimization of neural networks for remote sensing. *Remote Sensing Reviews*, 9 :97–114.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43 :59–69.

Korczak, J. J. et Quirin, A. (2003a). Découverte de règles de classification par approche évolutive : application aux images de télédétection. In *Journées francophones d'Extraction et de Gestion de Connaissances (EGC2003)*, Lyon.

Korczak, J. J. et Quirin, A. (2003b). Evolutionary approach to discovery of classification rules from remote sensing images. In *5th European Workshop on Evolutionary Computation in Image Analysis and Signal Processing (EvoIASP2003)*, Essex.

Korczak, J. J. et Quirin, A. (2003c). Evolutionary mining for image classification rules. In *6th International Conference on Artificial Evolution (EA 2003)*, Marseille.

Koza, J. R. (1992). *Genetic Programming*. Cambridge : The MIT Press/Bradford Books.

Kriebel, K. T. et Koepke, P. (1987). Improvements in the shortwave cloud free radiation budget accuracy. Part II : Experimental study including mixed surfaces albedos. *Journal of Climate and Applied Meteorology*, 26 :396–409.

Landis, J. R. et Koch, G. G. (1977). The measurement of observer agreement for categorical data. *Biometrics*, 33(1) :159–174.

Lanzi, P. L. (1999a). Extending the representation of classifier conditions. Part I : From binary to messy coding. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*, pages 337–344.

Lanzi, P. L. (1999b). Extending the representation of classifier conditions. Part II : From messy coding to s-expressions. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*, pages pages 345–352.

Lennon, M., Mouchot, M. G., et Hubert-Moy, L. (2001). Spectral unmixing of hyperspectral images with the independent component analysis and wavelet packets. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS '01), Vol. 6*, pages 2896–2898.

Leone, A. et Menenti, M. (2000). Review of spectral features in the 400 - 2500 nm region, LSPIM EXP CCN-2 Final Presentation.

Liu, X., Chen, K., et Wang, D. L. (2001). Extraction of hydrographic regions from remote sensing images using an oscillator network with weight adaptation. *IEEE Transactions On Geoscience And Remote Sensing*, 39(1).

Liu, Z., Liu, A., Wang, C., et Niu, Z. (2004). Evolving neural network using real coded genetic algorithm (GA) for multispectral image classification. *The International Journal of Grid Computing : Theory, Methods and Applications*, 20 :1119–1129.

LIV (2005). Laboratoire Image et Ville (UMR 7011), http://imaville.u-strasbg.fr.

Llorà, X. et Goldberg, D. E. (2003). Bounding the effect of noise in multiobjective learning classifier systems. *Evolutionary Computation*, 11(3) :279–298.

Logo (2005). Logo Foundation, http://el.media.mit.edu/logo-foundation.

Long, E. R. D., Long, D. M. D., et Clarke-Pearson, D. L. (1988). Comparing the areas under two or more correlated receiver operating characteristic curves : a nonparametric approach. *Biometrics*, 44(3) :837–45.

Lucieer, A. et Kraak, M. (2004). Interactive and visual fuzzy classification of remotely sensed imagery for exploration of uncertainty. *International Journal of Geographical Information Science*, 18(5) :491–512.

Luo, J.-C., Zhou, C.-H., et Leung, Y. (2001). A knowledge-integrated RBF network for remote sensing classification. In *Proceedings of ACRS 2001 - 22nd Asian Conference on Remote Sensing, Vol. 1*, pages 181–186, Singapore.

Manry, M. T., Apollo, S. J., Allen, L. S., Lyle, W. D., Gong, W., Dawson, M. S., et Fung, A. K. (1994). Fast training of neural networks for remote sensing. *Remote Sensing Reviews*, 9 :77–96.

Marani, M., Belluco, E., Ferrari, S., Silvestri, S., D'Alpaos, A., Lanzoni, S., Feola, A., et Rinaldo, A. (2005). *Estuarine, Coastal and Shelf Science*, chapter Analysis, synthesis and modelling of high-resolution observations of saltmarsh ecogeomorphological patterns in the Venice lagoon (in press).

