Laboratoire des Sciences
de l'Image, de l'Informatique
et de la Télédétection
UMR-7005 CNRS

RAPPORT DE STAGE DE
DEA D'INFORMATIQUE

Découverte de règles de classification : classifieurs évolutifs

par

Arnaud QUIRIN

sous la direction

de Jerzy J. KORCZAK
et de Pierre GANÇARSKI

Illkirch, Juillet 2002

REMERCIEMENTS

Tout d'abord, un grand merci au professeur Jerzy Korczak, qui m'a accueilli aimablement dans l'équipe qu'il dirigeait. Il m'a fait confiance tout au long de mon stage, tout en m'apportant ses précieux conseils aussi bien pour la partie technique que théorique de mon projet.

Pierre Gançarski en a fait de même. Ses corrections pertinentes et pleines de bon sens ont su remettre en place certaines idées de mon rapport, en me laissant toutefois le dernier mot, sans tailler sauvagement dans le texte.

Ces compliments doivent aussi être partagés avec Cédric Wemmert qui, par ses éclaircissements, m'a permis d'avancer dans le traitement de certaines images volumineuses (n'oublions pas de commuter les bits premiers et de permuter le monoïde).

Je témoigne aussi de la gratitude à Anne Puissant, pour ses classifications, et à Christiane Weber pour la superbe lettre d'appréciation qu'elle m'a écrite.

Enfin merci et courage à Christian Michel, mon rapporteur, ainsi qu'à tous ceux qui s'intéressent à mes travaux alors qu'ils ne constituent pas forcément leurs spécialités.

Une dernière dédicace à mes grands-parents qui m'ont procuré une chose aussi bête qu'un toit et donc sans qui ce rapport n'aurait peut être jamais existé.

Pour finir, je souligne l'agréable ambiance de travail et de bonne humeur, forgée par toute l'équipe de stagiaires, de thésards et de permanents, à travers laquelle même les disques durs enrayés, les alimentations rétives, les virus espiègles et le réseau claudiquant n'ont jamais pu s'interposer. Quoique, …

SOMMAIRE

LISTE DES FIGURES

LISTE DES ALGORITHMES

LISTE DES TABLEAUX

1. INTRODUCTION

Préliminaires ...

Notre objectif n'est pas de retracer ici tout l'intérêt de l'intelligence artificielle. En effet, celle-ci a déjà prouvé son efficacité lorsque l'on souhaite traiter des problèmes faisant intervenir la capacité d'évaluation, de reconnaissance et de traitement de l'information (regroupement, classification) par l'ordinateur.

Ces aspects nécessitent bien souvent l'emploi d'algorithmes évolués et complexes utilisés dans de nombreux domaines, en particulier le commerce, la production industrielle, l'ingénierie. L'un des domaines qui à mon avis nécessite l'emploi de cette technique est bien la télédétection. Un domaine où l'apprentissage et l'extraction automatique de connaissances se révèle indispensable, compte tenu des innombrables applications que l'on en fait (cartographie, évaluation et reconnaissance des caractéristiques du terrain, catalogue de ressources, gestion d'environnement naturel, aménagement de territoire urbain). Un domaine aussi très fastidieux à traiter, vu la quantité quasi invraisemblable de données différentes à manipuler, données très volumineuses dûes à l'amélioration constante des techniques de prises de vues et de la sensibilité des capteurs, que ce soient ceux de satellites ou d'avions haute altitude. Un domaine qui nécessite aussi une interaction simple et efficace entre l'homme et l'ordinateur, afin de réaliser un travail efficace, même et surtout par des personnes non expertes en informatique, ce qui reste le cas en télédétection.

Nos objectifs

Nous nous intéressons à la classification d'images, c'est-à-dire à l'extraction de concepts thématiques intéressants (immeubles, routes, ...) et à leur localisation sur des images brutes, produites notamment par des satellites. Pour cela, nous étudierons principalement deux types d'images, les images haute résolution et les images hyperspectrales, particulièrement difficiles à traiter à la main ou par des méthodes déterministes. En effet, ce type d'images contient beaucoup d'informations (quelques méga-octets), parfois erronées (due à l'interaction avec l'atmosphère dans le cas d'images satellites), parfois mélangées ou brouillées (on parle de pixels mixtes). Il serait particulièrement intéressant de pouvoir disposer d'une méthode efficace qui puisse s'adapter à ces différents aspects.

A l'inverse des algorithmes de classification traditionnels, produisant une image classifiée sans fournir les détails du processus de production à l'utilisateur, nous souhaitons pouvoir générer ce que l'on nomme des *règles de classification*, permettant de synthétiser la connaissance acquise durant la reconnaissance et de pouvoir la reproduire éventuellement sur une image nouvelle. C'est un problème nouveau, difficile, qui n'a pas encore été abordé dans la littérature. Ces règles ou *classifieurs* sont plus que la traduction littérale de cette connaissance sous forme d'une expression symbolique de la forme : ***si*** *conjonctions de conditions* ***alors*** *classe*. Elles permettent à l'utilisateur de comprendre de manière simple comment l'algorithme est capable de discriminer deux classes distinctes.

Afin de concrétiser ces objectifs, nous avons étudié plusieurs aspects de la question : nous avons conçu la méthode de représentation de ces règles ainsi que l'algorithme et les opérateurs génétiques pour les brasser. Enfin un logiciel d'expérimentation a été développé pour valider nos solutions sur des cas concrets d'images de télédétection. Cette étude s'accompagne en outre d'une évaluation de la qualité du système de classifieurs proposé, notamment leur faculté d'adaptation

dynamique à la variété des problèmes présentés, ainsi que leur disposition à explorer correctement et de manière homogène l'espace des solutions.

Le contexte

Dans l'équipe AFD (Apprentissage et Fouille de Données) du LSIIT (Laboratoire des Sciences de l'Image, de l'Informatique et de la Télédétection, UMR 7005 CNRS) plusieurs méthodes d'extraction de concepts et de classification d'images ont déjà été développées et validées ([NET 4]). L'objectif du stage est de proposer une nouvelle méthode de classification - robuste et évolutive - qui, contrairement aux algorithmes de classification habituels, permettra de générer de manière supervisée des classifieurs autonomes. Notre position se trouve donc être en marge des systèmes courants puisque nos classifieurs s'adapteront aux données fournies et affineront leur règles de classification en fonction de l'évolution des classes et de leurs interactions. Ces classifieurs, créés par un algorithme génétique, seront initialisés selon des connaissances déjà collectées auprès d'experts (d'où la notion de méthode supervisée), dans le domaine concerné par les images étudiées, essentiellement des images SPOT de Strasbourg et des environs.

Ce projet s'inscrit ainsi dans le cadre d'une collaboration avec le laboratoire LIV (Image et Ville, Strasbourg, UMR 7011 CNRS, [NET 9]) et constitue l'un des thèmes de recherche scientifique sur les images de télédétection chers au **Pôle Image** de Strasbourg (mis en place par un contrat de Plan Etat - Région couvrant la période 2000-2006).

De plus, nos recherches intéressent très fortement le projet européen de recherche et développement technique **TIDE** (*Tidal Inlets Dynamics And Environment*), plus exactement dans le groupe de travail 6 (ang. *Workpackage 6*) : *Accurate Vegetation Classification from Remote Sensing Data* ([NET 10]). Le but du projet est de développer des modèles dynamiques permettant d'incorporer des environnements complexes, délicats et évoluant dans le temps comme les lagunes, les estuaires, ... L'un des sites étudiés se situe dans la région de Venise (San Felice), couvert par des satellites renvoyant des images assez lourdes en mémoire (voir le chapitre 3). Par leur adaptation à chaque environnement, nos classifieurs s'accommodent sans difficultés à ce type d'image, comme on peut le constater dans le chapitre 5.

Plan du rapport

Ce rapport est découpé en quatre parties principales. Nos travaux s'appuyant sur des méthodes supervisées, la première partie (chapitre 2) en trace une brève récapitulation. Nous y décrivons aussi sous leurs formes classiques quelques méthodes d'apprentissage dites par renforcement, principalement les systèmes de classifieurs.

La deuxième partie (chapitre 3) permet de comprendre *pourquoi* et *comment* nous avons spécialisé et amélioré ces systèmes de base pour les appliquer à la télédétection. Après un aperçu des données utilisées dans ce domaine, nous verront *pourquoi* nous étions obligés de proposer notre propre système et *comment* l'avons nous conçu.

Nous nous attarderons dans le chapitre suivant (chapitre 4) sur une caractéristique importante des systèmes de classifieurs : son évolutivité. Cette partie s'attachera à démonter son mécanisme et définira toute une série d'opérateurs adaptés au problème posé.

Nous exposerons dans une quatrième partie (chapitre 5) le prototype qui a autorisé l'étude de nombreux cas réels. Au fil de ces différents exemples, nous étudierons la robustesse et la performance de nos classifieurs.

Enfin nous conclurons dans le dernier chapitre et nous proposerons des perspectives de développement de ces travaux.

2. ETAT DE L'ART

2.1. Classification d'images de télédétection

Les classifieurs sont d'une manière générale, des algorithmes permettant de créer des *classes thématiques* à partir d'une image ou de toute autre donnée à analyser comprenant des informations spectrales. Ce partitionnement de l'image est appelé *zonage* par les experts en urbanisme, ou *classification* par les informaticiens. Il en existe plusieurs approches dont les principales sont :

- l'approche par pixel : chaque pixel est analysé indépendamment des autres en fonction de ses caractéristiques multispectrales (par exemple on réunit les régions de pixels qui ont sensiblement les *mêmes* valeurs radiométriques dans l'image à traiter),
- l'approche par zone : ici le partitionnement est sensiblement plus "intelligent" : on détecte les contours des régions, on les délimite par leur *texture* (relation de voisinage examinant la variabilité interne des objets constituant un paysage), les motifs répétitifs qu'ils présentent, ...
- l'approche par objet : c'est un haut niveau de reconnaissance : on détecte les formes des différentes régions en introduisant des connaissances expertes préalables (analyse morphologique et sémantique), par exemple pour les bâtiments, les terrains de sport, ...

Nous avons besoin de plusieurs notions inhérentes à celle de classification :

- on appelle *classe* une zone de pixels ayant des caractéristiques communes. On distingue les classes thématiques dont la représentation abstraite est appelée *concept*, qui dépendent du domaine d'application (par exemple, végétation ou bâtis) des classes spectrales qui ne sont que le regroupement de pixels de valeurs semblables.
- les *critères de ressemblance* qui permettent de comparer deux pixels entre eux. On utilise dans les algorithmes statistiques le plus souvent la distance euclidienne, mais il y en a d'autres : la distance ultramétrique s'exprime pour trois points x, y et z par :

$$d_{ultra}(x,z) \leq MAX(d_{ultra}(x,y), d_{ultra}(y,z)) \qquad \text{(Eq. 2.1)}$$

- le fait qu'un pixel appartienne à telle classe ou à telle autre est calculé par une *fonction discriminante*, qui choisira de l'*appartenance* de ce pixel à une classe.
- la *qualité* ou la *performance* d'une classification indique sa disposition à classifier correctement les pixels, et se mesure selon diverses méthodes (matrice de confusion).

De nombreuses méthodes de classification existent, mais chacune présentent des défauts et des qualités que les autres n'ont pas. Nous allons en voir quelques unes, mais nous restreindre à celles utilisables en télédétection. De plus, ce chapitre doit nous permettre d'avoir une vue globale sur les méthodes existantes d'apprentissage et de découverte de classes en télédétection.

Ces classifieurs peuvent demander ou non un paramétrage ou une initialisation par un expert ou par d'autres classifications pré-établies. On peut donc distinguer plusieurs types d'apprentissage, selon qu'ils soient supervisés, non supervisés ou par renforcement.

2.2. Les méthodes supervisées

Ce sont des méthodes nécessitant l'appui d'un expert ou d'autres classifications existantes. Pour classifier une zone, elles nécessitent une connaissance préalable de cette zone. Souvent, avant la classification proprement dite, on précise les classes à trouver et leurs caractéristiques spectrales, puis leur définition passe par un entraînement sur des échantillons représentatifs des classes à trouver, connu sous le terme d'*apprentissage*.

Une classification supervisée repose généralement sur une procédure semi-automatique qui inclut des connaissances *a priori* sur les objets à segmenter (taille, orientation, radiométrie moyenne, etc.). Ces méthodes permettent de découvrir les irrégularités des exemples d'apprentissage et à exploiter les ressemblances entre ces exemples pour construire des classes.

2.2.1. Les méthodes adaptées aux réseaux de neurones

Etudions quelques exemples d'apprentissage supervisé par réseaux de neurones. Un réseau neuronal est un ensemble de cellules de calcul appelées *neurones*, connectés par des liens nommés *connexions* portant chacune un *poids*, et prenant des données en entrées et renvoyant d'autres données en sortie [Rosenblatt, 1962]. Ils font partie de la grande famille des mémoires associatives ([Williams, 1989]), c'est-à-dire qu'ils peuvent apprendre à associer certains concepts à d'autres. Souvent 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 voit que pour apprendre cette association, il faut entraîner ces réseaux au préalable sur des exemples choisis, d'où l'appellation d'apprentissage supervisé. On donne donc à ces algorithmes un ensemble de paires de valeurs (entrée et sortie attendue), et l'apprentissage trouve une fonction déterministe qui mappe les entrées sur les sorties, en essayant de minimiser les conflits qu'il peut y avoir sur de nouvelles présentations de paires. Voici quelques exemples :

- **Les Perceptrons MultiCouches** (**PMC**) [Rumelhart, 1986] qui permettent de séparer les données par des hyperplans. Dans ce cas, le réseau détermine la classe d'un objet présenté en entrée selon qu'il appartienne à la région supérieure ou inférieure situées de part et d'autre de l'hyperplan. Plusieurs algorithmes d'apprentissage ont été développé pour l'ajustement des poids des connexions. Ils utilisent chacun l'une ou l'autre des règles suivantes :

    - La règle de Hebb [Hebb, 1949] basée sur l'interprétation suivante : si deux neurones de chaque côté d'une synapse sont activés simultanément, le poids de la synapse va augmenter. La règle est la suivante :

$$w_{ij}^{new} = w_{ij}^{old} + e_i s_j \qquad \text{(Eq. 2.2)}$$

      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_j est la valeur de l'entrée du neurone et s_i la valeur de sa sortie.

- De nombreuses variantes existent, par exemple la règle de Hebb avec taux d'apprentissage, avec terme d'oubli [Gluc, 1987] ou la règle de Widrow-Hoff, [Widrow, 1960]). Ces règles ajoutent des termes supplémentaires qui adaptent les poids des connexions de manière à avoir 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 [Rumelhart, 1986], décrit aussi par [Fodor, 1988]. A partir d'une base d'exemples constitués chacun d'une paire entrée et sortie désirée que l'on présentent successivement au réseau, l'algorithme calcule l'erreur commise par le réseau en comparant l'erreur en sortie commise par le réseau à partir de l'exemple donné et la sortie attendue. 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 sorties vers les neurones d'entrées) l'erreur calculée sur les neurones de sorties.

- **Les réseaux HyperConvexes**, introduits par [Novak, 2000] qui définissent un nouveau type de neurone utilisant le principe des *polygones de Thiessen* ([Blekas, 1997]), pour délimiter l'espace en régions géométriques rectangulaires ou sphériques. Cette famille de réseaux a été améliorée 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.

- **Les réseaux RBF** (**Radial Basis Function**) ([Powell, 1985], [Boser, 1992]) 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 les pixels adjacents à un pixel donné doivent jouer un rôle important durant l'apprentissage, tandis que les pixels trop éloignés ne doivent pas intervenir. Une telle méthode d'apprentissage est développée dans [Blanzieri, 1998].

Citons enfin des méthodes d'extraction de règles, dites de production, à partir de réseaux neuronaux découverts par ces algorithmes, par exemple la méthode VIA de [Thrun, 1993] ou de [Fu, 2000] dans le cadre des réseaux RBF, qui ont inspiré les travaux de [Grzelak, 2001]. En effet, il se trouve que la découverte de règles est très utile dans le cas des réseaux neuronaux, car elle résoud l'un de leur plus grand défaut : la difficulté de leur interprétation.

2.2.2. Graphes d'induction

Ces méthodes permettent de produire à partir d'un ensemble d'exemples classifiés, un arbre ou un graphe d'induction permettant de relier un attribut 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 dûe à leurs nombreuses qualités :

- Elles permettent de générer un ensemble minimal de règles de décision.

- Elles 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.
- Elles 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 attributs explicatifs sont souvent appelés variables exogènes, tandis que ceux à prédire sont dénommés variables endogènes. 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 une 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. Voici comme illustration, l'algorithme ID3, qui nécessite au préalable quelques définitions :

- Soit $P=(p_1, ..., p_n)$ une distribution de probabilité des classes, alors l'entropie $I(P)$ se calcule par :

$$I(P) = -\sum_{i=1}^{n} p_n \cdot \log(p_n) \qquad \text{(Eq. 2.3)}$$

- Si nous divisons la 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 :

$$Info(X,T) = \sum_{i=1}^{n} \frac{Card(T_i) \cdot I(T_i)}{Card(T)} \qquad \text{(Eq. 2.4)}$$

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

$$Gain(X,T) = I(T) - Info(X,T) \qquad \text{(Eq. 2.5)}$$

ALGORITHME 1
Création d'un arbre de décision par ID3
FONCTION : ID3(R,C,S)
RETOURNE : T, un arbre de décision

Variables :
R est un ensemble de variables exogènes
C est la variable endogène (à prédire) de valeurs c_1, ..., c_n
S est l'ensemble d'exemples
Fonctions :
FAIRE_NOEUD(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.
COMPTE(S,C,v) renvoie le nombre d'enregistrements de S ayant C= v
ENRACINE(T,N) enracine l'arbre N à la racine de T

```
si S = { }
      retourner ERREUR
fin si
si COMPTE(S,C,v) = Card(S)          // Si C a la même valeur v dans S
      T := FAIRE_NOEUD(TRUE,v)
      retourner T
fin si
```

```
si R = { }
      trouver    v tel que COMPTE(S,C,v) ≥ COMPTE(S,C,v') pour toute valeur v' possible
                 pour C
      T := FAIRE_NOEUD(TRUE,v)
      retourner T
fin si
trouver D tel que Gain(D,S) ≥ Gain(D',S) pour toute variable D de R
trouver { d_i | i=1, ..., n } l'ensemble des valeurs de la variable D
trouver { S_i | i=1, ..., n } l'ensemble des enregistrements de S dont la variable D
        vaut d_i
T := { }
pour i de 1 à n
      N := FAIRE_NOEUD(D=d_i , ID3(R-{ D} , C, S_i))
      T := ENRACINE(T,N)
fin pour
retourner T
```

Cet algorithme renvoie un arbre dans lequel chaque nœud permet de partager l'ensemble d'exemples selon le critère qui semble le plus discriminant pour l'algorithme. L'utilisateur n'a 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.

L'algorithme C4.5 est une version améliorée de l'algorithme précédent. Il passe par les deux phases suivantes : une phase d'expansion, utilisant la fonction d'entropie et dont le but est de diviser l'ensemble d'apprentissage. Puis une phase d'élagage emploie une fonction heuristique permettant d'estimer, à l'aide de l'ensemble d'apprentissage, l'erreur réelle d'un sous-arbre donné qui est alors remplacé par une feuille (simplification de l'arbre de départ).

Enfin, citons la méthode SIPINA ([NET 11]) 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 sommets). 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 sommets relativement restreint.

2.2.3. Systèmes inductifs

De nombreux domaines d'étude génèrent souvent des masses d'informations gigantesques qui sont ardues à être disséquées, 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, 1999]) comme l'algorithme AQR ne générant qu'une seule règle de classification par classe à décrire (cette méthode est célèbre depuis qu'elle a été testée avec succès sur des plans de soja), ou CN2 combinant les avantages d'AQR et d'ID3 ([Clark, 1989], [NET 15]).

Les algorithmes correspondants sont souvent utilisés pour extraire des connaissances à partir d'une base de données. Pour satisfaire les deux critères présentés plus haut, ils utilisent des critères 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, 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 (Missing Value Imputation ou MVI, [Cao, 2001]).

2.2.4. La recherche de règles d'association

Ce procédé est apparu en 1993, à l'initiative de chercheurs en base de données travaillant pour le compte du Centre de Recherche d'IBM en Californie ([Agrawal, 1993]). Bien qu'elles puissent servir à l'analyse de documents loués (cassettes vidéos, livres, ...) ou au dépouillement de questionnaires, la première application de ces recherches était de découvrir, à des fins de marketing, des associations entre les produits achetés par les clients d'une chaîne de supermarchés, c'est-à-dire de déduire les familles de produits qui pouvaient être achetés ensemble. Ces méthodes ont été étudiées pour être applicables sur des entrepôts de données (ang. *Data Warehouse*) de plusieurs millions d'entrées. [Srikant, 1996] a étendu leurs possibilités en permettant l'analyse de données temporelles, à condition que l'on puisse garder la trace des clients : il s'agit par exemple de déterminer si deux produits sont systématiquement vendus à quelques jours d'intervalle.

Le principe est de maximiser deux quantités S et C, appelées respectivement *support* et *confiance* et définies uniquement à partir d'un ensemble de données, par exemple la base de *transactions* (opérations de vente) de supermarchés. Soit t une transaction d'un ensemble T et soit X et Y deux ensembles d'items (articles vendus). On définit :

$$S = SUPP(X \rightarrow Y) = \frac{card\{t \in T \,/\, X \cup Y \subseteq t\}}{card\{T\}} \qquad \text{(Eq. 2.6)}$$

$$C = CONF(X \rightarrow Y) = \frac{card\{t \in T \,/\, X \cup Y \subseteq t\}}{card\{t \in T \,/\, X \subseteq t\}} \qquad \text{(Eq. 2.7)}$$

Ces quantités permettent de caractériser une règle d'association donnée, c'est-à-dire l'association entre certains produits X achetés par le client, et d'autres produits Y vendus à la même personne en même temps. Le support indique pour cette règle, le pourcentage de transactions ayant impliqué les produits X, ce qui permet de connaître les achats fréquents. La confiance met en évidence le pourcentage de transactions comprenant les produits Y si les produits X y sont aussi, donc éventuellement de prévoir des promotions d'un pack de produits X et Y. L'algorithme d'extraction des règles sélectionne celles dont les quantités S et C sont supérieures à un certain seuil, défini par l'utilisateur. Les travaux qui ont suivis ont eut pour but d'améliorer la vitesse de traitement de ces algorithmes, et de proposer la prise en compte de taxinomies, c'est-à-dire de hiérarchies d'objets (plusieurs articles de vêtements sont regroupés dans la même catégorie, [Srikant, 1995]).

Enfin des recherches ont pu aboutir à des algorithmes intégrant aussi des séries temporelles, comme le parcours caractéristique d'un candidat à un poste donné ou l'analyse de séries boursières, ce que l'on nomme des *séquences fréquentes*. Il ne s'agit plus de trouver les motifs les plus fréquents dans un ensemble de données, mais de trouver les relations causes-effets les plus significatives de la base. Le principe se base là encore sur la notion de support, vue plus haut. Son point fort est de découvrir des événements fréquents mais pas obligatoirement contigus. Des extensions ont été proposées pour prendre en compte certaines contraintes temporelles (par exemple des achats effectués de manière rapprochée dans le temps, ou ceux effectués aux alentours de certaines fêtes).

2.3. L'apprentissage par renforcement

2.3.1. Introduction

L'apprentissage par renforcement est une catégorie à part entière mais peut être considéré comme de l'apprentissage non supervisé, car dans le cas des deux types de problèmes traitables par ces méthodes, aucune action optimale n'existe dans une configuration donnée, mais l'algorithme doit en identifier une qui maximise le bénéfice obtenu par la réponse de l'environnement.

Le principe général d'un apprentissage par renforcement est de savoir quoi faire (quelles actions effectuer en fonction de la situation courante de l'*environnement*, c'est-à-dire l'espace sensoriel du système), en maximisant le bénéfice de ces actions. Les interactions (actions) avec l'environnement sont réalisées par des *agents*, sortes d'entités biologiques, mécaniques, informatiques ou autre, par l'intermédiaire d'une interface entre les deux (les *capteurs* et les *effecteurs*). La grande différence par rapport aux apprentissages supervisés est qu'on ne lui donne jamais d'informations sur les actions à effectuer (pas d'associations possibles donc). Au lieu de cela, le système va produire une réponse positive ou négative après chaque action que va prendre l'algorithme. Ainsi, les seuls renseignements dont dispose un système pour s'adapter est une information globale sur sa performance. On perçoit tout de suite que des résultats valables ne peuvent être obtenus qu'après de longues séquences d'évaluations des messages reçus. Si l'on reprend un exemple simple et compréhensible tiré du domaine du jeu, même s'il nous éloigne de la télédétection, par exemple un jeu de Morpion (mais tout autre exemple serait valable), on peut considérer que l'ordinateur doit apprendre à jouer à ce jeu qu'après avoir joué certains coups et obtenu le résultat final de la partie (match gagné, perdu ou nul). Ce sont les pénalités apportées par le résultat de la partie qui vont influencer sa manière de jouer. Il est important de voir dans cet exemple qu'il y a un net décalage entre le mo-

ment où il faut choisir un coup et le moment où il sera "évalué" (où l'on connaîtra réellement l'apport du coup dans la partie), et encore dans cet exemple, il est difficile d'associer un coup précis au résultat du match. Grâce à cet exemple, on voit toute la difficulté (pour l'ordinateur et pour nous aussi !) d'un algorithme de renforcement : puisqu'à chaque étape, il n'y a pas de coup optimal, il faut qu'il trouve un compromis entre exploration et exploitation des résultats. Généralement, l'algorithme va exploiter les actions fructueuses qui ont déjà été essayées par le passé, bref il va devoir utiliser ses connaissances apprises (ang. *knowledge*). Cependant, pour découvrir ces actions, il doit tester des cas nouveaux, non explorés. Ce dilemme (explorer ou non des espaces vierges), est évoqué dans [Sutton, 1998], qui présente une bonne introduction au problème. Nous pensons qu'il n'existe pas de solution générale, mais qu'une solution optimale passerait par la considération des deux techniques à la fois, de manière non exclusive. De nombreuses références sur la théorie des classifieurs se trouvent dans [Herbrich, 2002] et dans [NET 3].

2.3.2. 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, 1986]). 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 serait par exemple une image classifiée par un expert, le système recevrait 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 action à effectuer 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 prouvèrent dans [Barto, 1985] la convergence dans le cas de sorties binaires, composant un ensemble de motifs linéairement indépendants. Cependant, il semblerait que ce soit un système difficile à mettre en place, dû aux nombreux paramétrages à effectuer.

2.3.3. Le cas des systèmes de classifieurs

Un exemple souvent évoqué d'apprentissage par renforcement sont les systèmes de classifieurs, l'objet de notre stage, que nous décrivons en détail dans la partie suivante et dans le troisième chapitre. Nous y parlons aussi des nombreuses extensions qui ont été apportées aux systèmes de base, qui ne peuvent être comprises qu'après la lecture du début du chapitre.

Le Q-Learning de [Watkins, 1989] et discuté dans [Kaelbling, 1996] constitue un autre exemple d'apprentissage par renforcement, mais ne sera pas détaillé ici.

2.4. Les systèmes de classifieurs

2.4.1. Description d'un système de classifieurs

Les fondements théoriques des premiers systèmes de classifieurs ont été décrits par [Holland, 1962]. Un système de classifieurs est une méthode intégrant apprentissage symbolique et évolution. Des *règles*, c'est-à-dire des sortes de formules permettant de décrire au programme la conduite à tenir dans un cas particulier sont apprises, voire oubliées, en fonction de leurs performances et de l'environnement courant. On remarque que ces règles ne seront introduites ni par le programmeur, ni par un expert du domaine. L'algorithme de sélection de ces règles joue donc un rôle capital.

L'un des premiers systèmes de classifieurs, défini par [Goldberg, 1989a], était composé des parties suivantes :

- Une base de classifieurs représentant les connaissances du système.
- Une interface d'entrée composée de capteurs. Ils permettent de traduire en message d'entrée les évènements qui se sont produits dans l'environnement.
- Une interface de sortie composée d'effecteurs. Ils permettent d'agir sur l'environnement en produisant certaines actions, pilotées par les messages de sorties du système.
- Une liste de messages, contenant tous les messages *actifs*, c'est-à-dire tous les messages qui coïncident avec l'entrée courante du système.

Pour gérer ces différents objets, un algorithme de sélection est nécessaire. Il se décompose en deux parties (éléments circulaires sur la **Fig. 1**) :

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

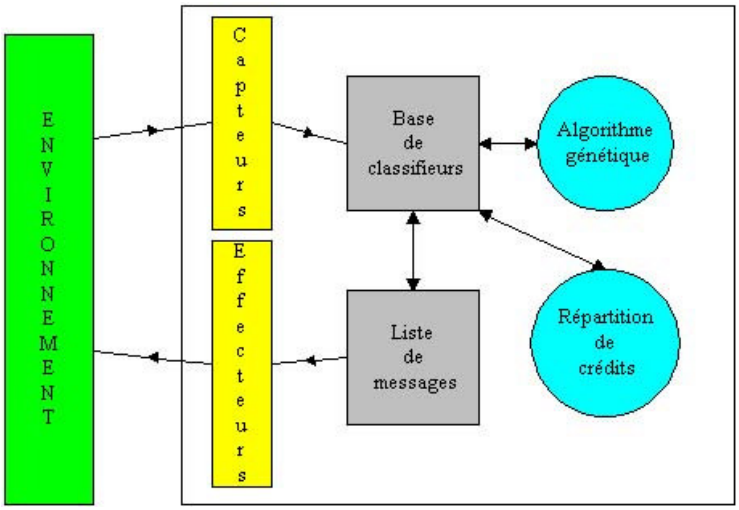


Fig. 1 : Structure d'un système de classifieurs

Le système de répartition de crédit (ang. *bucket brigade algorithm*) a été décrit par J. 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 capteurs déclenchent des classifieurs, ils s'inscrivent pour une vente aux enchères. Chaque classifieur y propose une enchère, généralement proportionnelle à sa force. 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 capteurs, 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 (porteurs d'eau). Le fait que le système n'a besoin que d'une réponse positive ou négative de l'environnement, en fait un apprentissage par renforcement.

Un algorithme génétique est utilisé dans la grande majorité des cas pour brasser entre eux tous ces classifieurs. Son rôle est important puisque c'est lui qui crée réellement les classifieurs. Il faut permettre à la population de se renforcer en gardant les meilleurs classifieurs, mais aussi de s'enrichir de nouveaux individus, qui peuvent alors aboutir à de meilleures solutions. Nous détaillons dans le chapitre 4 le principe général des algorithmes génétiques. Il y a deux méthodes de traitement [NET 7] : soit on remplace toute la population (GRGA, *Generational Replacement Genetic Algorithm*), soit on en remplace qu'une partie (SSGA, *Steady State Genetic Algorithm*). C'est généralement la seconde solution qui est adoptée. Les travaux sur les classifieurs sont relativement récents. Seules deux approches ont été étudiées concernant la relation entre un individu et un classifieur :

- L'approche de Pittsburgh [DeJong, 1988] considère que la population est un ensemble de bases de règles. 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 du Michigan considère quant à elle que chaque règle est un individu. Les règles sont évaluées de manière individuelle par l'algorithme du *bucket brigade* 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 a fait le plus ses preuves.

Un système de classifieur est dit *évolutif* s'il présente une capacité d'adaptation : il peut alors modifier son comportement et ses connaissances en fonction de la situation de l'environnement à un instant donné. Ainsi il peut traiter des problèmes non plus fixes (comme c'est le cas lorsqu'on utilise un algorithme génétique pour un problème d'optimisation de fonction), mais évolutif dans le temps. Supposons que l'on applique les systèmes experts à la télédétection et que l'on souhaite déterminer les classes d'une image. Si au cours de la reconnaissance la taille des classes doit changer (d'autres classifieurs ont apporté leurs contributions pendant ce temps et ont modifié le découpage actuel de l'image), il ne faudra plus forcément appliquer les mêmes stratégies de classification. Par exemple, on peut imaginer que les pixels sont classés selon leur couleur ou leur proximité par rapport à d'autres pixels classés. Cependant il restera toujours entre les classes des pixels non classifiés, ou correspondant à plusieurs classes (pixels mixtes). Lorsqu'un classifieur intégrera l'un de ces pixels restants à l'une ou l'autre des

classes, il faudra adapter dynamiquement l'algorithme : certaines règles qui ne traitaient que les cas simples (pas de pixels mixtes) deviendront désuètes, et parallèlement il faudra de nouvelles règles efficaces pour les autres cas. Les systèmes évolutifs sont conçus pour traiter ce genre de problème.

2.4.2. Définition d'un classifieur

Tout d'abord, un classifieur (dans le sens composant d'un système de classifieur) est une règle représentant les connaissances du système, de la forme suivante ([Wilson, 1999]) :

<*condition*> : <*action*>

où chaque partie est un message comportant des 0 ou des 1, permettant de coder des informations élémentaires, comme une couleur, une forme ou tout autre constante.

- <*condition*> est utilisée comme un message d'entrée dans le système, correspondant à une condition à remplir pour déclencher cette règle (la règle est dite alors *active*).

- <*action*> est le message de sortie correspondant, codant l'action à effectuer lorsque cette règle est active.

On peut donc résumer une règle à :

si <*condition*> alors <*action*>

On voit rapidement que pour raccourcir la base de règles, on peut mettre plusieurs conditions au lieu d'une par règle, ce qui ne changera rien à l'affaire. Ainsi, pour généraliser certains cas, on utilise le symbole '#' (appelé le *wildcard* ou le *joker*) permettant de coder l'une ou l'autre des valeurs binaires (0 ou 1) de la condition. On appelle *spécificité* d'un classifieur, le nombre de valeurs binaires d'une condition. S'il n'y a pas de joker, la spécificité est maximale et vaut la longueur de la condition, si tous les caractères sont des jokers, la spécificité est nulle. Ainsi dans l'exemple suivant, la base de règles BR_1 est parfaitement équivalente aux deux bases BR_2 et BR_3, mais pas à la base BR_4 :

BR_1	001010 : 11100 101010 : 11100 001110 : 11100
BR_2	#01010, 001110 : 11100
BR_3	001#10, 101010 : 11100
BR_4	#01#10 : 11100

Cela paraît évident, mais il faudra en tenir compte au moment des manipulations des classifieurs par l'algorithme génétique, pour contrôler le niveau de généralisation que l'on rajoute à chaque fois : ainsi on ne peut croiser deux classifieurs n'importe comment.

2.4.3. Quelques améliorations aux systèmes classiques

Même si l'on peut imaginer d'autres méthodes de gratification et 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), simplement l'apprentissage peut être plus efficace dans un cas ou dans l'autre ([Richards, 2001]).

[Richards, 1995] par exemple, ajoute un bruit aléatoire à l'enchère du classifieur vainqueur pour promouvoir l'exploration de l'espace de classifieurs. De plus grâce à deux sortes de taxations supplémentaires, il évite que des classifieurs inutiles mais avec une grande force ne persistent. Il définit ainsi une taxe à vie (ang. *life tax*), 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 inactives ont de plus en plus de chances d'être utilisées.

Reste le cas où aucun classifieur ne satisfait le message envoyé par les capteurs. [Robertson, 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, on va générer aléatoirement un classifieur avec une certaine force (qui est assez complexe à déterminer car elle ne doit être ni trop faible, ni trop forte) pour traiter ce message.

Enfin, il semble logique d'éliminer en priorité les classifieurs de faible force. Cependant le simple remplacement des plus mauvais individus est un peu expéditif et doit être amélioré. Par exemple [DeJong, 1975] proposa de créer une sous population d'individus peu performants en les croisant.

De nombreuses autres extensions ont été proposées, nous pouvons citer :

- **Les systèmes de classifieurs flous**, Fuzzy Classifier System [Rendon, 1997] :

  Ils permettent d'utiliser des variables réelles afin de résoudre certains problèmes de contrôle de tâches dans des systèmes dynamiques continus. Rappelons que la logique floue permet d'utiliser des facteurs de certitude *en langage naturel* (ainsi qu'une base de règles floues associées [Mascarilla, 1995]) comme chaud, froid, sûr, certain, peu sûr, etc. L'implémentation concrète d'un tel système de classifieurs est la même, mais on y incorpore un module permettant de faire la correspondance entre des messages flous et les classifieurs. Malgré la puissance apparente de ces classifieurs, il semblerait selon [Furuhashi, 1993] que lorsque le système cherche de nouveaux classifieurs, l'imprécision transférée d'un classifieur à un autre explose et le système ne peut plus rien déduire.

- **Les niches implicites** :

  Pour maintenir une certaine diversité dans la population de classifieurs, [Horn, 1994] a exploité un système de compétition dans lequel les ressources (c'est-à-dire la quantité de bénéfices 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. L'étude de Horn a permis de montrer que sans sa méthode, l'algorithme génétique seul associé aux systèmes de classifieurs ne pouvait pas maintenir cet ensemble coopératif de règles.

- **Les systèmes XCS**, XCS Classifier System [Wilson, 1999] :

Pour palier la limitation du modèle traditionnel dûe à la seule utilisation des valeurs binaires 0 ou 1, alors que la majorité des problèmes nécessitent des variables continues, [Wilson, 1999] a proposé une amélioration en considérant une représentation réelle : on code les valeurs des messages par un intervalle réel (par exemple [2.1 ; 5.3]) ou bien par un centre et un rayon (le même exemple : [3.7 ; 1.6]). Dans [Wilson, 1999] il y ajoute aussi un système prédictif : il remplace la fonction fitness habituelle par une prédiction de la pénalisation (ou du gain). On privilégie ainsi les classifieurs qui réussissent le mieux à prédire le gain obtenu lorsque le système les sélectionne.

- **Les s-classifieurs** [Koza, 1992] :

Les classifieurs, c'est-à-dire les individus croisés par l'algorithme génétique, sont des expressions Lisp, plus particulièrement des arbres, dont l'avantage est une évaluation rapide. Bien que ces expressions puissent être écrites de manière linéaire, elles sont visualisées et traitées par les opérateurs génétiques comme étant des arbres. 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. Le mécanisme d'évolution par croisement (ang. *crossover*) consiste à échanger des sous-arbres, associé à une pondération permettant l'échange plus rare de feuilles que de sous-arbres plus importants. Les mutations sont quasi inexistantes et jouent un rôle très réduit. 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. L'évaluation des individus va fournir un comportement qui va pouvoir être mesuré dans l'environnement : cette technique a été utilisée dans la classification d'images et le traitement d'images satellitaires [Banzhaf, 1999].

- **La méthode COGIN,** COverage-based Genetic INduction [Greene, 1994] :

Elle est basée sur la notion des *espèces* et des *niches*, déjà introduite par [Goldberg, 1989a]. Une *espèce* est une sous-population d'individus (dans le sens classifieur, mais aussi de manière plus générale, ensemble de gènes utilisés par un algorithme génétique) qui reste stable au fur et à mesure de son évolution. Une *niche* est une portion de l'environnement dans laquelle une espèce peut espérer survivre, autrement dit les individus reçoivent 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, 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%).

- **Le modèle IMGA,** Island Model Genetic Algorithm [Whitley, 1993] :

Ce modèle distribué 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, qui peuvent traiter simultanément leur population. Ensuite, au bout d'*n* itérations, on envoie *x%* de la population sur le processeur voisin, qui remplace une partie de la sienne.

2.5. Conclusion

Nous venons de retracer ici de nombreux exemples d'apprentissage supervisé, ainsi que le cadre dans lequel s'inscrit l'approche que nous voulons présenter pour le traitement d'images de télédétection. Nos travaux s'inspirent profondément de l'apprentissage par renforcement et utilisent la notion de classifieurs vues précédemment, dans la partie précédente (partie 2.4.). Cependant, nous allons le voir dans le chapitre suivant, les images à examiner sont très complexes, et ce par la quantité d'informations à traiter (dimensions spatiales et spectrales imposantes) et la présence de bruit, d'informations manquantes, etc. Nous avons donc ajouté certaines améliorations aux systèmes de classifieurs traditionnels, pour qu'ils soient plus robustes à cette complexité. Ces perfectionnements sont tout l'objet du chapitre 3.