Marani, M., Lanzoni, S., Silvestri, S., et Rinaldo, A. (2004). Tidal landforms, patterns of halophytic vegetation and the fate of the lagoon of venice. *Journal of Marine Systems*, 51(1-4) :191–210.

Marani, M., Silvestri, S., Belluco, E., Camuffo, M., D'Alpaos, A., Defina, A., Lanzoni, S., Marani, A., Tortato, M., et Rinaldo, A. (2003). Patterns in tidal environments : salt-marsh channel networks and vegetation. In *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium*, Toulouse.

Marchand, Y. et Damper, R. I. (2000). A multi-strategy approach to improving pronunciation by analogy. *Computational Linguistics*, 26(2) :195–219.

MDD (2005). ACI *Masse de données*, document en ligne disponible sur http://acimd.labri.fr.

Meerkoetter, R. (1990). Reflection functions for inhomogeneous land surfaces. *Applied Optics*, 29 :4192–4198.

Metz, C. E. (1978). Basic principles of ROC analysis. *Seminars In Nuclear Medicine*, 8(4) :283–298.

Metzler, V., Palm, C., Lehmann, T., et Aach, T. (2000). Texture classification of graylevel images by multiscale cross-cooccurrence matrices. In *International Conference on Pattern Recognition (ICPR'00), Volume 2*.

Michalewicz, Z. (1996). *Genetic algorithms + datastructures = evolution programs*. Third, revised and extended edition, Springer Verlag, Berlin.

Michalski, R. S. (1983). A theory and methodology of inductive learning. *Machine Learning : An Artificial Intelligence Approach*, 1 :83–134.

Mitchell, M., Forrest, S., et Holland, J. H. (1991). The royal road for genetic algorithms : Fitness landscapes and GA performance. In *Toward a Practice of Autonomous Systems : First European Conference on Artificial Life*, pages 245–254, Cambridge.

Mitchell, T. (1997). *Machine Learning*. McGraw Hill.

Mohr, T. (1999). Répertoire CGMS des Applications Météorologiques Satellitales, EUMETSAT, document n°EUM BR 08, version en ligne sur http:\\www.eumetsat.int.

Montana, D. J. (1994). Strongly typed genetic programming. Technical Report #7866, Bolt Beranek and Newman, Inc., 10 Moulton Street, Cambridge.

Murtagh, F. et Starck, J.-L. (1999). *Encyclopedia of Microcomputers, A. Kent and J. G. Williams, Eds., Marcel Dekker*, chapter Multiscale Image and Data Analysis (submitted).

Naur, P. (1960). Revised report on the algorithmic language algol 60. In *Communications of the ACM, Vol. 3, No.5*, pages 299–314.

Nayak, R. (2000). *Gyan : A Methodology for Rule Extraction From Neural Networks*. PhD thesis, Queensland University of Technology.

Nayak, R., Hayward, R., et Diederich, J. (1997). Connectionist knowledge representation by generic rules extraction from trained feedforward neural networks. In *Proceeding of Connectionist Systems for Knowledge Representation and Deduction Workshop*, pages 87–98, Townsville.

Nielsen, A. A. (2001). Spectral mixture analysis : Linear and semi-parametric full and iterated partial unmixing in multi- and hyperspectral image data. *International Journal of Computer Vision*, 42(1-2) :613–619.

Novak, J. P. (2000). *Méthodes neuronales pour la segmentation d'images de télédétection, et l'apprentissage de concepts.* PhD thesis, Université Louis Pasteur, Strasbourg.

Osorio, F. S. (1998). *INSS : Un système hybride pour l'apprentissage automatique constructif.* PhD thesis, Laboratoire Leibniz, INPG, IMAG, Grenoble, v.1, 305p.

Ozcan, E. et Mohan, C. K. (1997). Partial shape matching using genetic algorithms. *Pattern Recognition Letters*, 18 :987–992.

Pal, M. et Mather, P. M. (2004). Assessment of the effectiveness of support vector machines for hyperspectral data. *Future Gener. Comput. Syst.*, 20(7) :1215–1225.

Palma, S. D., Rugna, J. D., et Zighed, D. A. (1997). Apprentissage supervisé polythétique : une évaluation. In *Cinquièmes Rencontres de la Société Francophone de Classification (SFC'97)*, Lyon.