3. LE PROBLÈME DE LA DECOUVERTE DE REGLES DE CLASSIFICATION

3.1. Introduction

3.1.1. La classification d'images

Les satellites que nous employons à l'heure actuelle conduisent à de nombreuses utilisations dans un nombre tout aussi important de domaines, allant des télécommunications aux prévisions météorologiques, en passant par les exploitations militaires. L'une d'entre elles, la télédétection consiste à photographier une zone bien précise de notre Terre, avec une précision pouvant atteindre l'ordre du mètre. Les images résultantes sont ensuite analysées dans le but de produire diverses cartographies, ou permettent différentes études d'existence (présence de routes, d'immeubles, ...), de densité (détection de ville en fonction de la densité des routes), des objets situés sur le terrain.

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 est loin d'être automatisée : elle nécessite le travail long et fastidieux d'un expert qui est souvent obligatoire dans le cas d'une production non-informatisée, et dans le cas contraire elle s'appuie encore dans la plupart des cas sur des algorithmes statistiques qui ne sont pas forcément fiables, de part la qualité des informations produites. Ce que l'on recherche avant tout dans ces types de travaux, c'est la rapidité de l'outil informatique : le délai de calcul restreint permet notamment le traitement de données qui évoluent dans le temps, comme la pousse de la végétation ou le reflux des marées.

Nous rappelons ici que nous sommes en étroite collaboration avec le laboratoire Image et Ville de Strasbourg sur [NET 9] qui nous a fourni les principaux documents de travail sur Strasbourg. Ceux-ci ont pu nourrir notre implication dans le projet **Pôle Image** [NET 12]. D'autres documents sur Venise nous ont été accessibles grâce à notre participation dans le projet européen **TIDE**, détaillé dans [NET 10].

3.1.2. Notre approche

Notre approche se distingue particulièrement des autres méthodes d'analyse effectuées dans le domaine de la télédétection. Les méthodes de classifications traditionnelles déterminent, à partir d'une image satellite brute, grâce à une analyse supervisée ou non, des associations entre l'ensemble des pixels de l'image et certaines classes thématiques. Cependant le processus est à recommencer lorsque l'image est à peine modifiée, c'est-à-dire lorsque l'on change de zone géographique, de position (rotation, ...), voire même de résolution (images multi-échelles).

A la différence de ces méthodes, nous voudrions pouvoir disposer de ce que l'on nomme des *règles de classification*, c'est-à-dire que le système produise de lui-même une certaine forme d'information permettant à partir de ces mêmes pixels (la même information utilisée par les procédés classiques) de déduire les classes thématiques des zones couvertes, mais que cette forme d'information soit du mieux possible réutilisable sur une partie non traitée de

l'image ou sur toute autre donnée se rapprochant de l'image analysée, sans recommencer à chaque fois le long processus d'apprentissage. Ces règles devraient, 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 leurs 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 qui lui sera propre, 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 rendent compte de la performance bien réelle de ces règles.

Enfin les règles produites devraient être d'une manière générale *simples*, c'est-à-dire que le nombre de contraintes qu'elles imposent afin de discriminer les pixels doivent être très réduit. Là encore, rien ne doit paraître trop illusoire puisque nous verrons qu'un critère favorable à la simplification de ces règles est utilisé pendant la phase d'apprentissage.

Le principal intérêt de ces règles de classification est leur utilisation reproductible dans un domaine ou la fréquence d'analyse des images est intensive et routinière : nul ne garantit que la structure de donnée étudiée ici (qui bien entendu gage de la qualité des résultats) permettra une réutilisation totale, par exemple quelle que soit la date dans l'année de la prise de vue. Malgré tout, l'étude reste tout à fait valable avec d'autres types de règles adaptées aux besoins de chacun, même si elles se distinguent de la structure imposée dans le paragraphe 3.3.1. Cependant le concept, s'il est poussé plus en profondeur, doit nous permettre d'imaginer des applications extrêmement intéressantes comme par exemple la classification d'images vidéos, même si la technologie ne suit absolument pas actuellement. Le but est donc avoué : utiliser cette idée dans un contexte temporel dans lequel les images et les informations représentées évoluent dans le temps (par exemple, plus prosaïquement que la vidéo : les formes de végétation, les milieux aquatiques, ...).

Ces règles de classification ne pourront être que générées automatiquement par programme : le meilleur système d'apprentissage disponible capable de digérer tant de contraintes est incarné par l'algorithmique génétique, que nous allons présenter plus en détail dans le chapitre 4. Ces algorithmes rendent nos règles évolutives : en effet ces méthodes permettent la découverte de descriptions de classes qui ne sont pas dépendants de la structure utilisée pour les créer (réseaux de neurones, arbres d'induction, ...), mais sont pratiquement engendrés à partir de rien ce qui les obligent à une grande adaptation aux données fournies (d'où la notion de classifieurs évolutifs, exposés vers la fin de ce chapitre). Ainsi la syntaxe et la complexité de nos règles ne dépendent pas du nombre de couches d'un réseau de neurone ou de centres de classes abscons (voir l'apprentissage par formation de concepts dans l'annexe A). Ce type de raisonnement est très important dans le cadre de la télédétection dans lequel des images relativement complexes interviennent, que nous décrivons dans la partie suivante.

3.2. Les données analysées

3.2.1. Les données brutes

Les données brutes sont l'ensemble des informations de travail, c'est-à-dire celles sur lesquelles la tâche va être réellement exécutée. 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. Ces données brutes utilisées dans notre étude se présenteront toujours sous la forme d'images de même taille (au sens du nombre de pixels en largeur et en hauteur) que l'image experte utilisée conjointement par notre algorithme, puisque bien souvent, que le processus soit humain ou informatique, l'image experte sera générée à partir de ces mêmes données brutes. Celles-ci sont 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.

Mise 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, il faut aussi parler des nombreux problèmes techniques que l'on peut rencontrer dans ces images : la résolution spatiale est plutôt insuffisante (30 mètres pour un satellite Landsat, 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 enfermer quantité d'objets. On parlera par exemple de pixels mixtes (déjà évoqués plus haut). 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 sa réflectance. Mais le capteur a ses limites, certains objets même très brillants sont vraiment trop petits pour pouvoir être représentés dans l'image.

Voici un aperçu des différents outils permettant de créer des données brutes exploitables :

3.2.1.1. *Les images SPOT*

Le programme SPOT (Satellite Pour l'Observation de la Terre) permet le lancement de satellites de surveillance de la Terre (SPOT5 a été lancé le 4 Mai 2002) dont les images pourront œuvrer 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 ([NET 8c]) permettant de prendre d'un seul coup une bande au sol de 60 km de longueur. La précision 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 ([NET 8b]). SPOT5 permettra 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. Voici dans le cas de SPOT4, les fréquences choisies pour les 4 bandes spectrales d'analyse, adoptée à cause de la spécificité de la réponse spectrale de certains objets comme le sol, la végétation, le désert ou la neige :

Tableau 1 : Longueurs d'onde de SPOT (d'après [NET 8a])

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

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.

3.2.1.2. *Les images haute résolution*

Ces images d'une très grande précision (un pixel représente au sol une taille de 1,30 m x 1,30 m) ont été produite par un avion volant à haute altitude, au cours d'un projet visant à tester les capteurs embarqués sur SPOT (elles contiennent donc le même nombre de bandes spectrales). 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.

Voici une image de Strasbourg, 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 environnants de l'Esplanade. Une partie seulement de cette image a été étudiée (voir le chapitre sur l'étude de cas), compte tenu de certains dangers que peut signifier une résolution élevée. L'un des intérêts de cette image, outre sa très grande précision, 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 3.2.2. La photo a été traitée par une égalisation d'histogramme afin d'en voir clairement les contours (elle serait beaucoup trop sombre sinon). On peut aisément 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 cette ombre, car leur réflectance est fortement amoindrie et altérée. Lors de la phase de reconnaissance, on préfèrera donc placer l'ombre dans une classe à part, plutôt que de déterminer celle des pixels sous-jacents. L'image suivante 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 avec une légère augmentation de contraste. Le poids de cette image représente 3 Mo en mémoire.

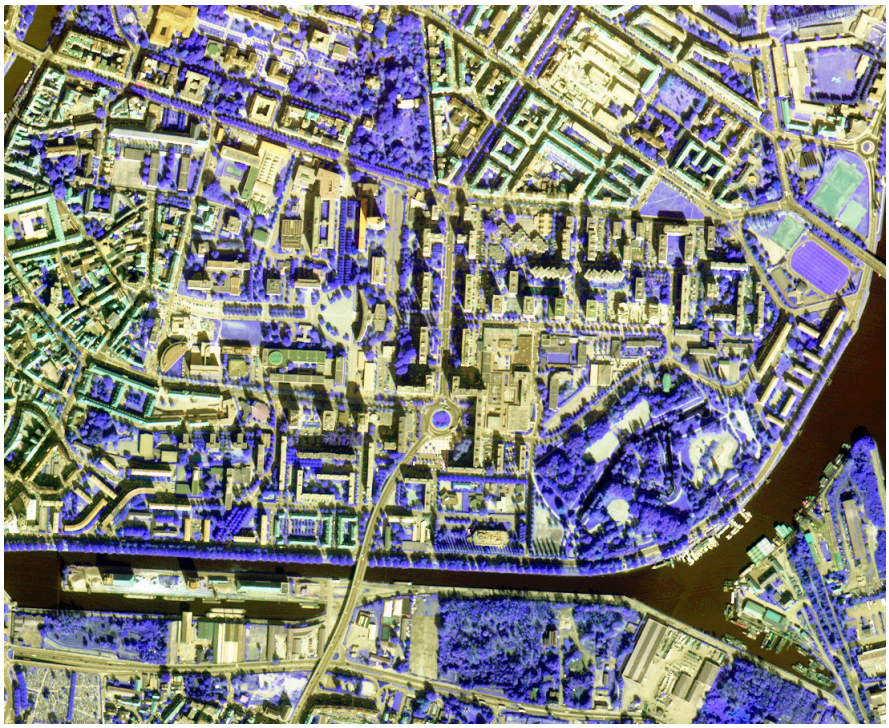


Fig. 2 : Image haute résolution de Strasbourg

3.2.1.3. *Les images hyperspectrales*

A la différence des images SPOT, les capteurs de ce type de satellite sont capables de brasser 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 réflectances pour toutes les longueurs d'onde du spectre étudié. Des satellites comme DAIS permettent une précision de l'ordre de 3 mètres par pixel, et définissent les valeurs de réflectance sur 16 bits, ce qui fait 65536 valeurs possibles ([DAIS, 2001]). D'autres satellites permettent une grande précision d'échantillonnage du spectre des pixels, comme CASI qui possède une étendue spectrale de 288 canaux ([NET 16]).

De telles images ne pourront jamais être affichées telles quelles à l'utilisateur. La prévisualisation de ces images étant tout de même importante dans un soucis de validation, des algorithmes d'affichage ont été développés, notamment au sein de notre équipe dans le cadre du logiciel Samarah ou de notre prototype. 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 de la page suivante représente une photo prise par DAIS, un satellite 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 jusqu'au terrain d'aviation du Polygone au Sud. 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 poids de cette image en mémoire dépasse les 220 Mo.

Fig. 3 : Image hyperspectrale de Strasbourg

Voici une image de San Felice (région de Venise), d'une taille de 932 x 368 pixels, prise par ROSIS, un autre satellite 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 poids en mémoire de cette image est proche de 80 Mo.

Fig. 4 : Image hyperspectrale de San Felice

3.2.1.4. *Les autres types de données*

Il existe bien d'autres formats de données possibles, mais nous ne les avons pas tous étudiés ici, tout simplement parce qu'elles ne nous ont pas été disponibles. Ces différentes sources d'information peuvent tout aussi bien être spatiales comme des images radar ou des ortho-photos (photos prises à la verticale d'un point donné), thermiques ou stéréoscopiques (images thermoformées en 3D). Nous nous sommes restreint à analyser la réponse spectrale du terrain, ce qui rend l'exploitation de ces photos inadaptée dans le cadre de notre étude. En effet, cette limitation permet de manipuler des structures de données plus simples et plus uniformes.

3.2.2. Les données expertes

Nous avons vu dans la partie 3.1.2. l'ensemble des souhaits dont on aimerait imprégner nos règles (rapides, simples, efficaces). Malheureusement la chimère devra s'arrêter là. Il est hors de question de produire de telles règles à partir de rien, ou du moins simplement à partir des seules données brutes du satellite. La phase d'apprentissage a besoin au préalable, pour suivre un déroulement correct, d'un certain nombre d'informations annexes sur les classes à trouver, dites *informations expertes*. L'apprentissage étudié ici est donc bien un apprentissage *supervisé*. Ces informations peuvent être produites indépendamment de plusieurs façons :

- L'expertise humaine, très coûteuse, permet néanmoins une analyse précise et intelligente 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 les situations extrêmes.

- L'analyse statistique, non supervisée, etc. produit un ensemble de classifications de manière systématique et rapide, mais pas toujours fiable. Ces analyses résultent d'algorithmes qui ne sont jamais parfaits, et qui ont tous leurs domaines de prédilection donc, à l'inverse, leurs domaines de vulnérabilité. Bien entendu ces classifications peuvent être validées par un expert humain, mais si pour obtenir une certaine qualité, cela doit être réalisé pour chaque pixel, ce type d'analyse n'apporte rien.

- L'exploitation d'images de télédétection pose souvent des problèmes non triviaux, que les méthodes statistiques ne peuvent pas résoudre. L'introduction de données sémantiques, structurelles ou de relations spatiales, 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*) peut bien entendu être utilisée sous forme de *règles de composition*, mais nous n'entrerons pas dans ce domaine qui relève bien souvent de la spéculation.

Nous allons voir plus tard, dans le chapitre sur l'étude de cas, la mise en œuvre des deux premières catégories d'informations : les deux premières études ont été validées par une expertise humaine et par la classification d'une chercheuse (Anne Puissant) travaillant au laboratoire Image et Ville Strasbourg. La troisième étude illustre l'utilisation des résultats d'une classification non supervisée dans la phase d'apprentissage. Durant cette phase, les renseignements se présenteront toujours sous forme d'une autre image, de la même taille que la zone à traiter (la synchronisation se fait naturellement dans la plupart des cas, par le fait qu'une image experte a été produite à partir des mêmes données brutes). Cependant celle-ci ne représente qu'une faible proportion de l'image utilisée durant la phase de test (plus que d'accélérer l'apprentissage, cet *échantillonnage* permet aussi une validation fiable sur les données originales). L'image experte présente exclusivement une information sur la classe *désignée* par l'expert pour chaque pixel de l'image brute, sous forme de jeux de couleurs attribués arbitrairement selon la sémantique de la classe en question.

Voici pour l'exemple, un tableau de correspondance entre les couleurs que nous avons choisies et leurs significations thématiques, dans le cadre de la classification d'un environnement urbain pour le projet **Pôle Image** :

Tableau 2 : Code couleur des classes thématiques

COULEUR	SIGNIFICATION
	OMBRE
	EAU
	VEGETATION DENSE (ARBRES, ...)
	PELOUSE (TERRAINS DE SPORT, ...)
	AUTRES ESPACES VERTS (ARBUSTES)
	TOIT TUILE
	TOIT / TÔLE / TERRE BATTUE
	TOIT BETON
	SOL GRAVIER
	SOL BETON / MACADAM
	ROUTE

Dans ce projet, l'expert a utilisé une analyse statistique des pixels (segmentation, ...), pour produire la classification associée à ce tableau, présentant l'information de manière spatiale :

Fig. 5 : Image experte de Strasbourg

Cette carte a été extraite directement à partir de l'image haute résolution présentée dans la partie 3.2.1.2., ce qui en fait son avantage puisqu'elle représente exactement la même zone. 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 et pour l'ombre, certaines rues du centre ville étaient considérées comme inondées. Il a donc fallut manuellement remplacer toutes les régions noyées par des zones ombragées. Cependant ce traitement se révèle être très laborieux dans la partie inférieure de l'image, au niveau de la zone située près de l'actuel *Ciné Cité* : la présence d'eau et d'ombre est ici mêlée et il a été décidé de ce fait de ne pas réaliser la phase d'apprentissage des règles dans cette région de l'image.

Nous verrons plus loin, dans la partie sur l'expérimentation d'autres types d'images expertes, entre autre des images produites par des classifieurs non supervisés. Cependant dans le cas de Strasbourg, toute classification experte affinée a été auparavant validée sur le terrain par nos propres soins.

En outre, pour les besoins spécifiques de nos classifieurs (que nous allons détailler dans la partie suivante), nous n'avons pas voulu nous contenter simplement d'un ensemble de

classes mis à plat. Nous avons défini une hiérarchie entre les différentes classes à reconnaître pour plusieurs raisons :

- D'une part le système de notation employé dans la fonction d'évaluation de ces classifieurs doit pouvoir produire une note révélant la performance de ces classifieurs tant sur le plan individuel que sur le plan global, en passant éventuellement par la mesure de la qualité de sous-arbres. Cela donne une information claire à l'utilisateur sur les arbres sémantiques de classes qui ont été problématiques durant l'apprentissage.

- D'autre part ce système hiérarchique va pouvoir être utilisé dans la phase de validation dans laquelle on va comparer le degré de spécialisation d'une règle par rapport à une autre en observant leurs places respectives dans l'arbre mais aussi les informations d'ordre syntaxiques contenues directement dans leurs structures de données.

Voici donc un exemple de hiérarchie de classes thématiques associé à la classification experte de Strasbourg :

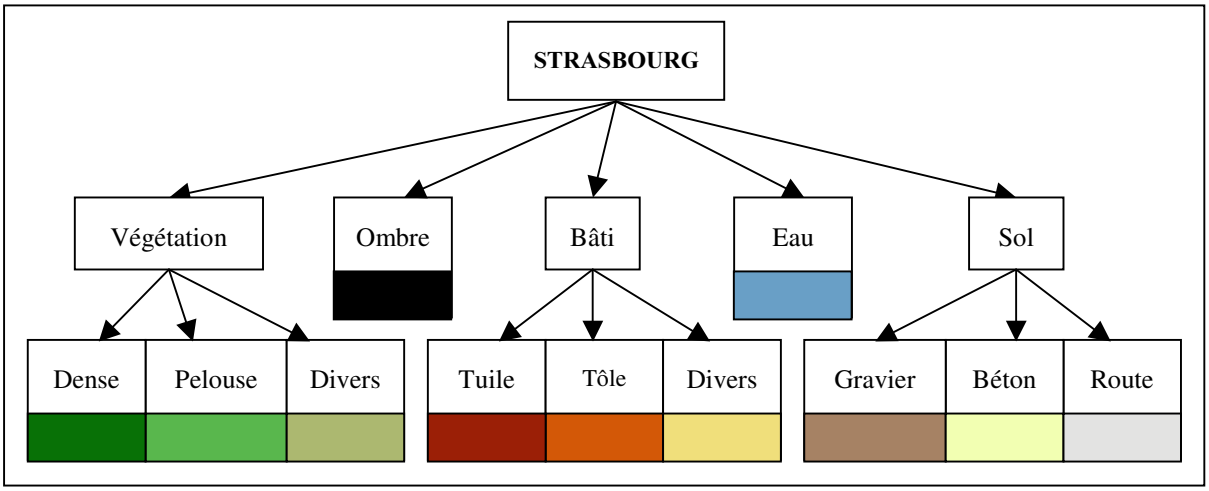


Fig. 6 : Hiérarchie de classes thématiques

Cet arbre n'est pour l'instant qu'un exemple, mais il résulte des remarques d'un expert du domaine de la télédétection, physicien et informaticien, qui sont les suivantes :

- 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 visible 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.

- l'eau se distingue profondément des catégories pré-citées : 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 abscisses et les valeurs de réflectance croissantes en ordonnées).

- Il nous est donc apparu utile de créer au moins trois sous-arbres distincts pour la végétation, le bâti (bâtiments et autres constructions humaines) et l'eau.

- Concernant l'ombre et les difficultés potentielles relatives à la discrimination des régions couvertes par cette ombre, nous avons préféré lui attribuer une classe à part.

- Afin de mieux cibler notre programme, et de distinguer sa performance sur les objets situés en altitude (toits, ...) de ceux aux sols (terrains, routes, ...), nous avons délibérément coupé la classe se rapportant au minéral en deux, de façon à pouvoir obtenir une note distincte pour ces deux groupes de classes. Par exemple la classification d'objets au sol est souvent prédisposée aux contre-performances dûes à la présence d'ombre.

3.3. Description de nos règles de classification

3.3.1. Syntaxe

Nous venons de le voir, le but de nos classifieurs est d'associer à un pixel caractérisé par un *vecteur spectral* (ensemble des valeurs de réflectance de ce pixel pour chaque bande) l'instance de la classe thématique correspondante. Il nous faut donc une structure de données capable de mémoriser une telle association.

La représentation général d'un classifieur (ou règle de classification), permettant la définition d'une classe donnée, reprend exactement la représentation classique vue précédemment :

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

Plus précisément, cela signifie que si le pixel de l'image brute satisfait la condition donnée, il devra être considéré comme appartenant à la classe thématique 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 :

- si le formalisme de la condition est trop restrictif, il y a un risque de produire des règles relativement inaccessibles. N'oublions pas que ces règles sont destinées à être générées automatiquement par l'algorithme, donc il faut supposer que toutes les contraintes présentes dans la condition (s'il y en a plusieurs) ne doivent pas être intimement liées pour que la règle puisse s'*activer* (c'est-à-dire pour que le pixel satisfasse la règle).

- si à l'inverse la syntaxe est beaucoup plus légère, il y aura des problèmes de recouvrements : de nombreuses règles seront actives pour le même pixel et l'algorithme devra

faire un choix après l'évaluation des règles (extra-règle), alors que c'est justement le rôle de celles-ci (choix intra-règle).

Tout d'abord, rappelons qu'un pixel est en fait un vecteur spectral, définissant une valeur de réflectance pour chaque bande :

$$<\text{pixel}> := [\ b_1\ b_2\ b_3\ ...\ b_n\ ] \qquad \text{(Eq. 3.1)}$$

où n est le nombre total de bandes dans l'image.

Après plusieurs essais de structures légèrement différentes, nous avons décidé que la condition 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 pixel et pour une bande donnée (soit 255 pour les pixels définis sur 8 bits, soit 65536 pour les pixels définis sur 16 bits), qui permet de découper l'espace des valeurs spectrales en deux espaces : celui des valeurs souhaitées pour les pixels d'une même classe, et le reste.
Pour généraliser le concept et mieux cibler les classes, nous affectons à chaque bande disponible dans l'image brute un ensemble d'intervalle éventuellement disjoint.

Voici, avant de rentrer dans les détails, la forme complète de la partie <*condition*> d'une règle :

$$<\textit{condition}> := E_1 \wedge E_2 \wedge E_3 \wedge ... \wedge E_n \qquad \text{(Eq. 3.2)}$$

- n est le nombre total de bandes.

- 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 :

$$E_i := [m_1 ; M_1] \vee [m_2 ; M_2] \vee [m_3 ; M_3] \vee ... \vee [m_p ; M_p] \qquad \text{(Eq. 3.3)}$$

- $[m_j ; M_j]$ est appelé un intervalle spectral, où m_j et M_j sont respectivement les réflectances minimale et maximale autorisées d'un pixel pour la bande i, pour que celui-ci active la règle (dans ce cas, $m_j \leq b_j \leq M_j$).

- Ces intervalles 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 :

$$E = [11 ; 105] \vee [138 ; 209] \vee [93 ; 208] \Leftrightarrow E = [11 ; 209] \qquad \text{(Eq. 3.4)}$$

- Le nombre d'intervalles spectraux dans un ensemble peut varier pour chaque règle et pour chaque ensemble, mais il doit être non nul ($p > 0$).

Pour satisfaire la règle, un pixel, défini a fortiori sur toutes les bandes de l'image, doit donc satisfaire chacun des ensembles d'intervalles spectraux de chaque bande et pour satis-

faire un ensemble d'intervalles, 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 la simplicité dans les résultats présentés à l'utilisateur et par le fait que l'expression de multiples contraintes reste tout de même compacte. De plus la notion d'ensemble d'intervalles est à même de saisir l'uniformité des pixels qui appartiendraient à la même classe. Ces règles permettent enfin une rapidité d'évaluation qui peut être importante dans certains domaines d'applications (images animées, ...).

On peut bien sûr imaginer des structures plus complexes, par exemple en utilisant le voisinage des pixels ou bien sous forme d'équations mathématiques munies de multiples opérateurs. Ceci pourrait être accessible (plus au sens de la durée de traitement que des problèmes de représentation) grâce à la programmation génétique visant à manipuler de telles formules sous forme d'arbres. Dans le cas d'une image hyperspectrale, il est possible de calculer certains indices synthétiques spécifiques au thème recherché, généralement à partir d'une combinaison linéaire des différentes valeurs spectrales. Par exemple pour SPOT, l'indice de végétation normalisé s'exprime le plus couramment par ([NET 13]) :

$$IVN = \frac{XS3 - XS2}{XS3 + XS2} \qquad \text{(Eq. 3.5)}$$

De la même façon, l'indice de brillance est utile pour caractériser les sols nus et se calcule par la moyenne quadratique des canaux rouge et proche infrarouge :

$$IB = \sqrt{XS2^2 + XS3^2} \qquad \text{(Eq. 3.6)}$$

Ces indices sont facilement généralisables dans le cadre d'un nombre plus important de bandes. Néanmoins, nous avons choisi de retarder l'étude de la production automatisée de telles formules jusqu'à la poursuite d'une thèse, compte tenu de certains obstacles présentés par la recherche automatisée de tels arbres. Cependant le calcul de ces indices est proposé en pré-traitement dans notre prototype, afin de faciliter la validation des résultats produits.

3.3.2. De la création à l'évolution

Ces règles délimitent donc la zone spectrale correspondante à la classe à traiter. Au départ de l'algorithme elles sont initialisées en fonction des classes attribuées par l'expert sur chaque zone validée de l'image, en considérant à chaque fois la valeur de réflectance minimum et maximum calculée pour chaque bande (domaine de valeurs des pixels) : on recherche donc au préalable un ensemble d'intervalles relativement proches de la solution à trouver, mais bruités par un facteur aléatoire pour obtenir un assortiment de règles assez diversifié.

Ensuite on confie ce *pool* initial à un algorithme génétique qui est chargé de les faire évoluer vers une solution acceptable. Nous verrons plus en détail le fonctionnement et les paramètres de cet algorithme dans le chapitre suivant. Il faut simplement retenir ici que notre système de classifieurs est en fait un ensemble de classifieurs, répondant chacun à une classe précise (dont le nombre est déterminé par l'expert), et évoluant de manière distincte. Plusieurs raison à cela : les règles efficaces ne perturbent pas les autres règles, qui mettent plus de

temps pour s'améliorer, au cours de la fonction d'évaluation qui serait alors commune et qui écraserait prématurément les mauvais. D'autre part, le processus peut ainsi être *parallélisé* (exécuté sur plusieurs machines en même temps), ce qui en augmente la vitesse.

Pour répondre au critère de qualité, il nous faut un système de notation basé sur l'image experte : les classifieurs sont jugés en fonction de la surface qu'ils couvrent pour chaque classe, ainsi qu'en fonction de la surface couverte par l'expert pour les mêmes classes, et il en découle une note réelle entre 0 et 1. Les règles sont donc sélectionnées en fonction de la note obtenue pour une certaine classe, mais il est aussi possible de définir des notes globales pour l'arbre thématique ou pour une sous-branche. Bien que chaque classifieur évolue indépendamment les uns des autres au cours de l'apprentissage, on pourra connaître à la fin la performance générale des règles obtenues. Si l'utilisateur exigera une efficacité optimale pour certaines classes (par exemple la végétation pour un géographe), il pourra relancer le processus avec un critère plus sélectif pour le sous-arbre thématique correspondant.

Il est important de souligner que l'on place dans la phase d'évolution, aussi longue que l'on souhaite, toute la performance des classifieurs, pour ensuite produire une information de qualité malgré la simplicité de leur représentation, de la même façon qu'un professeur aboutira à une information claire et exploitable après de longues recherches, qui conservera toujours infailliblement en regard des étudiants toute son exactitude.

3.4. Conclusion

Nous venons de définir notre objet de travail, les règles de classification, qui seront à la fois le matériel utilisé pour transmettre l'information à travers les différents modules de notre programme, mais aussi l'enveloppe contenant les résultats finaux présentés à l'utilisateur. De cela découle quelques contraintes, comme la simplicité des règles produites, auxquelles notre structure répond avec ténacité. Le problème qui est de découvrir les relations entre tout pixel de l'image brute à l'une des classes sémantiques proposées par l'expert, s'en trouve donc à moitié résolu, par le biais des règles que nous venons d'étudier. Il ne reste plus qu'à définir un algorithme capable de les manipuler, de les juger, et de converger en conservant la meilleure d'entre elles. Cet objectif est parfaitement rempli par l'algorithmique génétique, dont nous présentons les principales caractéristiques dans le chapitre suivant.

4. PROCESSUS DE LA DECOUVERTE DE REGLES

Les règles présentées dans le chapitre 3 sont relativement simples du point de vue de la syntaxe, par contre elles se révèlent être plutôt complexes lorsque l'on doit découvrir de manière automatique la kyrielle de nombres qui les composent (plus exactement les bornes de tous les intervalles de ces règles) afin d'obtenir des classifieurs optimaux. En effet, si nous demandons à l'algorithme d'utiliser 5 intervalles par bande afin d'être sûr de pouvoir traiter tous les cas, alors la définition d'une règle est équivalente à la définition d'un point dans un espace à 30 dimensions dans le cas simple de SPOT (3 bandes). Ce nombre de dimensions atteint rapidement 200 voire plus dans le cas de l'hyperspectral.

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 ce type de problème d'optimisation et pour ses qualités puissantes d'exploration et de recherche. L'idée des algorithmes génétiques a été proposée pour la première fois par [Holland, 1975] puis reprise et étendue notamment par [Goldberg, 1989b] et [Wilson, 1999]. C'est en fait un modèle d'apprentissage stochastique s'inspirant des principes de l'évolution naturelle des espèces de Darwin pour déterminer si ce n'est l'une des solutions optimales, au moins une solution s'en approchant.

Afin de spécifier un algorithme génétique, plusieurs notions fondamentales, s'articulant autour de l'*individu*, de la *fonction d'adaptation* et de la *génération de la population suivante* doivent être définies.

Ainsi, l'algorithme génétique introduit, comme dans le modèle de Darwin, une notion d'*individu*, correspondant à une structure atomique qui doit porter toute l'information (dont l'ensemble est nommé *génotype*) permettant de caractériser son comportement, ses manifestations externes (nommé *phénotype*) dans un environnement donné. Dans notre cas, l'individu en question est incarné clairement par une règle permettant, grâce à tous les paramètres vus au chapitre 3 qu'elle intègre, de déterminer la classe thématique des pixels de l'image. Un individu représente à la fois le moyen de mémoriser les données à optimiser au cours de l'apprentissage, mais aussi à la fin une solution à notre problème. Ces informations sont stockées conjointement en un ensemble de *gènes* appelé *chromosome* (structure contenant l'information utile).

Afin d'avoir une exploration de l'espace équiprobable, 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 d'adaptation (ang. *fitness*), dont le rôle est d'affecter à chaque individu une note en fonction de sa capacité par rapport aux autres à répondre au but attendu : dans notre cas, les meilleurs individus seront ceux qui sauront déterminer au mieux la classe correspondante à un ensemble de pixels de l'image, et ce en fonction du 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 seront conservés, d'autres seront éliminés, et d'autres encore donneront naissance à de nouveaux individus permettant de les faire

évoluer. Une génération consiste à effectuer un cycle de l'algorithme, c'est-à-dire à faire évoluer une population complète. Cette opération permet d'explorer la diversité des comportements possibles et d'en découvrir de nouveaux encore plus performants que celui des parents. On distingue deux opérations génétiques : le *croisement*, permettant d'obtenir deux nouveaux individus à partir de deux parents, et la *mutation*, beaucoup plus rare, qui modifie un ou plusieurs gènes du chromosome.

Ces trois notions fondamentales sont la clef de voûte de notre algorithme génétique dont voici le déroulement général :

ALGORITHME 2
Déroulement général de notre algorithme génétique

R est un individu (règle)
P, P' et P'' sont des populations (pool) d'individus

```
R := REGLE_INITIALE(données)                    // Création d'une règle selon les données
P := INITIALISATION(R)                          // Création d'un pool initial aléatoire
EVALUATION(P)                                   // Calcul de la fonction fitness
répéter
      P' := SELECTION_CROISEMENT(P)             // Sous-population à croiser
      P' := CROISEMENTS(P') U RECOPIE(P)        // Exploitation des individus
      EVALUATION(P')
      P'' := SELECTION_MUTATION(P')             // Sous-population à muter
      P'' := MUTATION(P'') U RECOPIE(P')        // Exploration de nouveaux individus
      EVALUATION(P'')
      P := RECYCLAGE(P,P'')                     // Recyclage générationnel
tant que CRITERE_ARRÊT(P) = false
```

Les fonctions principales introduites par cette présentation simplifiée de l'algorithme seront détaillées dans les sections suivantes.

4.1. Architecture de notre système de classifieurs

Nous avons conçu notre système de classifieurs comme une réunion de classifieurs indépendants. Nous en avons déjà évoqué les raisons plus haut (cf. partie 3.3.2.) : nous voulons éviter qu'un classifieur performant (c'est-à-dire comprenant des règles qui sont le plus approchantes des indications de l'expert) engloutisse un classifieur qui mettrait plus de temps à s'améliorer. De plus, l'indépendance garantit l'impartialité de la note obtenue lorsqu'on s'intéresse à la performance globale d'un type de classifieur particulier (par exemple tout ceux qui traitent les classes de type végétation, voir l'arbre thématique défini sur la Fig. 6).

Ainsi, chaque classifieur C_n de la figure suivante est l'union de tous les individus d'une population (pool génétique) donnée, spécialisée dans la découverte de règles d'une classe bien précise.

La figure 7 présente l'allure général de notre système de classification.

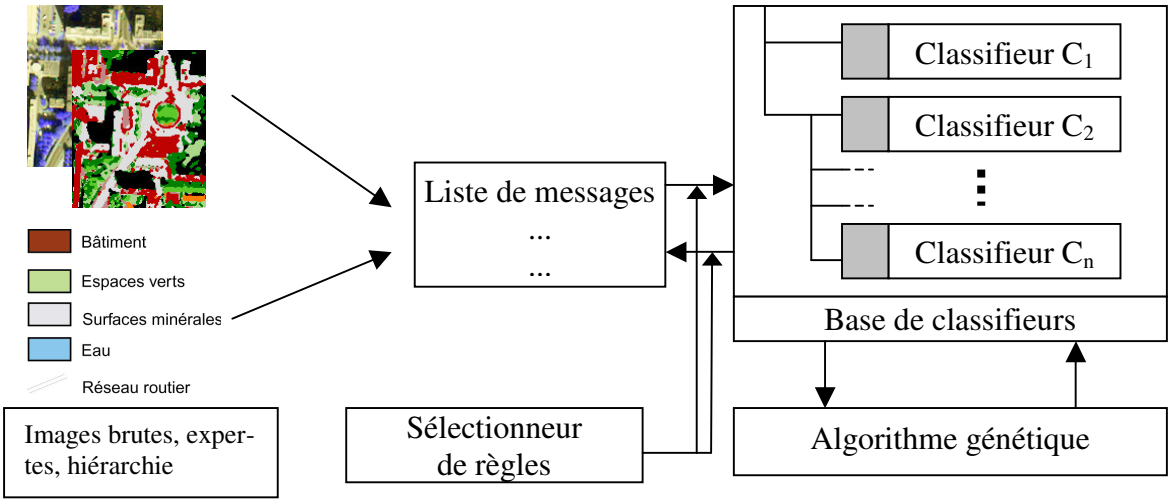


Fig. 7 : Système de classifieurs

L'utilisateur peut donc définir une hiérarchie de classifieurs (en pré-traitement) afin d'avoir une information plus précise sur les qualités de certains classifieurs, permettant éventuellement de relancer l'algorithme avec des paramètres mieux adaptés au traitement des classes correspondantes.

La liste de messages récupère les informations sur les images (valeurs de réflectance, ...) et les transmet à la base de classifieurs par l'intermédiaire du sélectionneur. Son rôle est de faire le tri entre les règles qui sont activées par les messages venant des images et celles qui ne le sont pas.

4.2. Codage d'un individu

La transcription informatique d'une règle, de manière à pouvoir être traitée correctement par les fonctions génétiques, est exactement celle que nous avons évoqué dans la partie 3.3.1. Il faut se souvenir que contrairement à la représentation binaire classique des gènes d'un chromosome, nous avons utilisé ici des valeurs entières pour coder les *allèles* (valeurs des gènes), donc une représentation beaucoup plus souple et puissante. En effet, elle permet de mieux se conformer aux données qu'un gène formé de valeurs 0 ou 1 : les valeurs de réflectance des images brutes et expertes sont entières. La représentation en mémoire d'un individu est indiquée sur la figure suivante.

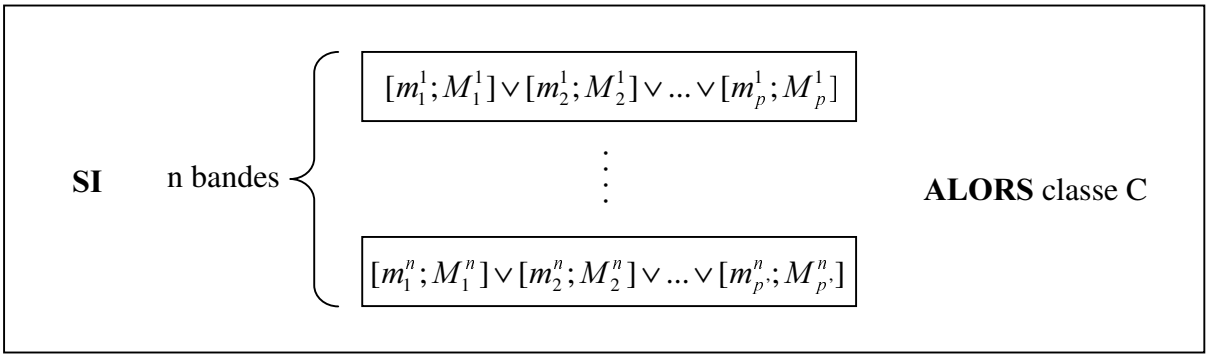


Fig. 8 : Représentation d'une règle

Les paramètres d'une règle sont les suivants :

- m_j^i et M_j^i indiquent respectivement la borne inférieure et supérieure du j-ième intervalle spectral pour la bande i. Chaque bande ne contient pas nécessairement le même nombre d'intervalles.

- C indique la classe de l'image experte que la règle est en train de définir.

Une règle permet donc de transcrire à la fois les ensemble d'intervalles spectraux qui caractérise le spectre des pixels de l'image, et la classe correspondante à cette caractérisation. Le codage exact de la règle en mémoire repose sur l'utilisation en C++ d'entiers pour la classe et les bornes, de tableaux d'entiers pour les ensembles d'intervalles, et de tableaux de tableaux d'entiers pour stocker la partie gauche de la règle (condition).

La figure suivante indique dans quelle mesure une règle peut caractériser le spectre d'un pixel 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 réflectance du pixel. Chaque intervalle est donc une contrainte sur le spectre : une bande en contenant plusieurs permet de décrire une disjonction de contraintes.

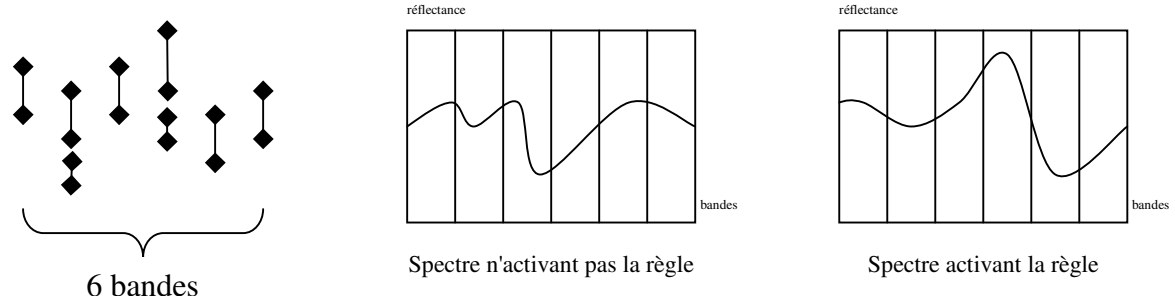


Fig. 9 : Correspondance entre une règle et un spectre donné

La partie suivante présente les différentes façons de manipuler cet objet, grâce aux opérateurs génétiques.