Parodi, A. et Bonelli, P. (1993). A new approach to fuzzy classifier systems. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 223–230, San Mateo : Morgan Kaufmann.

Plutowski, M., Sakata, S., et White, H. (1994). *Advances in Neural Information Processing Systems 6*, chapter Cross-validation estimates IMSE, pages 391–398. San Francisco : Morgan Kaufmann.

Post, E. L. (1943). Formal reductions of the general combinatorial decision problem. *American Journal of Math*, 52 :264–268.

Powell, M. (1985). Radial basis functions for multivariable interpolation. In *IMA Conference on Algorithms for the Approximation of Functions and Data, RCMS Shrivenham*, pages 143–167.

Preparata, F. P. et Shamos, M. I. (1985). *Computational Geometry : An Introduction.* Springer-Verlag, New York.

Qin, W., Goel, S., et Wang, B. (1996). The hotspot effect in heterogeneous vegetation canopies and performances of various hotspot models. *Remote Sensing Reviews*, 14 :283–332.

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

Quinlan, J. R. (1993). *C4.5 : Programs for Machine Learning.* Los Altos : Morgan Kaufmann, 302p.

Quinlan, J. R. (1994). *Computational Learning Theory and Natural Learning Systems - Volume I : Constraints and Prospects*, chapter Comparing connectionist and symbolic learning methods, pages 445–456. MIT Press.

Quirin, A. (2002). Découverte de règles de classification par classifieurs évolutifs. Master's thesis, Université Louis Pasteur, Strasbourg.

Quirin, A. et Korczak, J. J. (2005a). *Genetic and Evolutionary Computation in Image Processing and Computer Vision*, chapter Discovering of Classification Rules from Hyperspectral Images. Springer-Verlag, LNCS series 2936.

Quirin, A. et Korczak, J. J. (2005b). Representation of genetic individuals for unmixing multispectral data. In *2005 IEEE Congress on Evolutionary Computation (CEC 2005)*, Edinburg.

Quirin, A., Korczak, J. J., Butz, M. V., et Goldberg, D. E. (2004). Learning classifier systems for hyperspectral images processing. Technical Report ULP-LSIIT-RR-2004-01, LSIIT, Université Louis Pasteur, Strasbourg.

Quirin, A., Korczak, J. J., Butz, M. V., et Goldberg, D. E. (2005). Analysis and evaluation of learning classifier systems applied to hyperspectral image classification. In *5th International Conference on Intelligent Systems Design and Applications (ISDA 2005)*, pages 280–285, Wroclaw.

Rakotomalala, R. (2005). TANAGRA : un logiciel gratuit pour l'enseignement et la recherche. In *EGC'2005, RNTI-E-3, vol. 2*, pages 697–702.

Rendon, M. V. (1997). Reinforcement learning in the fuzzy classifier system. Technical report, Institut Technologique des Etudes Supérieures, Monterrey.

Richards, R. A. (2001). *Classifier Systems & Genetic Algorithms*, chapter Zeroth-Order Shape Optimization utilizing a Learning Classifier System, document en ligne disponible sur http://www.stanford.edu/~buc/SPHINcsX/book.html.

Ricotta, C., Avena, G., et Palma, A. D. (1999). Mapping and monitoring net primary productivity with AVHRR NDVI time-series : statistical equivalence of cumulative vegetation indices. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54(5) :325–331.

Riolo, R. L. (1988). *Empirical Studies of Default Hierarchies and Sequences of Rules in Learning Classifier Systems*. PhD thesis, Computer Science and Engineering Department, University of Michigan.

Robertson, G. G. et Riolo, R. L. (1988). A tale of two classifier systems. *Machine Learning*, 3 :139–159.

Robilliard, D. et Fonlupt, C. (2001). Backwarding : An overfitting control for genetic programming in a remote sensing application. In *Proceedings of Artificial Evolution, LNCS 2310*, pages 245–254.

Rosenblatt, F. (1962). *Principles of Neurodynamics : Perceptron and the Theory of Brain Mechanisms*. Washington : Spartan Books.