4.3. Les différents opérateurs génétiques

Il est temps de s'intéresser maintenant aux fonctions entre-apperçues dans le schéma général de l'algorithme.

4.3.1. Fonction d'adaptation

La fonction calculant la note (ang. *fitness*) des individus est le noyau d'un algorithme génétique. Dépendant fortement du domaine d'application, elle est à déterminer avec soin. Dans notre cas, cette fonction utilise l'individu qui lui est présenté, donc une règle, mais aussi l'image experte. Cela revient à utiliser deux images binaires $I_{règle}$ et I_{expert}, calculées de la façon suivante :

- Pour chaque pixel *p* de l'image brute, on teste si *p* satisfait la règle en question. Si c'est le cas, on met à ***1*** le pixel correspondant dans $I_{règle}$, on le met à ***0*** sinon.

- Pour chaque pixel *q* de l'image experte, on teste si *q* correspond à la classe en question, traitée par la règle. Si c'est le cas, on met à ***1*** le pixel correspondant dans I_{expert} , à ***0*** sinon.

Après de nombreuses expériences, nous avons déduit les principales caractéristiques de la fonction d'adaptation :

- Puisque c'est le principe d'une classification supervisée, cette fonction doit être basée sur une comparaison des données réelles et des données expertes pixel par pixel. Une image étant une matrice de pixel n'ayant *a priori* aucune relation entre eux, nous avons décidé que l'application de la fonction *fitness* doit être indépendante du choix d'un sens de parcours de ces pixels (donc pas de fonction progressive qui traite le pixel suivant en fonction du résultat réalisé par le pixel précédent).

- Cette fonction ne doit prendre en compte qu'une seule classe à la fois (car les classifieurs sont indépendants entre eux). Nous avons décidé que notre fonction d'adaptation sera basée sur des comparaisons de surfaces de classes entre l'expert et les résultats de notre programme (notion intuitivement exacte, mais aussi justifiée par une rapidité de calcul qui se révèle être très importante surtout dans le cadre de manipulations de pools de plusieurs centaines d'individus). Cela nous amène de manière immédiate à l'utilisation exclusive des quatre quantités résumées dans le tableau qui suit, représentant les quatre cas possibles pour un pixel donné selon sa valeur binaire dans $I_{règle}$ et I_{expert} (***00***, ***01***, ***10*** ou ***11***).

- Enfin une attention particulière devait être accordée au problème des classes sous-représentées : si une classe contient peu de pixels sur l'image experte ou après classification sur les résultats de notre programme, la note ne doit pas être perturbée par quelques pixels qui ne sont pas à leur place. Nous allons voir que ce problème a été en partie résolu par l'utilisation de *coefficients de reclassement*.

Le tableau suivant résume les quantités employées, correspondantes au cardinal de l'ensemble de pixels obtenus dans chaque cas.

Tableau 3 : Calcul de la fonction d'adaptation

		Image classifiée par la règle ($I_{règle}$)	
		Pixels valant ***1***	Pixels valant ***0***
Image classifiée par l'expert (I_{expert})	Pixels valant ***1***	$P_{\exp}^{reg}$	$P_{\exp}^{\overline{reg}}$
	Pixels valant ***0***	$P_{\overline{\exp}}^{reg}$	$P_{\overline{\exp}}^{\overline{reg}}$

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

$$N_{classe} = \frac{P_{\exp}^{reg}}{P_{\exp}^{reg} + P_{\exp}^{\overline{reg}}} \qquad \text{(Eq. 4.1)}$$

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

$$N_{\overline{classe}} = \frac{P_{\overline{\exp}}^{\overline{reg}}}{P_{\overline{\exp}}^{\overline{reg}} + P_{\overline{\exp}}^{reg}} \qquad \text{(Eq. 4.2)}$$

La moyenne de ces deux notes permet de résoudre le problème des classes sous-représentées. En effet, le déficit d'une note N_{classe} obtenue si quelques pixels ne sont pas bien classifiés par la règle sera compensé par la note $N_{\overline{classe}}$, puisque dans ce cas le dénominateur de cette dernière est beaucoup plus important.

Cependant, il est encore possible de rééquilibrer la note grâce aux coefficients de reclassement C_{classe} et $C_{\overline{classe}}$, déterminés par l'utilisateur (de manière empirique bien entendu, mais la valeur de 0,5 leurs convient apparemment parfaitement) :

$$N_{finale} = C_{classe} \cdot N_{classe} + C_{\overline{classe}} \cdot N_{\overline{classe}} \qquad \text{(Eq. 4.3)}$$

Une vérification rapide : la fonction donne bien sa valeur maximale (1) lorsque :

- tous les pixels étiquetés *X* par l'expert sont classés dans la même classe *X* par notre algorithme, et
- tous les pixels non étiquetés *X* par l'expert ne sont pas appariés à cette classe par notre programme.

Inversement, la note prend sa valeur minimale (0) dans le cas contraire.

Notre fonction *fitness* garantit donc une mesure de qualité efficace, quelque que soit la taille des classes. La note augmente lorsque les règles reclassent les pixels dans les bonnes catégories (procédure d'*affinage* des règles). Pendant l'affinage d'une règle, les notes suivent une progression continue, évitant au maximum la présence de minima locaux, si embarrassante dans la recherche d'optima globaux, ce qui accélère la convergence de l'algorithme.

4.3.2. Génération du pool initial

La génération du pool initial de la population initiale est l'une des étapes les plus importantes dans un algorithme génétique. Quelque soit le pouvoir de découverte des opérateurs génétiques, il n'est jamais bon de leur proposer un pool initial trop éloigné de la solution à trouver. Dans le cas qui nous occupe, nous avons déterminé 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 : la distribution des individus serait donc trop uniforme et les opérateurs de croisements pourraient mal réagir.

Nous avons proposé deux approches pour générer ce pool, selon deux méthodes différentes : la méthode des bornes et la méthode Spectro, que nous décrivons ci-dessous. La première présente l'avantage d'être rapide, la seconde permet de traiter certains cas plus complexes.

Chacune de ces méthodes suppose que l'on désire obtenir un pool d'individus permettant de reconnaître une classe spécifique donnée mais il est tout à fait possible d'appliquer une méthode plusieurs fois, afin de traiter toutes les classes désirées. Leur but est d'obtenir une règle initiale, qui va ensuite être dupliquée un certain nombre de fois, afin de créer le nombre d'individus désiré. Chaque règle de l'ensemble ainsi obtenu est mutée afin de constituer un pool diversifié. Enfin, une fonction de validation garantit que les règles obtenues ne sont pas *exotiques*, c'est-à-dire qu'elles ne sortent pas d'un domaine de validité préalablement fixé. Le schéma général de notre création d'un pool initial est présenté dans la figure suivante.

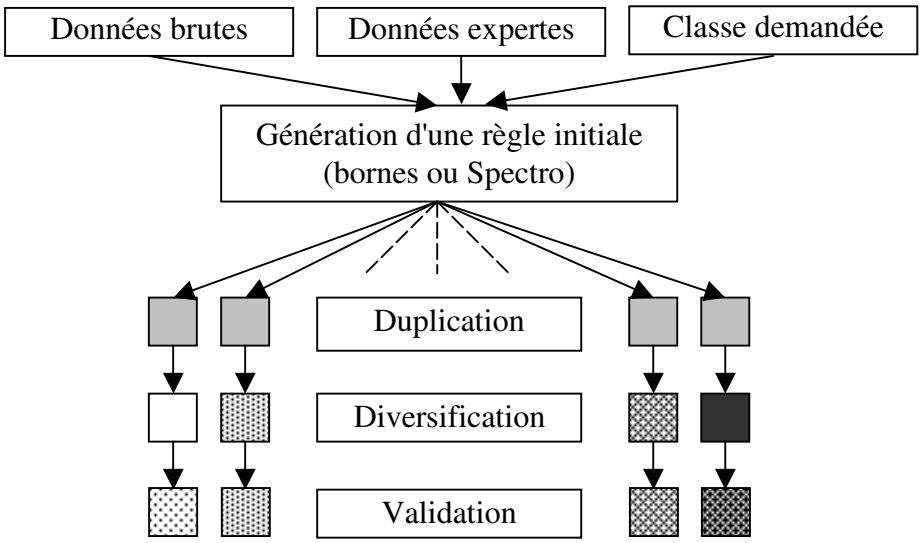


Fig. 10 : Création d'un pool initial

La duplication s'effectue par simple copie, la diversification consiste à modifier aléatoirement les deux bornes des intervalles des règles par un facteur d'échelle (basé sur la largeur des intervalles) paramétré par l'utilisateur, tandis que la validation effectue les vérifications de contraintes suivantes sur chaque intervalle [A;B] :

- A doit être inférieur ou égal à B ;
- A doit être positif ou nul ;
- B doit être inférieur ou égal à la plus grande valeur de réflectance possible (souvent 255 ou 65535).

Le nombre d'individus dupliqués est quand à lui laissé aux soins de l'utilisateur, puisque ce nombre induit fortement la durée de traitement de l'algorithme. Selon la complexité du cas à traiter et selon nos expérimentations, il faudra en effet faire varier la population d'environ 100 à 1000 individus.

Etudions maintenant les deux méthodes permettant la création d'une règle initiale.

4.3.2.1. *La méthode des bornes*

Elle consiste à comparer les données brutes avec l'expert. Elle définit pour chaque pixel appartenant à la classe demandée et pour chaque bande spectrale disponible la valeur de réflectance minimale et maximale observée sur les données brutes. Ensuite elle duplique chaque intervalle un certain nombre de fois, spécifié par l'utilisateur (mais la valeur par défaut de 5 semble parfaitement convenir dans notre cas) afin de créer un ensemble d'intervalles spectraux, définis dans la partie 3.3.1. Nous rappelons ici que ces ensembles permettent d'avoir des disjonctions de contraintes (par exemple $[10;15] \vee [23;34]$), ce qui peut se révéler important dans bien des cas.

On obtient ainsi une règle de classification grossière, classant correctement tous les pixels de la classe en question (puisque chaque pixel sera forcément dans la plage encadrée par les bornes minimale et maximale). Bien entendu des règles issues de plusieurs classes différentes se recouvriront et la méthode ne marche pas correctement si pour la même classe, l'image contient des valeurs de réflectance extrêmes. Cependant pour des images de type SPOT (faible nombre de bandes et petit domaine de définition des valeurs de réflectance) cette méthode simpliste convient parfaitement : en effet les pixels appartenant à la même classe n'ont de toute façon jamais de valeurs trop éloignées de la moyenne pour toute la classe (les calculs d'écart type ont donné des valeurs faibles).

4.3.2.2. *La méthode Spectro*

Cette méthode beaucoup plus complexe est à réserver aux images présentant des domaines de définition des valeurs de réflectance plus vaste. Le nombre de bandes n'entre pas en compte puisque chacune est analysée de manière indépendantes comme dans la première méthode. Cependant nous avons cherché à résoudre le problème des valeurs de pixels extrêmes, et à insuffler une première idée de discrimination à notre algorithme. Le principe consiste à découvrir pour une bande donnée, la répartition des valeurs de pixels, 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 à partir des données brutes et des données expertes (voir en annexe B). A partir des vecteurs de répartition représentant pour chaque bande et dans la classe voulue le nombre de pixels présents dans cette classe selon l'expert, le programme détermine des *paquets* de même densité caractérisant le spectre des pixels de cette classe. Le nombre de paquets est déterminé par l'utilisateur et équivaut au nombre de disjonctions maximales qu'il souhaite obtenir dans un ensemble d'intervalles. La figure suivante présente la répartition des valeurs d'un pixel dans une classe et une bande arbitraire, et la création de la règle originelle, selon la méthode Spectro.

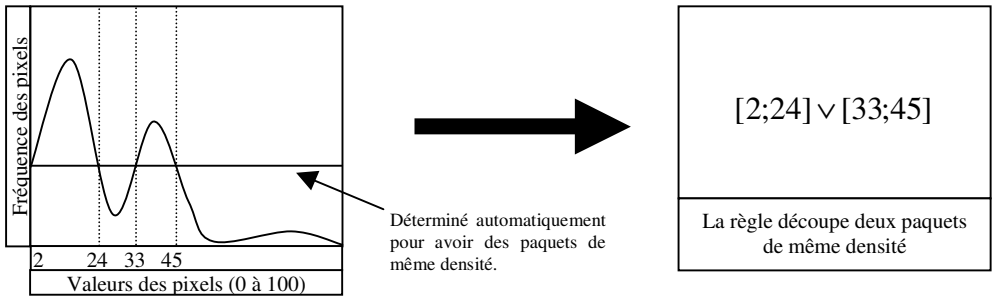


Fig. 11 : Méthode Spectro

4.3.3. Croisement des classifieurs

L'opérateur de croisement (ang. *crossover*) est utilisé afin d'exploiter au maximum les qualités des individus dont on dispose. Basé sur le fonctionnement biologique du croisement, on considère [Goldberg, 1989a] que deux bons individus croisés ont de meilleures chances de donner un individu de qualité plus élevée que celui obtenu en croisant deux mauvais. De plus, le croisement peut indirectement 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.

La création de la population des enfants passe par plusieurs phases :

- On détermine une certaine quantité Q d'individus à croiser, le reste étant recopié directement dans la population des enfants.
- Un opérateur de sélection pour la reproduction, utilisé uniquement pour la sélection des individus à croiser ou à muter, choisi les individus à croiser selon l'une ou l'autre des méthodes suivantes (voir la partie 4.3.6. pour plus de détails) : sélection selon le rang, sélection élitiste aléatoire, sélection aléatoire. Le type de cet opérateur est paramétrable par l'utilisateur.
- Après avoir choisi les individus, on détermine les gènes à croiser de chaque chromosome et on effectue l'échange. Les deux individus résultants sont ajoutés à la population suivante.

L'algorithme général est le suivant :

ALGORITHME 3
Croisement d'une population

```
_____________________________________________________________________

I et I' sont des individus (règles)
P, P' et P'' sont des populations (pool) d'individus
i et i' sont des intervalles
_____________________________________________________________________

P' := { }
répéter
      I := SELECTION_REPRODUCTION(P)              // Individu à croiser
      I' := SELECTION_REPRODUCTION(P)
      pour chaque bande b
            si ALEA() < PROBA_BANDE_CROISEE
                  i_b := CHOIX_INTERV(I)          // Choix aléatoire d'un intervalle
                  i'_b := CHOIX_INTERV(I')
                  I := (I \ i_b) U i'_b           // Echange de i et i' dans la bande b
                  I' := (I' \ i'_b) U i_b
            fin si
      fin pour
      VALIDATION(I, I')                           // Vérification des individus
      P' = P' U { I, I' }
tant que TAILLE(P') < Q
répéter
      P' = P' U SELECTION_RECOPIE(P)
tant que TAILLE(P') <> TAILLE(P)
EVALUATION(P')                                    // Calcul de la fonction fitness
```

Remarques sur les paramètres définis par l'utilisateur :

- La quantité Q d'individus à croiser. Beaucoup comme [Goldberg, 1989b] ou [Michalewicz, 1996] considèrent que 80% est une valeur acceptable, mais elle varie souvent entre 70% et 100%.
- La probabilité de croisement d'une bande donnée : si elle est trop forte, l'algorithme effectue trop de modifications à la fois et la fonction d'adaptation ne peut pas, par le truchement de la note, distinguer les bonnes modifications des mauvaises. On considère que des modifications *pas à pas* des individus sont beaucoup plus fructueuses, ce qui revient à ce que par défaut l'algorithme ne modifie qu'une seule bande par croisement (et ne modifie qu'un seul intervalle dans la bande choisie). Avec une probabilité plus forte, on simule un *croisement uniforme*.
- Le type de sélection de l'individu à croiser.
- Le type de sélection de l'individu à recopier.

La figure 12 présente sur un exemple concret, le mécanisme de l'opérateur de croisement, sur deux individus représentant deux règles à 4 bandes.

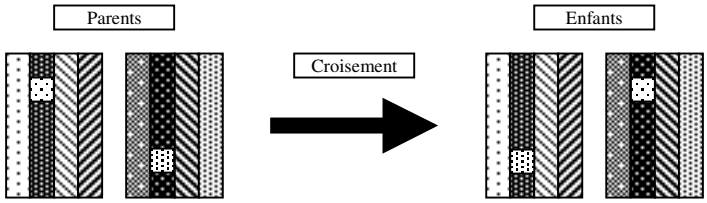


Fig. 12 : Opérateur de croisement

Les bandes échangées sont les mêmes, donc l'opération est concevable. 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 13 le résultat du croisement pour une bande donnée de deux règles.

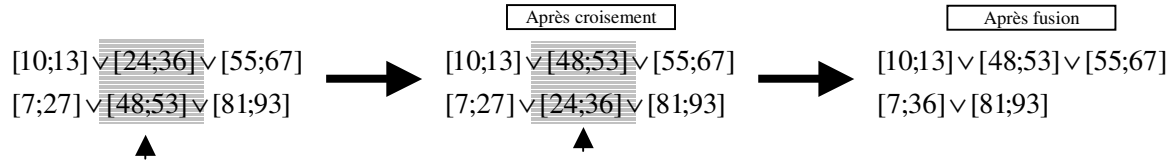


Fig. 13 : Fusion après un croisement

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 peut être important pour la génération suivante, de conserver deux intervalles distincts. Dans notre exemple, une petite modification (voir l'opérateur de mutation dans la partie suivante) de la borne 27 en 22 conduira peut être à améliorer la note de la règle, en éliminant la valeur 23 (on obtiendra $[7;22]\vee[24;36]\vee[81;93]$), alors qu'une fusion est une opération irréversible : la règle risque d'être éliminée prématurément, sans avoir eu le temps de démontrer son potentiel. 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. Lorsqu'on échange de tels intervalles, il est possible que l'efficacité de l'opérateur de croisement soit complètement annulé par la présence de ces liaisons. L'opération de fusion accentue encore plus la dépendance de deux intervalles entre eux (par exemple si A est lié avec B et sont deux intervalles disjoints, la fusion de A avec C lie automati-

quement B à C). C'est pourquoi elle a donc été désactivée par défaut, puisque des expérimentations ont de surcroît montré une perte de performance.

4.3.4. Mutation des classifieurs

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 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. Outre la capacité d'exploration, cet unique opérateur qui en fait en comprend plusieurs, permet de *corriger* les erreurs commises par l'opérateur de croisement, grâce à la grande finesse des modifications qu'il procure. Ainsi écraser la valeur d'une borne par une autre aléatoire est de façon évidente un procédé bien trop grossier pour être applicable. Par contre des opérations de plus haut niveau comme des déplacements d'intervalles sont beaucoup plus justifiables. Nous montrons, dans la figure 14, les différents *niveaux* d'opérateurs que nous proposons, représentés sous forme d'un arbre.