Ross, B. J., Gualtieri, A. G., Fueten, F., et Budkewitsch, P. (2002). Hyperspectral image analysis using genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002)*, pages 1196–1203.

Rumelhart, D. E., Hinton, G. E., et Williams, R. J. (1986). *Parallel Distributed Processing : Explorations in the Microstructure of Cognition - Volume 1 : Foundations*, chapter Learning Internal Representations by Error Propagation. Bradford Book, The MIT Press.

Ruttkay, Z., Eiben, A. E., et Raué, P.-E. (1995). Improving the performance of genetic algorithms on GA-hard constraint satisfaction problems. In *Workshop on Solving Hard CSP Problems, Constraint Programming'95 Conference*, pages 157–171, Cassis.

Samadzadegan, F., Azizi, A., et Abootalebi, A. (2000). Automatic determination of the optimum generic sensor model based on genetic algorithm concepts. *Journal Of The American Society For Photogrammetry And Remote Sensing*, 71(3) :277–288.

Schaeffer, J. et Schuurmans, D. (1989). Representational difficulties with classifier systems. In *International Conference of Genetic Algorithms*, pages 328–333.

Schmitt, L. M. (2004). Theory of genetic algorithms II : models for genetic operators over the string-tensor representation of populations and convergence to global optima for arbitrary fitness function under scaling. *Theoretical Computer Science*, 310 :181–231.

Schoenauer, M. et Michalewicz, Z. (1997). Evolutionary computation, control and cybernetics. *Special Issue on Evolutionary Computation*, 26(3) :307–338.

Setiono, R. (2000). Extracting M-of-N rules from trained neural networks. *IEEE Transactions on Neural Networks*, 11(2) :512–519.

Setiono, R. et Liu, H. (1996). Symbolic representation of neural networks. *IEEE Computer*, 29(3) :71–77.

Shao, J. et Tu, D. (1995). *The Jackknife and Bootstrap*. New York, Springer-Verlag.

Sharon, E., Brandt, A., et Basri, R. (2000). Fast multiscale image segmentation. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 70–77, South Carolina.

Shavlik, J. W., Mooney, R. J., et Towell, G. G. (1991). Symbolic and neural learning algorithms : An experimental comparison. *Machine Learning*, 6(2) :111–143.

Sigaud, O. (2002). *Automatisme et subjectivité : l'anticipation au coeur de l'expérience*. PhD thesis, Université Paris I.

Silvestri, S., Defina, A., et Marani, M. (2005). Tidal regime, salinity and salt-marsh plant zonation. *Estuarine Coastal and Shelf Science*, 62 :119–130.

Silvestri, S. et Marani, M. (2004). *The Ecogeomorphology of Tidal Marshes. Coastal and Estuarine Studies, 59*, chapter Salt marsh vegetation and morphology, modelling and remote sensing observations, page 266. Hardbound, ISBN 0-87590-273-1.

Silvestri, S., Marani, M., et Marani, A. (2003). Hyperspectral remote sensing of salt marsh vegetation, morphology and soil topography. *Physics and Chemistry of the Earth*, 28(1-3) :15–25.

Silvestri, S., Marani, M., Settle, J., Benvenuto, F., et Marani, A. (2002). Salt marsh vegetation radiometry. data analysis and scaling. *Remote Sensing of Environment*, 80 :473–482.

Singh, S. (1998). Effect of noise on generalisation in massively parallel fuzzy systems. *Pattern Recognition*, 31(11) :25–33.

Sordo, M. (1997). Kbann : Learning from small data sets. Technical Report 9701, School of Cognitive and Computing Sciences, University of Sussex.

Sousa, S., Caeiro, S., Jr, R. G. P., et Painho, M. (2003). Sado estuary management areas : hard versus soft classification maps comparison. In *Conference proceedings of CoastalGIS 2003, Fifth International Symposium on GIS and Computer Cartography for Coastal Zone Management*, Genova.

SPOT (2005a). Informations sur le programme SPOT, document en ligne disponible sur http://www.ac-creteil.fr/svt/Teledec/Ima_spot/ima_spot.htm.

SPOT (2005b). Informations sur le programme SPOT, document en ligne disponible sur http://www.spotimage.fr/accueil/system/introsat/payload/welcome.htm.

SPOT (2005c). Site de SPOT Image, http://www.spotimage.fr.

Studley, M. et Bull, L. (2005). *Artificial Life*, chapter Using the XCS Classifier System for Multi-objective Reinforcement Learning Problems (in press).

Sutton, R. (1991). Reinforcement learning architectures for animats. In *From animals to animats : Proceedings of the First International Conference on Simulation of Adaptative Behavior*, Cambridge : MIT Press.

Thrun, S. (1995). Extracting rules from artificial neural networks with distributed representations. In Tesauro, G., Touretzky, D., et Leen, T., editors, *Advances in Neural Information Processing Systems (NIPS) 7*, Cambridge. MIT Press.

Thrun, S. B. (1993). Extracting provably correct rules from artificial neural networks. Technical Report IAI-TR-93-5, Institut fur Informatik III, Bonn.

TIDE (2005). Tidal Inlets Dynamics and Environment, Research Project Supported by the European Commission under the Fifth Framework Programme, contract n° EVK3-CT-2001-00064, document en ligne disponible sur http://www.istitutoveneto.it/tide.

Towell, G. G. et Shavlik, J. W. (1993). Extracting refined rules from knowledge-based neural networks. *Machine Learning*, 13 :71–101.

Valenzuela-Rendón, M. (1991). The fuzzy classifier system : a classifier system for continuously varying variables. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 346–353, San Mateo : Morgan Kaufmann.

Valenzuela-Rendón, M. (1998). Reinforcement learning in the fuzzy classifier system. *Expert Systems with Applications*, 14(1-2) :237–247.

Valigiani, G., Fonlupt, C., et Collet, P. (2004). Analysis of GP improvement techniques over the real-world inverse problem of ocean colour. In *EUROGP'04*, Coimbra.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory.* Berlin : Springer-Verlag.

Velasco, J. R. (1998). Genetic-based on-line learning for fuzzy process control. *International Journal of Intelligent Systems*, 13(10-11) :891–903.

Velez, M. (2003). Unsupervised feature subset selection using matrix factorization methods with application to hyperspectral imagery band subset selection. In *CISE Lecture II.*

Venturini, G. (1993). SIA : a supervised inductive algorithm with genetic search for learning attribute based concepts. In *Proceedings of the European Conference on Machine Learning*, pages 280–296, Viena.

Verbyla, D. et Hammond, T. (2002). How To Lie With An Error Matrix, Remote Sensing & Image analysis, document en ligne disponible sur http://nrm.salrm.uaf.edu/~dverbyla/online/errormatrix.html.

Vose, M. D. (1999). *The Simple Genetic Algorithm : Foundations and Theory.* MIT Press, Cambridge.

Whitley, D. (1993). *Foundations of Genetic Algorithms II*, chapter An Executable Model of a Simple Genetic Algorithm, pages 45–62. Morgan Kaufmann, San Mateo.

Widrow, B. et Hoff, M. E. (1960). Adaptive switching circuits. In *IRE Western Electric Show and Convention Record, Part 4*, pages 96–104.

Williams, R. J. et Zipser, D. (1989). *A Learning Algorithm for Continually Running Fully Recurrent Neural Networks*. Neural Computation, 1, pages 270-280.

Wilson, S. W. (1994). ZCS : A zeroth level classifier system. *Evolutionary Computation*, 2(1) :1–18.

Wilson, S. W. (1995). Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2) :149–175.

Wilson, S. W. (1998). Generalization in the XCS classifier system. In *Proceedings of the Third Annual Genetic Programming Conference*, pages 665–674, Madison (WI), Morgan Kaufmann, San Francisco.

Wilson, S. W. (2000a). *Learning Classifier Systems : From Foundations to Applications, LNAI 1813*, chapter Get real ! XCS with continuous-valued inputs, pages 209–220.

Wilson, S. W. (2000b). Mining oblique data with XCS. *Lecture Notes in Computer Science*, 1996 :158–176.