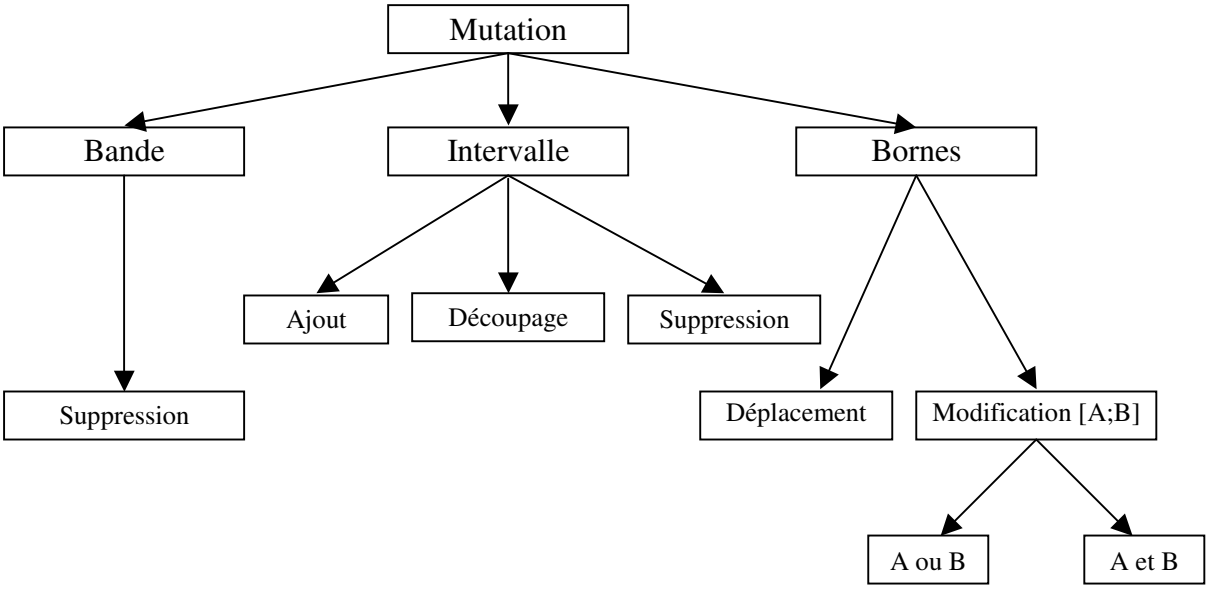


Fig. 14 : Opérateur de mutation

La mutation ne concerne qu'un et un seul objet de la règle à la fois : soit 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 l'algorithme de tester les modifications les unes après les autres et à la fonction *fitness* de privilégier le type de modification et l'effet correspondant au cas par cas. Cependant il est possible, bien que cela ne soit pas le cas par défaut, de demander la mutation de plusieurs bandes en même temps, en définissant la probabilité qu'une bande donnée soit sélectionnée.

Comment la mutation opère-t-elle, après l'opérateur de croisement ? Une fois la nouvelle population obtenue après une certaine quantité de croisements et de recopies, l'algorithme détermine un nombre d'individus relativement faible (de l'ordre de 1 à 5%, paramétrable par l'utilisateur) qu'il va muter et réinsérer dans le pool génétique. Nous verrons plus loin comment fonctionne exactement l'opérateur de sélection pour la reproduction, dont le rôle est de choisir l'individu à muter (le même est utilisé dans le croisement). Il faut juste ajouter que

ce choix ne dépend absolument pas de l'origine de l'individu (croisement ou recopie) : tout individu a donc une certaine chance d'être sélectionné.

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 (voir à ce propos la partie 3.2.1.3.). Les bandes bruitées sont éliminées parce qu'elles n'apportent aucune information utile pour discriminer une classe, mais il faut souligner que toute bande non bruitée et non porteuse d'une information importante est de la même façon retirée de la règle. 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.

  Un paramètre permet de contrôler la probabilité qu'une bande donnée puisse subir une mutation. Lorsque l'opérateur choisi de supprimer une bande, au moins une bande est sélectionnée sinon l'opération risquerait d'être superflue.

- 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, mais dont la largeur peut être paramétrée par l'utilisateur, en pourcentage de la valeur maximale possible. La suppression est elle aussi aléatoire puisqu'on ne connaît pas à priori l'intervalle qui pourrait défavoriser la note finale (nous nous plaçons toujours dans le cadre d'un apprentissage par renforcement, donc la note n'est connue qu'après évaluation globale de la règle). Le découpage d'un intervalle se fait à partir d'un point aléatoire prélevé dans l'intervalle (par exemple le découpage de [10;100] peut donner [10;14] et [15;100]) : cette mutation permet d'éliminer en plusieurs temps une ou plusieurs portions contiguës ou non à l'intérieur d'un intervalle. Le découpage puis la suppression permet donc de cibler exactement le tube spectral dans lequel les valeurs spectrales des pixels peuvent discriminer une certaine classe.

- La modification ponctuelle d'une ou deux bornes d'un intervalle (ponctuelle car elle n'agit que sur un unique intervalle d'une règle qui peut en comprendre plusieurs centaines) prolonge et affine l'idée de ciblage réalisée par les autres types de mutation. Elle peut sembler un peu plus complexe, mais il faut avoir à l'esprit qu'une modification brutale des deux bornes surchargerai de hasard un opérateur qui en comporte déjà suffisamment. Nous avons donc proposé de le compléter par un déplacement (ang. shift) aléatoire dont l'amplitude (petite ou forte) est paramétrable par l'utilisateur, tout comme l'amplitude de la modification d'une borne.

Suite à la présentation de cet opérateur, un lecteur avisé remarquera certainement le déséquilibre présent au niveau de la suppression de bande : aucune modification ne permet d'ajouter les bandes supprimées. En réalité, cette opération n'aurait pas de sens. La suppression n'est pas réalisée exactement comme on pourrait le croire : en fait, on remplace dans la règle les intervalles affectés à cette bande par un seul intervalle, mais de taille maximale, ce qui conduit à considérer n'importe quelle valeur comme activant la règle. Le rétablissement de l'usage de cette bande se fait naturellement en découpant cet intervalle maximal, puis en employant l'un des opérateurs de mutation déjà vus. Malgré tout, une certaine instabilité subsiste : par ces suppressions, l'algorithme simplifie les règles à long terme, ce qui l'amène vers

une généralisation parfois non souhaitée par l'utilisateur. On lui réserve donc la possibilité de paramétrer de manière très fine la fréquence d'apparition de chaque opération présentée dans la Fig. 12 : chaque flèche peut en fait être valuée par une probabilité de déclenchement. L'utilisateur 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. Sans que les paramètres aient besoin d'être ajustés à la perfection, c'est la puissance du programme qui s'adapte aux différents cas, souvent présents en même temps dans un même problème.

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 déjà décrite dans la partie 4.3.2. La fusion des intervalles est aussi proposée à l'utilisateur après une mutation, mais nous ne la conseillons pas, là encore pour des raisons déjà évoquées plus haut.

4.3.5. Recyclage générationnel

Il existe de nombreuses définitions et interprétations de ce que peut être un recyclage générationnel. Nous le considérons ici comme une procédure visant simplement à produire la génération suivante à partir de la génération des parents et de celle des enfants obtenue après les croisements et les mutations (P et P" dans l'algorithme 1). Il existe plusieurs stratégies pour produire le pool final qui survivra à la génération suivante, mais nous utilisons principalement les deux suivantes (le choix peut se décider avant de lancer l'algorithme) :

- La stratégie révolutionnaire. On n'utilise que la population des enfants pour produire le pool final :

Fig. 15 : Stratégie révolutionnaire

- La stratégie préservative. Plus intéressante, elle conserve certains individus de la population des parents (l'ensemble d'individus de départ qui n'ont jamais été modifiés durant la génération) :

Fig. 16 : Stratégie préservative

Souvent, les meilleurs individus de l'ancienne population remplace les plus mauvais de la population modifiée (élitisme). Cette stratégie permet d'être certain que les individus dont la qualité est maximale passent directement à la population suivante. Cependant elle présente le risque de voir des individus perdurer à travers les âges : ceci n'est a priori pas gênant, sauf dans le cas d'un pool génétique faible avec quelques individus de performances moyennes qui bénéficieraient alors d'une immunité bien discutable. Une solution serait d'affecter à chaque individu une durée de vie (ang. *time to live*) qui éliminerait alors les individus trop anciens. Une autre est de sélectionner l'individu à garder non plus en fonction d'un critère élitiste, mais en introduisant une

proportion de hasard. Nous proposons dans la partie suivante plusieurs solutions de sélection d'individus. L'utilisateur peut alors choisir le pourcentage d'individus à éliminer ou à garder et sa stratégie préférée pour réaliser le tri.

4.3.6. Sélection des classifieurs

La séparation de l'algorithme génétique en couches (croisement, mutation, ...) nous est apparu comme une nécessité, pour offrir à l'utilisateur un fort pouvoir de modularité. 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 recyclage générationnel qui reprend certains individus de la population des ancêtres et d'autres de la dernière population générée. Cette vision peu fréquente et plutôt nouvelle de l'implémentation d'un algorithme génétique permet une plus grande flexibilité. Nous proposons ainsi une forme de généralisation et d'abstraction des algorithmes actuels qui, il faut le souligner, sont très hétérogènes et présentent de nombreuses disparités sur les principes employés.

Chaque couche a besoin de matériel (les individus) qu'elle va piocher dans la couche précédente. Il est donc intéressant d'avoir des fonctions de sélection d'individus, différentes selon le type de la couche sollicitant ces individus. La sélection est donc un problème à part entière, et l'utilisateur pourra soit choisir les relations couche/fonction-de-sélection qui lui semblent les mieux adaptées à chacun des cas, dans un *catalogue* de fonctions prêtes à l'emploi, soit définir lui-même une telle fonction. Les travaux de [Blickle, 1995] ont permis d'étudier et de justifier formellement de nombreuses méthodes de sélection, dont nous nous inspirons ici.

Nous distinguons plusieurs 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 la reproduction est utilisée pour le choix d'un individu qui va être croisé ou muté. Les stratégies retenues sont : le rang, l'élitisme aléatoire et le hasard.

- La sélection pour la recopie sélectionne les individus complétant le pool génétique après avoir effectué ces croisements. Les stratégies retenues sont : le rang, l'élitisme aléatoire, l'élitisme pur et le hasard.

- La sélection de l'individu à garder choisi dans l'ancien pool génétique l'individu qui sera conservé pour le cycle générationnel suivant. Les stratégies retenues sont : le rang, l'élitisme pur et le hasard.

- La sélection de l'individu à éliminer choisi dans le nouveau pool génétique (individus modifiés) les enfants à éliminer du cycle générationnel suivant. Le recyclage se passe donc de la manière suivante (Fig. 17) :

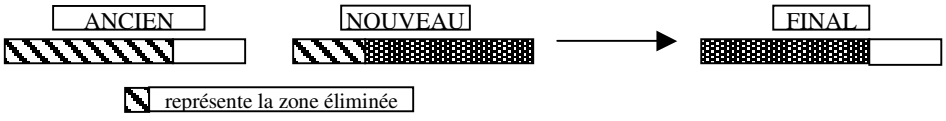


Fig. 17 : Sélection de l'individu à éliminer

Les stratégies retenues sont : le rang, l'eugénisme pur et le hasard.

4.3.6.1. *Sélection selon le rang*

La sélection selon le rang est une méthode basée sur l'ordre d'un individu dans un pool ordonné par leurs notes. Les notes proposées par la fonction d'adaptation (ang. *fitness*) ne sont qu'une représentation relative de la qualité d'un individu par rapport à ses congénères. 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. La figure 18 présente l'explication de cette stratégie, comparée à celle de la roue de la roulette.

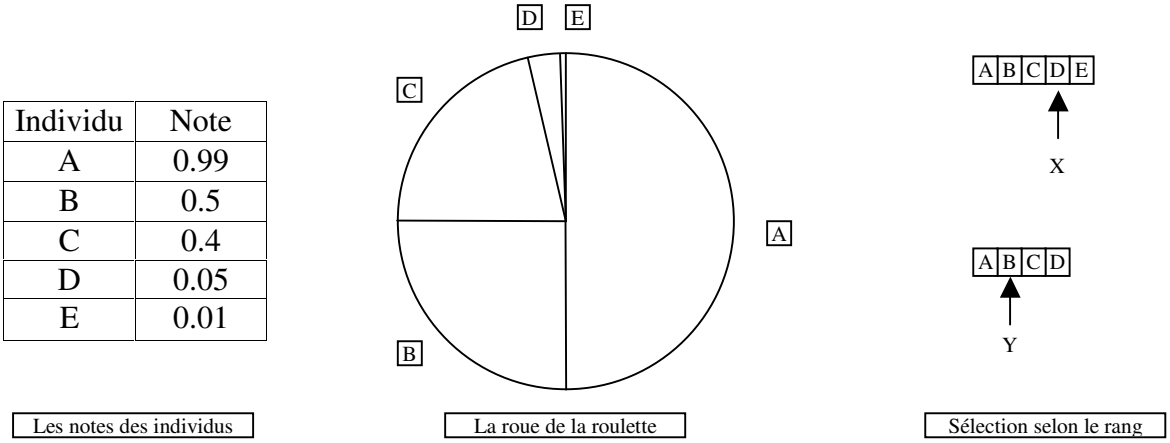


Fig. 18 : Sélection en fonction de la note

La roulette correspond au tirage aléatoire des individus dans une sorte de camembert dont chaque part a une taille proportionnelle à la note de l'individu concerné, et donc chaque probabilité de sélection est directement liée à la note.

La sélection par le rang 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.

Un individu très faible n'est donc pas éliminé. De plus la fonction de qualité n'a pas pour but de produire une note impartiale (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 leurs chances de sélection soient comparables, ce qui est difficile à garantir dans la pratique), mais simplement de trier les notes entre elles (on laisse donc à chaque fonction génétique son propre rôle).

Dans le cadre de la sélection par le rang, la probabilité de sélection n'est plus une fonction de la note, mais du rang lui-même. Etudions-la :

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 :

$$P_M(\{X = i\}) = \frac{1}{M} \qquad \text{(Eq. 4.4)}$$

Soit $P'_M(\{Y=i\})$ la probabilité qu'en choisissant Y selon la procédure décrite ci-dessus, on obtienne i. On a :

$$P'_M(\{Y=i\}) = \sum_{j=i}^{M} \left[P_M(\{X=j\}) \cdot P_j(\{Y=i\}) \right] \qquad \text{(Eq. 4.5)}$$

En effet, il faut que $1 \le i \le X \le M$. Pour tous les choix 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 :

$$P'_M(\{Y=i\}) = \sum_{j=i}^{M} \left[\frac{1}{M} \cdot \frac{1}{j} \right] = \frac{1}{M} \cdot \sum_{j=i}^{M} \frac{1}{j} \qquad \text{(Eq. 4.6)}$$

Pour l'exemple, le choix du meilleur individu dans une population de 100 individus se calcule ainsi :

$$P'_{100}(\{Y=1\}) = \frac{1}{100} \cdot \sum_{j=1}^{100} \frac{1}{j} = 0{,}0519 \qquad \text{(Eq. 4.7)}$$

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%, ce qui est très grand, mais la méthode est plutôt conçue pour les cas réels d'au moins 100 à 200 individus.

4.3.6.2. *Elitisme*

Cette stratégie consiste à garantir que le ou les meilleurs individus seront toujours sélectionnés. L'élitisme pur inclus les N meilleurs individus dans le pool suivant, tandis que l'eugénisme pur permet de garantir l'élimination les N plus mauvais individus. Dans le cas de l'élitisme pur, supposons que l'on dispose d'une liste d'individus triés par leurs notes, le premier étant le meilleur. La probabilité que l'individu X choisi à l'étape q porte le numéro i, avec $1 \le q \le N$ dépend du nombre d'individus nécessaires N et s'exprime de la façon suivante :

$$P_q(\{X=i\}) = \begin{cases} 1 \text{ si } i = q \\ 0 \text{ sinon} \end{cases} \quad \text{avec} \sum P_q(\{X=i\}) = 1 \qquad \text{(Eq. 4.8)}$$

Quelle que soit l'étape et pour N tirages, on a :

$$P(\{X=i\}) = \begin{cases} 1 \text{ si } i \le N \\ 0 \text{ sinon} \end{cases} \quad \text{avec} \sum P(\{X=i\}) = N \qquad \text{(Eq. 4.9)}$$

Cette méthode ne peut pas être utilisée pour la reproduction (comment croiserait-on par exemple les 11 meilleurs individus ou comment déterminerait-on des couples d'individus obtenus par ce système de tirage sans remise ?) car elle suppose qu'un même individu ne peut être sélectionné deux fois. Elle n'est donc utilisée que pour la recopie ou le recyclage. Nous proposons donc une autre méthode de sélection, permettant d'éliminer les plus mauvais, mais en ajoutant une certaine quantité de hasard pour éliminer le déterminisme de la sélection pure. L'élitisme aléatoire consiste à choisir au hasard et avec remise N individus parmi les M meil-

leurs individus du pool. Dans ses conditions, la probabilité que l'individu *X* sélectionné à l'étape *q* porte le numéro *i* est la suivante :

$$P_q(\{X = i\}) = \begin{cases} \frac{1}{M} \text{ si } i \leq M \\ 0 \text{ sinon} \end{cases} \quad \text{(Eq. 4.10)}$$

Soit *S* une variable aléatoire égale au nombre de sélections de l'individu *X*. Sur *N* tirages, la probabilité qu'il soit sélectionné *k* fois obéit à la loi binomiale de Bernoulli :

$$P(S = k) = C_N^k \left(\frac{1}{M}\right)^k \left(1 - \frac{1}{M}\right)^{N-k} \quad \text{(Eq. 4.11)}$$

On en déduit la probabilité que l'individu *X* de numéro *i* a d'être sélectionné au moins une fois lorsqu'on tire *N* individus au hasard parmi les *M* meilleurs du pool :

$$P(\{X = i\}) = \sum_{k=1}^{N} C_N^k \left(\frac{1}{M}\right)^k \left(1 - \frac{1}{M}\right)^{N-k} \quad \text{(Eq. 4.12)}$$

L'utilisateur décide du paramètre *M* par l'intermédiaire d'un certain pourcentage de la taille totale du pool. Cette stratégie ne fixe plus de limitation pour le nombre de tirages tout en privilégiant les individus de tête.

4.3.7. Convergence de l'algorithme

Aussi appelé *terminaison* (ang. *fitness threshold*), elle s'appuie sur un test permettant de savoir s'il faut arrêter les générations : 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. Nous allons détailler plusieurs critères utilisés dans notre système de classifieurs. Ces critères sont inclusifs, c'est-à-dire qu'il suffit qu'il y en ait un seul qui s'active pour que le programme soit stoppé.

Tout d'abord l'obtention du meilleur individu possible peut se vérifier par sa note : si de manière absolue celle-ci atteint un certain seuil proche de 1 (par exemple 0.99), il y a peu de chance d'avoir un individu encore plus performant : nous demandons donc à l'algorithme de s'arrêter.

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. Nous utilisons 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, nous mettons 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 :

$$\left| \frac{\sum_{k=1}^{P} Q_k}{P} - Q_0 \right| \leq E \qquad \text{(Eq. 4.13)}$$

où P représente la longueur maximale d'un palier et E la variation maximale de ce palier par rapport à la note courante : plus E est petit, plus le palier devra être constant afin de pouvoir être détecté.

Enfin un contrôle beaucoup plus direct sur la durée de l'exécution s'effectue par le choix d'un nombre maximum de générations à calculer. Cette limite nous donne l'assurance que l'algorithme se terminera dans tous les cas.

L'utilisateur possède le contrôle de la totalité de cette paramétrisation. Des valeurs par défaut lui sont proposées, en accord avec nos observations : le palier doit varier entre 10 et 20 générations, selon la fiabilité que l'on désire pour le test de stabilité de la solution. La variation quand à elle peut se satisfaire d'une valeur très faible (10^{-4}) voire nulle, car les quantités discrètes qui ont permis de calculer la performance d'un individu rendent aisément l'évolution des notes constante. Enfin, le nombre de générations ne devrait pas excéder 200, même dans les problèmes les plus complexes.

4.4. Conclusion

Ce chapitre était réservé à une approche plus pragmatique de notre système de classifieurs et de son évolution à travers l'algorithmique génétique, après l'étude théorique de nos règles présentée dans le chapitre 3. Nous avons donc spécifié un certain nombre d'opérateurs adaptés qui d'une part manipulaient ces règles pour en trouver de meilleures, et qui d'autre part s'assuraient que le processus se termine correctement (fonction de convergence) avec la plus performante d'entre elle (par l'utilisation de notre fonction *fitness*). Il ne reste plus qu'à appliquer le tout sur des exemples concrets. Le chapitre suivant expose la mise en œuvre de tous les opérateurs, les paramétrages et les remarques que nous venons d'exposer afin de démontrer la puissance de nos classifieurs dans des cas réels.

5. ETUDE DE CAS

5.1. Introduction

L'étude effectuée au sein de ce stage a permis de développer une méthodologie robuste et originale, basée sur des systèmes de classification dans un domaine où ils n'ont encore jamais été testés. Comme toute approche originale, elle doit être validée par la pratique, d'où la réalisation d'un prototype qui a été utilisé pour tous nos tests.

Ce qui pouvait être utile aux géographes, écologues, urbanistes et à toute personne ayant besoin d'images classifiées, est un outil mettant en œuvre de façon pratique et simple toute la théorie développée pendant nos travaux. Pour apprécier pleinement le travail accompli, ce chapitre s'articule autour de deux grandes parties. La première présente cet outil ainsi que certaines des fonctionnalités annexes qui ont pu lui être ajoutées et qui facilitent le prétraitement. La seconde expose trois exemples, choisis pour leur intérêt scientifique et de difficulté croissante. Nous présentons en premier lieu un cas pédagogique, afin d'introduire le lecteur au procédé utilisé. Le second et le troisième exemple sont plus complexes, afin de pousser les limites de notre prototype au maximum et de le valider dans des situations intéressantes.

La programmation et les tests ont été effectués sur une machine milieu de gamme, possédant un processeur AMD Duron cadencé à 800 MHz, ainsi que 380 Mo de mémoire vive. Des tests supplémentaires ont nécessité l'emploi de deux machines de plus haute gamme, dotées de processeurs AMD Athlon de 1,2 GHz et de 256 Mo de mémoire vive chacune.

5.2. Prototype

5.2.1. Une approche rigoureuse

Le prototype développé dans notre équipe, nommé *ICU* pour *I See You*, a été spécialement conçu pour répondre aux obstacles spécifiques rencontrés durant ce stage. En effet, il n'y a que peu ou pas d'outils existants dans notre équipe ou sur l'Internet, adaptés à notre approche. Nous nous sommes donc contraint à tout refaire à neuf, proposant ainsi une réponse plus ciblée au problème posé.

Le C++, que nous avons employé, est un langage permettant de faire de la *conception objet*. Sans entrer dans les détails, ce type de programmation possède de nombreuses qualités, dont nous avons tiré profit :

- L'*extensibilité*, autorisant de manière aisée l'intégration de nouvelles spécifications. Une règle est définie en interne comme un objet avec ses structures et ses opérations propres. Il est donc entièrement possible de rajouter les modules correspondants à d'autres types de règles, selon les besoins de l'utilisateur. Une règle est donc un objet générique avec des fonctions communes (mutation, évaluation, ...), spécialisées pour gérer les particularités de chaque règle dérivée. Du point de vue du programme, ces objets sont les mêmes quels que soit la syntaxe exacte des règles. On peut donc faci-

lement en rajouter sans devoir modifier la structure interne des algorithmes qui les gèrent.

- La *correction* ou la *validité*, garantissant que le logiciel effectue bien les tâches qu'on lui a demandé. L'implémentation de notre logiciel passe donc par la programmation de multiples objets, encapsulant chacun une notion fondamentale : que ce soit pour une règle, un classifieur, ou la lecture d'un fichier, chaque objet possède ses *attributs* (paramètres permettant de le contrôler) et ses *opérations* propres (comme l'évaluation d'une règle par exemple). Il est ainsi plus facile de contrôler de manière indépendante les accès à ces objets et de garantir que leur fonctionnement suivra les spécifications qu'on en a faites.

- Enfin la *robustesse*, qui est l'un des critères les plus ardus à respecter : le logiciel doit se comporter correctement même dans des conditions *anormales* (non prévues par la spécification). La gestion des erreurs en C++ par blocs permet à notre logiciel de détecter toutes les situations non prévues et de réagir en conséquence, ou dans le pire des cas, de prévenir l'utilisateur du disfonctionnement.

5.2.2. Une interface orientée vers l'utilisateur

La plupart des utilisateurs qui ont besoin des résultats de classification que nous produisons les manipulent sous les environnements OS X (Macintosh) ou Windows. Le style d'interface MDI (Multiple Documents Interface) qui a été retenu est un classique dans l'environnement Windows. Le programme s'articule autour d'une fenêtre principale, permettant l'édition et le traitement simultané de plusieurs documents. La figure 19 présente cette fenêtre, contenant quelques images venant d'être chargées.

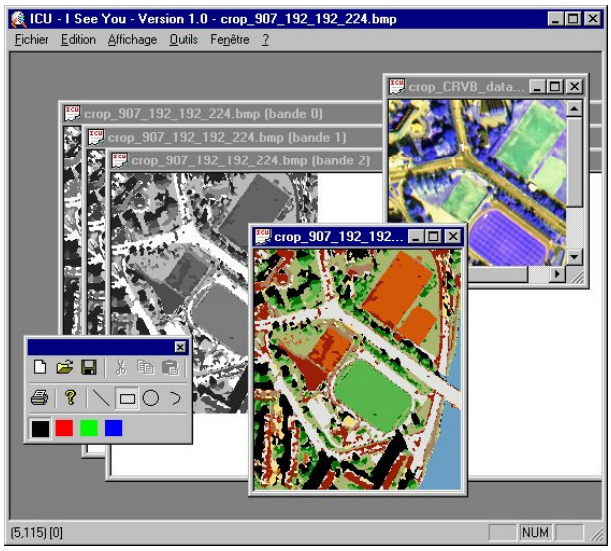
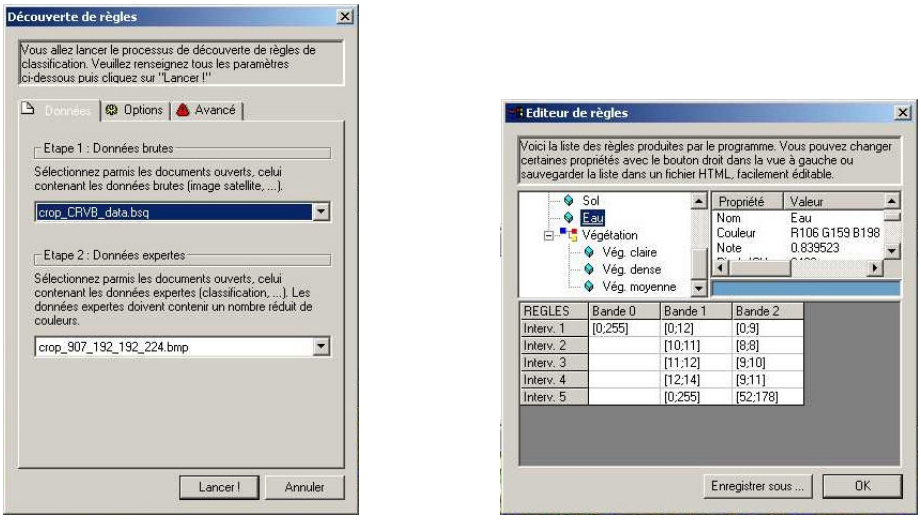


Fig. 19 : Accueil du logiciel

L'utilisateur peut, à l'aide du menu **Fichier** ou par glisser-déposer, importer des images satellites (de format BSQ/HDR) ou des images expertes (de format BMP/PPM/PGM). Ces formats sont aussi pris en charge pour l'export des différents produits intermédiaires ou finaux, générés par le programme.

La convivialité de l'interface permet une prise en main rapide et prend en charge les quatre grandes fonctionnalités du prototype que sont la découverte des règles de description des classes thématiques d'images de différentes tailles et résolutions, la visualisation des règles extraites de ces images, leur manipulation en vue d'en classifier d'autres, ainsi qu'une phase de pré-traitements ou de post-traitements permettant entre autres de traiter les données avant l'analyse et de faciliter ensuite les validations visuelles.

De plus, ce programme a été conçu pour effectuer en quelques clics la totalité des traitements depuis l'ouverture des fichiers jusqu'à l'obtention des classifications finales, grâce au menu **Outils**, qui ne contient d'ailleurs qu'une seule proposition utile. Elle affiche une boîte, visible sur la figure 20 en (a), permettant de lancer l'apprentissage. Après avoir chargé ses images, l'utilisateur n'a plus qu'à sélectionner le nom de l'image brute et celui de l'image experte dans les listes déroulantes prévues à cet effet, puis à cliquer sur **Lancer**. Le traitement est multitâche, ce qui autorise pendant ce temps l'utilisation d'autres fonctionnalités du logiciel dont des traitements qui vont permettre ensuite la validation des classifications obtenues (par exemple l'affichage de l'indice de végétation). Durant l'apprentissage, une barre de progression informe en permanence sur l'avancée de l'opération, avec la prévision du temps restant. Est aussi affichée la performance actuelle du meilleur classifieur calculé, ce qui donne la possibilité à l'utilisateur soit d'annuler le traitement, soit de changer n'importe lequel des paramètres, en cours d'apprentissage (par exemple pour l'accélérer en diminuant le nombre d'individus).



(a) (b)

Fig. 20 : Apprentissage et édition des règles de classification

L'apprentissage terminé, la boîte d'information en (b) permet l'édition, la visualisation et le contrôle des règles obtenues. Des menus contextuels permettent alors :

- la sauvegarde des règles dans un format très répandu, le HTML, qui est facile à utiliser, qui est importable dans beaucoup de logiciel notamment Excel et Access, et qui permet la visualisation de chaque règle avec leurs performances et leurs codes couleur sous forme de pastilles colorées,
- la classification d'autres images satellites grâce aux règles de classification générées,
- l'affichage de cartes de différences entre notre classification et l'image experte, sortes de matrices de confusion graphiques permettant de tester la validité de nos règles.

D’autres fonctionnalités viennent compléter l'usage de base du programme, afin d'aider davantage l'utilisateur. Nous n'allons pas toutes les décrire ici, mais elles sont relativement nombreuses, parmi lesquelles :

- le découpage spatial et spectral de l'image pour produire des extraits plus appropriés aux tests,
- l'égalisation d'histogramme pour l'affichage des images de manière réaliste,
- l'affichage simultané de plages de bandes sous forme de vignettes (détection des bandes bruitées),
- la création de spectrogramme (voir l'annexe B),
- le calcul d'indices de végétation ou de brillance (voir la partie 3.3.1.),
- la pipette graphique permettant de se promener sur l'image avec la souris et de connaître le spectre de chacun des pixels,
- le zoom pour afficher de très grandes ou de très petites images, ...

5.3. Expérimentations

5.3.1. Introduction

Nous avons eu à notre disposition un ensemble assez conséquent de données (voir la partie 3.2.) ce qui nous a offert une certaine possibilité de choix. Ces données couvrent à la fois un espace assez vaste (60 km de région pour l'image hyperspectrale de Strasbourg) mais représentent aussi une abondance de types d'informations différents : pléthore de canaux, richesse de leurs contenus, dimensions et résolutions variées, qualités hétérogènes (bruit, ...).

Il est évident que nous n'avons pas tout utilisé, bien au contraire. Cependant le florilège de renseignements à notre disposition nous est apparu comme un avantage : nous allons pouvoir tester nos classifieurs et comparer nos résultats sur des données de types variés.

Notre démarche consista donc à définir une zone dans chacune de ces images dont le lieu et l'orientation restent inchangés. L'explication peut se résumer en plusieurs points :

- Tout d'abord, cela permet une meilleure comparaison de l'efficacité de la phase d'apprentissage. Dans l'analyse de nos résultats, il y aura une distinction plus nette entre la performance de nos classifieurs, qui opérera de la même façon sur chaque image, et les qualités de celles-ci qui ne seront pas constantes. En bref, prendre des zones différentes sur des images de types différents ajoute à la confusion.

- Ensuite, cela permet une meilleure validation lors de la phase de test. Les données étant sensiblement les mêmes, la même quantité d'eau, de terre, ... sera étudiée par notre prototype dans chaque cas. La comparaison s'en trouve donc beaucoup plus rigoureuse.

- Enfin, à force de travailler sur les mêmes données, on finit pas s'y habituer, ce qui nous évite à chaque fois une phase d'adaptation afin de *comprendre* l'environnement étudié. Ainsi certaines incohérences entre notre intuition et les classifications obtenues sont immédiatement détectées et corrigées si elles s'avèrent venir du prototype. En ou-

tre, nous en avons profité pour faire directement nous-même une validation sur le terrain.

A côté de chaque exemple qui va suivre, nous présentons les principaux paramètres (ou ceux qui ont pu changer d'un cas à l'autre) de l'algorithme utilisé. Les autres paramètres sont initialisés avec des valeurs par défaut, peu intéressantes à discuter ici. Ce sont des paramètres de moindre influence, qui n'agissent pas directement sur la qualité de la reconnaissance, néanmoins, nous les avons ajusté durant les expérimentations pour les rendre optimaux. Dans cette étude, les critères discutés seront essentiellement la qualité de l'apprentissage et l'apport de notre système de classifieurs dans le travail de classification d'images.

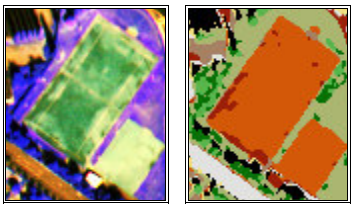
Parmi toutes nos expérimentations (nous avons aussi étudié des régions de Venise), nous en avons retenus trois qui apparaissaient comme importantes, de la plus simple à la plus complexe. Le premier exemple, de dimensions assez réduites, présente une zone relativement homogène. Il nous permettra d'avoir une idée concrète du traitement effectué sur les images de télédétection. Le deuxième exemple expose une image de dimensions et de nombre de classes plus élevés. Nous y étudierons en particulier les problèmes de bordures dûs aux combinaisons de classes mitoyennes plus nombreuses. Enfin, nous testerons, grâce au dernier exemple, la robustesse de nos classifieurs face à la complexité de l'expression des règles générées par l'augmentation de la taille spectrale de l'image.

5.3.2. Terrains d'entraînement du Stade Vauban

5.3.2.1. *Présentation*

Cet exemple, choisis pour des raisons pédagogiques, illustre la capacité de notre prototype à déduire des règles de classification à partir de deux images qu'on lui présente : celle des données brutes contenant les valeurs de réflectance telles qu'elles ont été captées par le satellite, et celle des données expertes fournissant une information préalable sur la disposition des classes. Chaque pixel de l'image experte représente la classe du pixel correspondant sur les données brutes, sous la forme d'une couleur choisie par un expert du domaine (voir par exemple le code couleur du tableau 2 de la partie 3.2.2.) et caractéristique de cette classe.

Les images de la figure suivante représentent un petit extrait des données que nous avions à notre disposition et dont nous avons parlé dans la partie 3.2. L'image en (Fig. 21a) a été acquise lors d'essais des capteurs du satellite SPOT4 sur Strasbourg, embarqués sur un avion volant à haute altitude, ce qui explique une résolution élevée (1,30 m x 1,30 m par pixel au sol) par rapport à la résolution habituelle de ce type de satellite (10 à 20 m). Sa taille de 80 x 96 pixels permet d'apercevoir la zone du Stade Vauban, proche du Quartier des Quinze et du Quartier Rotterdam. Cette photo très sombre a été traitée avant l'impression par une égalisation d'histogramme afin d'en apprécier toutes les (fausses) couleurs. L'image en (Fig. 21b) a été fournie par une experte géographe (Anne Puissant), travaillant au laboratoire Image et Ville de Strasbourg ([NET 9]). Elle est parfaitement synchronisée avec les données brutes puisque ces dernières ont été directement utilisées pour la produire, par des méthodes de segmentation automatique.



(a) (b)

Fig. 21 : Images brute et experte des terrains d'entraînement du Stade Vauban

L'algorithme a été lancé avec les paramètres suivants :

Tableau 4 : Paramètres pour l'étude de cas n° 1

Paramètre	Valeur	Signification
NbIndividus	1000	1000 règles vont évoluer simultanément, et la meilleure d'entre elles sera à chaque fois gardée pour la population de règles suivante.
PerformanceMin	1,00	L'erreur souhaitée est minimale (performance maximale), le critère du palier sera utilisé pour détecter l'arrêt de l'algorithme.
LongueurPalier	20	Si la performance n'évolue pratiquement pas durant 20 cycles, l'algorithme est stoppé.
DiffPalierMin	10^{-4}	
PourCroisement	80 %	80 % d'individus seront croisés (voir la partie 4.3.3.)
PourMutation	5 %	5 % d'individus seront mutés (voir la partie 4.3.4.)
PourcElimNouv	1 %	1 % des meilleurs individus subsisteront (voir 4.3.6.)

5.3.2.2. *Résultats*

Tableau 5 : Récapitulatif de l'étude de cas n° 1

Durée du traitement (temps CPU)	32 min., 36 s.
Nombre de classes apprises	9
Performance générale	90,17 % de pixels correctement classifiés
Performance pour la classe *ombre*	97,93 %

Le tableau suivant présente les règles trouvées pour discriminer les différentes classes. Ces règles ont été simplifiées par une opération de fusion (voir l'équation 3.4 dans la partie 3.3.1.) afin d'obtenir une information claire pour l'utilisateur.

Tableau 6 : Classification des terrains d'entraînement du Stade Vauban

COULEUR	SIGNIFICATION	REGLE	PERFORMANCE
	OMBRE	*[0;255]* [7;22] [3;32]	97,9 %
	VEGETATION DENSE	[18;41] [0;26] *[0;255]*	88,3 %
	PELOUSE	*[0;255]* [4;38] [70;158]	93,5 %
	ARBUSTES	[2;87] [21;40] [43;83]	87,9 %
	TOIT TUILE	*[0;255]* [30;150] [31;56]	69,1 %
	TOIT / TÔLE	*[0;255]* [36;91] [37;58]	93,7 %
	SOL GRAVIER	*[0;255]* [21;44] [32;57]	78,2 %
	SOL BETON	[11;40] [15;36] *[0;255]*	78,9 %
	ROUTE	*[0;255]* [16;52] [17;34]	96,4 %

Pour prendre un exemple, tout pixel de l'image brute dont les valeurs spectrales des bandes XS1, XS2 et XS3 sont respectivement situées dans les intervalles [2;87], [21;40] et [43;83] appartiendra à la classe des arbustes (plus exactement de la végétation moyenne). La performance correspondante est de 88 %, c'est-à-dire que 88 % des pixels sont classifiés de la même façon par l'expert (voir à ce sujet le paragraphe sur la fonction *fitness* en 4.3.1.).

Les règles obtenues peuvent ensuite être appliquées, dans une phase dite de *prédiction*, à une image de notre choix, contenant les valeurs de réflectance des pixels que l'on veut traiter. Le schéma suivant permet de saisir les phases de traitement des images.

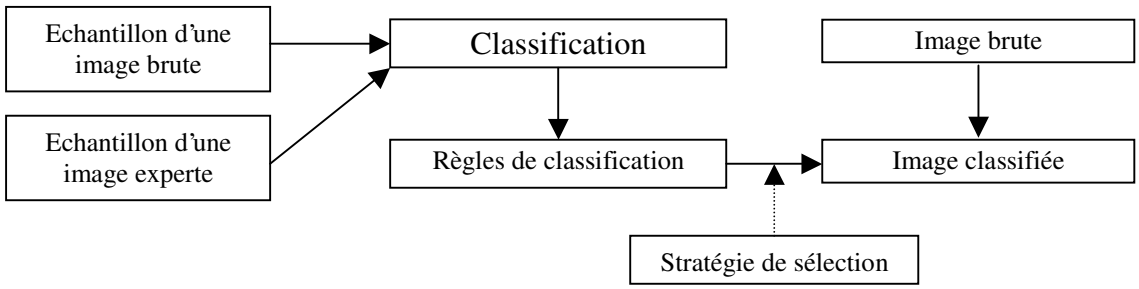
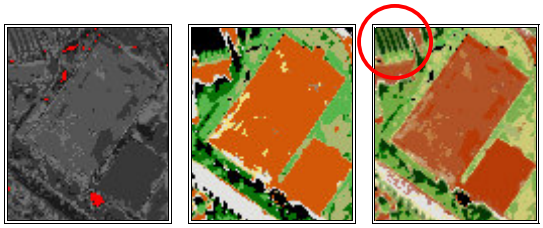


Fig. 22 : Chaîne de traitement des images

De par leur nature, les règles ne découpent pas forcément l'espace spectral, dans lequel les pixels de l'image brute vient prendre ses valeurs, en un ensemble de classes parfaitement disjointes les unes des autres. Une stratégie de sélection est alors nécessaire pour choisir les règles qui pourraient se recouvrir. Nous proposons trois stratégies dans notre prototype : la première consiste à choisir la classe dont la règle correspondante obtient la performance maximale. La seconde, utilisée pour l'étude de cas suivante, consiste à considérer l'écart entre les valeurs des pixels et les centres des intervalles des règles. Enfin la troisième n'est pas réellement une stratégie de sélection en tant que telle : elle consiste à afficher la teinte moyenne des couleurs des classes sélectionnée par la règle. Cependant nous verrons qu'elle peut être utile pour déterminer par superposition graphique si deux classes différentes sont en fait semblables ou s'il faut au contraire créer une classe supplémentaire pour distinguer les pixels qui sont simultanément dans ces deux classes.