Wilson, S. W. (2000c). Mining oblique data with XCS. Technical Report 2000028, Department of General Engineering The University of Illinois, Urbana-Champaign.

Wilson, S. W. et Goldberg, D. E. (1989). A critical review of classifier systems. In *Proceedings of 3th ICGA*.

Xie, X. et Beni, G. (1991). Validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8) :841–847.

Yip, C. L. et Wong, K. Y. (2004). Efficient and effective tropical cyclone eye fix using genetic algorithms. In *Proceedings of Knowledge-Based Intelligent Information and Engineering Systems : 8th International Conference (KES 2004)*, Wellington.

Yuhas, R. H., Goetz, A. F. H., et Boardman, J. W. (1992). Discrimination among semiarid landscape endmembers using the spectral angle mapper (SAM) algorithm. In *Summaries of the Third Annual JPL Airborne Geoscience Workshop*, pages 147–149, Pasadena, Jet Propulsion Laboratory.

Zhou, Q. (2003). *Adaptive Knowledge Discovery Techniques for Data Mining*. PhD thesis, University of Otago, New Zealand.

Zighed, D. A., Auray, J. P., et Duru, G. (1992). *SIPINA, Méthode et Logiciel*. Edition Lacassagne, 190p.

Résumé

En classification supervisée d'images de télédétection, la découverte de classes précises et exactes et leur explication, comptent parmi les objectifs essentiels que se fixent les thématiciens. Les images de télédétection haute résolution et hyperspectrales contiennent des données volumineuses et complexes. De la part d'un algorithme de classification, répondre à cette requête est une mission relativement ardue. En effet, de nombreuses sources d'informations de différentes natures et souvent très complexes sont disponibles, à partir desquelles les concepts thématiques étudiés par l'expert doivent être extraits et surtout expliqués. Dans de nombreux cas, il peut s'avérer nécessaire de considérer – en plus du problème habituel de classification – l'idée qu'un même pixel peut appartenir à plusieurs classes. La technologie l'autorisant de plus en plus, seules des images de très haute résolution (< 60 cm) peuvent être exploitées pour obtenir une précision acceptable.

Les travaux de cette thèse, consacrés à la découverte de règles de classification, se découpent en trois parties : (1) explorer l'influence de la représentation des classifieurs sur la qualité de reconnaissance des différentes classes de terrain en classification *hard* ou *soft*, (2) étudier plusieurs post-traitements des bases de règles produites par les algorithmes afin d'en améliorer ou d'en simplifier le contenu et (3) modifier des représentations existantes ou utiliser de nouveaux paradigmes pour traiter les classifications floues.

Pour faire face à la complexité et la taille des données, il est connu dans la littérature que les algorithmes évolutifs présentent une approche idéale pour ce genre de problème. Nous avons notamment retenu une approche basée sur un système de classifieurs couplé à un algorithme génétique. Ces systèmes permettent de développer des populations de règles de classification simples, lisibles et génériques. Ces bases de règles favorisent l'application de la connaissance apprise sur une autre partie de l'image voire une nouvelle image tout en garantissant un taux correct de vrais positifs et vrais négatifs pour la classification de nouveaux exemples. Le couplage avec un algorithme génétique permet une découverte de solutions de manière évolutive, ce qui ce qui facilite l'absorption d'une grande partie de la complexité de traitement d'une telle masse de données et surtout d'être tolérant au bruit (robustesse).

Enfin, un certain nombre de mesures de qualité ont été développées pour juger de l'efficacité de ces algorithmes et des protocoles de validation ont été proposés pour comparer les différents résultats entre eux. Ces algorithmes ont tous été validés sur des données réelles de télédétection, dans le cadre du projet européen TIDE et du projet d'ACI FoDoMust. Nous pensons que les résultats obtenus par les différentes expérimentations et par les validations, présentés dans la thèse, sont encourageants. Ces recherches ont abouti à la conception et au développement de plusieurs logiciels fonctionnels disponibles sur le site web : http://lsiit.u-strasbg.fr/afd.

Mots clés : intelligence artificielle, apprentissage supervisé, découverte de connaissances, découverte de règles de classification, évolution artificielle, systèmes de classifieurs, images de télédétection, images hyperspectrales.