Puisque nous sommes dans notre exemple pédagogique sur terrains d'entraînement, observons la classification de l'image qui a été utilisée pour l'apprentissage.

La figure 23 présente en (a) une image symbolisant le degré de recouvrement des règles pour chaque pixel de l'image. Les pixels rouges indiquent les endroits qui ne sont recouverts par aucune règle, et en dégradé de gris les pixels recouverts par plusieurs règles, depuis le noir (une seule règle) au blanc pur (toutes les règles). Il est évident que ce dernier cas ne se présente jamais, on rencontre expérimentalement parfois trois, voire très rarement quatre règles qui se recouvrent. L'image en (b) présente la classification que l'on obtient par la sélection de la meilleure performance. L'image en (c) présente une carte moyennant les teintes des classes lors d'un recouvrement.



(a) (b) (c)

Fig. 23 : Classifications des terrains d'entraînement du Stade Vauban

5.3.2.3. *Discussion*

Cet exemple est beaucoup trop simple pour en tirer des conclusions valables. Néanmoins, il est sans doute nécessaire d'éclaircir quelques points intéressants.

Après avoir utilisé notre prototype pour classifier ses images, l'utilisateur s'aperçoit très vite de la durée des traitements, même lorsqu'ils sont appliqués à des images considérées comme étant de petite taille. En fait, ils sont effectivement et seront souvent relativement longs, comparés à des algorithmes de segmentation ou de classification non supervisée. Nous tenons bien évidemment à nous en expliquer.

Tout d'abord une image ne sera *jamais* de petite taille : un échantillon comprend souvent en moyenne plusieurs centaines de milliers de pixels. Bien évidemment, effectuer un échantillonnage réduirait énormément la durée de calcul, et ce n'est pas la difficulté de sa programmation qui nous l'interdirait. En fait, nous avons choisi de considérer tous les pixels durant l'apprentissage pour des raisons de qualités (il nous fallait des résultats qui ne dépendent pas de pixels écartés).

Ensuite, le prototype manque sérieusement d'optimisations, car là n'était pas le but de notre projet. Il effectue plusieurs tris intermédiaires, utiles pour certaines stratégies de sélection d'individus (voir la partie 4.3.6.) mêmes si elles ne sont pas choisies par l'algorithme. De plus il a une tendance assez nette à insister sur les opérations de pré et post conditions afin de vérifier la bonne tenue des règles obtenues (voir la partie 4.3.2.).

Enfin cette lenteur est fortement liée à l'algorithmique génétique, dont le principe est de créer plusieurs centaines d'individus par génération (donc une centaine de millier étudiés au total), pour n'en sélectionner qu'un au final. Malgré un chapitre entier consacré au sujet, nous en profitons ici pour en rappeler l'intérêt. Notre objectif est de produire des règles de classification, et non pas une image classifiée. Ces règles ne doivent pas dépendre d'une structure imposée par l'algorithme (par exemple un réseau de neurones) mais doit pouvoir être déterminée par l'utilisateur ayant sa propre idée sur la question. Durant notre projet, nous avons choisi des intervalles, mais l'algorithme proposé fonctionnerait parfaitement avec une toute autre description (relations impliquant les pixels voisins, ...). On voit ici toute la subtilité des algorithmes génétiques : créer ces intervalles à l'aide de formules déterministes ou non, adaptées à la représentation imposée par l'utilisateur est impossible directement ou exige une étude dédiée à chacun des cas. Notre algorithme n'en a cure : il gère la cohabitation de chaque règle (notamment les problèmes de recouvrement) en fonction de leur structure et de surcroît, le tout supervisé par l'image experte.

Un coup d'œil à la performance générale obtenue (plus de 90%) nous confirmera que nous avons choisi des paramètres adaptés à un traitement efficace et non bâclé. Les notes sont effectivement bonnes pour chaque classe, particulièrement pour la classe ombre qui pose souvent quelques soucis aux géographes. Notre prototype reproduit fidèlement durant la phase de

test l'image d'apprentissage, avec même l'intention de l'améliorer : les rangées de pins, visibles lors de notre validation directement sur le terrain, sont réapparus sur l'image moyennant les teintes (Fig. 23 (c)). Les règles obtenues recouvrent la quasi-totalité de l'image classifiée, et les différents éléments du paysage (route, végétation, terrain de football) se détachent correctement.

Enfin il faut noter que l'utilisation des 3 bandes de SPOT4 n'est pas essentielle pour discriminer toutes les classes. D'après le Tableau 6, huit intervalles (représentés en italique) couvrent la largeur maximale du domaine de définition ([0;255], qu'on aurait pu représenter par un joker) et n'ont donc pas été jugés utiles, du moins à partir de l'extrait présenté au programme.

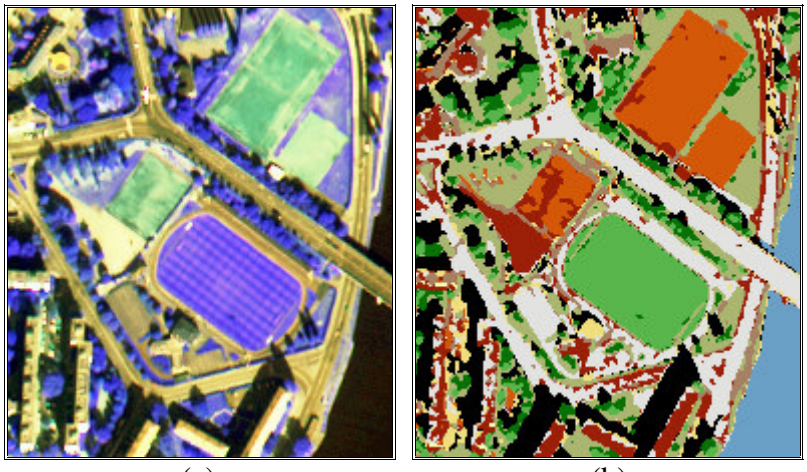
Passons maintenant à une étude plus approfondie du Stade Vauban, avec une image légèrement plus complexe.

5.3.3. Stade Vauban complet et alentours

5.3.3.1. *Présentation*

Cet exemple est plus complexe que celui vu auparavant. Nous avons choisi de garder la même zone, afin de pouvoir les comparer, cependant nous avons légèrement augmenté sa taille, pour inclure des pixels de toutes les classes présentes dans l'image experte, notamment les classes d'eau et d'ombre qui comptent parmi les plus difficiles à discriminer. La complexité vient du fait que l'image possède des dimensions conséquentes (192 x 224 pixels, soit plus de 40 000 pixels) avec de grandes chances d'avoir la présence de pixels bruités et de pixels mixtes qui viennent perturber l'apprentissage.

La figure 24 présente en (a) une représentation traitée pour l'impression des données brutes du satellite SPOT, et en (b) la partie extraite de l'image experte utilisée par notre système de classifieurs comme données d'apprentissage.



(a) (b)

Fig. 24 : Images brute et experte du Stade Vauban complet

L'algorithme a été lancé avec les paramètres suivants :

Tableau 7 : Paramètres pour l'étude de cas n° 2

Paramètre	Valeur	Signification
NbIndividus	1500	1500 règles vont évoluer simultanément, et la meilleure d'entre elles sera à chaque fois gardée pour la population de règles suivante.
NbGenerations	200	On limite l'apprentissage à 200 générations maximum, dans le cas où le critère du palier serait trop long.
LongueurPalier	20	Si la performance n'évolue pratiquement pas durant 20 cycles, l'algorithme est stoppé.
DiffPalierMin	10^{-4}	
PourCroisement	80%	80% d'individus seront croisés (voir la partie 4.3.3.)
PourMutation	5%	5% d'individus seront mutés (voir la partie 4.3.4.)
PourcElimNouv	1%	1% des meilleurs individus subsisteront (voir 4.3.6.)

5.3.3.2. *Résultats*

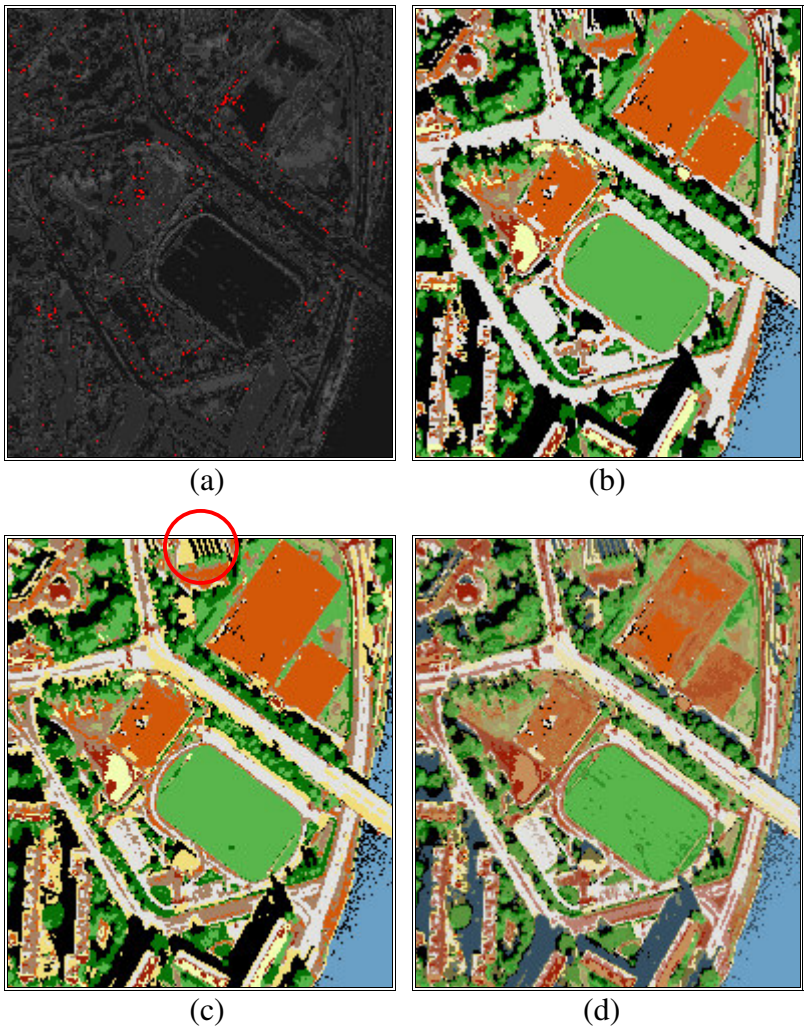
Tableau 8 : Récapitulatif de l'étude de cas n° 2

Durée du traitement (temps CPU)	1 j., 16 h., 12 min., 29 s.
Nombre de classes apprises	11
Performance générale	90,74 % de pixels correctement classifiés
Performance pour la classe *eau*	95,58 %
Performance pour la classe *ombre*	95,98 %
Meilleure classe reconnue	SOL BETON
Plus mauvaise classe reconnue	ARBUSTES (qualité tout de même très satisfaisante)

Tableau 9 : Classification du Stade Vauban complet

COULEUR	SIGNIFICATION	REGLE	PERFORMANCE
	OMBRE	[8;38] [0;21] [10;35]	96,0 %
	EAU	[15;37] [9;40] [1;15]	95,6 %
	VEGETATION DENSE	[14;78] [0;25] [31;87]	92,3 %
	PELOUSE	[21;52] [12;46] [70;160]	97,0 %
	ARBUSTES	*[0;255]* [15;43] [42;77]	81,0 %
	TOIT TUILE	[45;255] [0;200] *[0;255]*	84,9 %
	TOIT / TÔLE	[0;48] [36;100] [36;67]	94,5 %
	TOIT BETON	[1;37] [15;66] [10;34]	92,7 %
	SOL GRAVIER	[35;61] [30;79] [33;68]	85,8 %
	SOL BETON	[43;183] [45;255] [52;184]	98,0 %
	ROUTE	[31;56] [15 ;43] [7 ;35]	92,7 %

Les onze règles obtenues peuvent ensuite être appliquée sur l'image d'apprentissage. La figure 25 montre en (a) le degré de recouvrement des règles et en (b) et (c) les classifications obtenues grâce aux méthodes de sélection des règles, respectivement selon la meilleure note et selon le centre des intervalles. Enfin, en (d) nous présentons une classification moyennant les teintes des classes superposées.



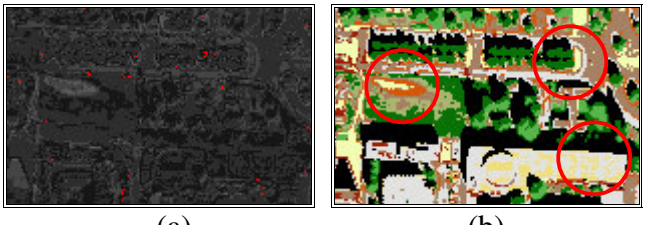
(a) (b)

(c) (d)

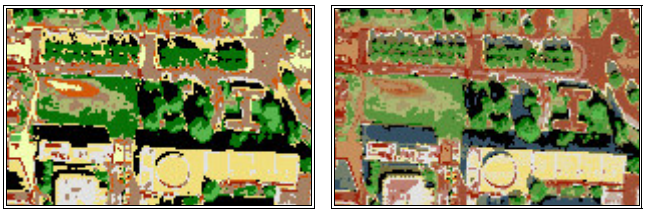
Fig. 25 : Classifications du Stade Vauban complet

L'image observée en (c) se calcule de la façon suivante : soit *P*, le vecteur spectral d'un pixel et *C*, le vecteur spectral des centres des intervalles d'une règle *i* (moyennes des bornes de chaque intervalle). La règle sélectionnée sera tout simplement celle dont l'écart type entre *P* et *C* est minimal.

Tout cela est très intéressant, mais nous ne nous sommes pas simplement contenté de l'image d'apprentissage. Observons maintenant l'application de ces règles de classification sur une zone représentant le parking étudiant rue René Descartes à l'Esplanade. La figure suivante présente les mêmes types de résultats qu'auparavant (respectivement la carte de recouvrement, les stratégies de sélection selon la meilleure note et les centres des intervalles, puis la carte moyennant les teintes).



(a) (b)



(c) (d)

Fig. 26 : Classifications du parking étudiant de l'Esplanade

5.3.3.3. *Discussion*

L'image présentée ici est plus complexe comme nous l'avons dit dans notre courte introduction. Elle est plutôt variée : les classes définies par l'expert cohabitent chacune avec une multitude d'autres classes, l'introduction de données sémantiques sur le voisinage telle qu'on la pratique actuellement n'est plus réellement adéquate (l'assertion "*pas d'eau autour d'une route*" est par exemple infirmée par la présence d'un pont sur la Fig. 25). Nous n'avons malheureusement pas eu le temps de proposer nos propres techniques. De plus, la taille des classes oscille entre petits détails et grands aplats. Les méthodes n'utilisant uniquement que des informations géométriques peuvent également échouer. Malheureusement, la présence de détails est l'un des inconvénients des images haute résolution car elle apporte une information parfois contradictoire à l'algorithme d'apprentissage : le nombre de classes données par l'expert est insuffisant pour reconnaître la tôle d'une voiture rouge, d'une jaune, d'une bleue, les planches d'un banc public, ... De toute façon aucun expert n'ira jusque là. Nombreux sont donc les pixels incorrectement classifiés par ce dernier (et il n'y est pour rien) qui perturbe fortement la classification.

Certaines remarques observées dans l'étude précédente restent donc valables ici. Le traitement est toujours aussi long : en outre il faut savoir que pour des raisons de rapidité, nous avons lancé plusieurs apprentissages sur la même machine, donc le chronométrage est quelque peu biaisé. Les résultats quand à eux sont très bons : près de 91 % de pixels sont correctement classifiés dans toute l'image, avec une performance exceptionnelle pour l'eau et l'ombre. Nous rappelons ici que l'image experte utilisée a été corrigée à la main car les classes *eau* et *ombre* étaient confondues. Notre prototype les discrimine sans intervention humaine.

L'introduction, par l'expert, de deux classes supplémentaires (l'eau et les toits en béton) se remarque immédiatement dans la base de règles obtenue : les règles sont plus spécifiques, peu d'intervalles sont éliminés. Elles sont malgré tout ressemblantes à celles trouvées dans la première étude de cas. Des données plus conséquentes ont donc permis d'affiner cette base et de réviser notre jugement des bandes inutiles. Cependant une comparaison classe à classe montre qu'en moyenne les performances ont augmentées, tout en sachant qu'obtenir une bonne qualité d'apprentissage sur une image simple est plus facile. Ces règles respectent la proportion des valeurs de réflectance mise en évidence par le spectrogramme (voir l'annexe B) de la figure suivante. On remarque que l'eau (colonne 11) a un spectre très précis, traduit par des intervalles assez courts, tandis que le sol en béton a un spectre beaucoup plus diffus (colonne 10).

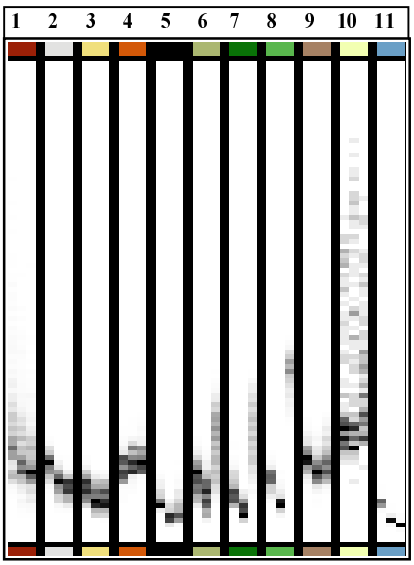


Fig. 27 : Spectrogramme de la zone du Stade Vauban

Vérifions ensuite les résultats obtenus sur l'ensemble d'apprentissage (Fig. 25). D'une part les pixels non classifiés par les règles sont extrêmement rares. Pour les inclure toute de même dans une classe quelconque, il faudrait dilater les intervalles de ces règles or il est sans doute impossible pour le modèle de les décrire sans immerger d'autres pixels qui ne sont pas de cette même classe et qui n'ont rien à y faire. D'autre part, les classifications obtenues en (b) et en (c) sont sensiblement identiques : l'ombre des immeubles et celle du pont, classe mitoyenne de l'eau, est particulièrement bien rendue. Cependant la stratégie de la meilleure note (b) se révèle être bien plus efficace, notamment pour les routes. En effet, en (c), d'autres types de sols (gravier et béton) viennent perturber cette classe. Toutefois la stratégie employée en (c) permet la distinction de l'ombre des pins dans la partie supérieure centrale de l'image. De plus, l'utilisation des centres des intervalles permet de sélectionner une règle dans les zones non recouvertes, et ceci par une méthode statistiquement exacte : les pixels proches d'un intervalle donné ont de grandes chances d'y être inclus.

Cette complémentarité entre les deux types de sélection s'observe sur l'image de test utilisée que l'algorithme d'apprentissage n'a pas connue. La reconnaissance est très bonne (peu de pixels oubliés et une relation pratiquement bijective entre les pixels de l'image experte et les règles : elles recouvrent toutes des zones mutuellement disjointes). En (b) la reconnaissance de la route longeant le parking étudiant, ainsi que le découpage des arbres sont très nets. Par contre le toit de la Faculté des Sciences Humaines, en bas au milieu, pose quelques difficultés à (b) qui le considère comme une zone ouverte aux automobiles (blanche pour la classe *route*). Notre stratégie en (c) résoud le problème (il s'agit bien d'un toit en béton). La carte des teintes moyennes en (d) est capable de distinguer visiblement la plupart des classes.

Cependant il faudrait noter une petite aberration au niveau du terrain vague situé sous le parking (qui a été transformé depuis en un terrain de football). Cette erreur est sans doute due à la présence d'un terrain de type différent (flaques après une pluie, sol argileux ou boues) qui n'existait pas dans les données d'apprentissage, d'où l'incapacité de reconnaissance. Pourtant l'algorithme a deviné qu'il s'est trompé : la zone correspondante sur la carte de recouvrement en (a) est légèrement plus claire. Il suffira de reclassifier cette zone, en y envoyant un expert ou en réitérant une classification non supervisée, puis de l'inclure dans les données d'apprentissage.

Bien que cette image fût déjà très complexe, nous avons décidé de traiter une image beaucoup plus grosse, pour mettre en valeur la capacité de traitement de notre logiciel.

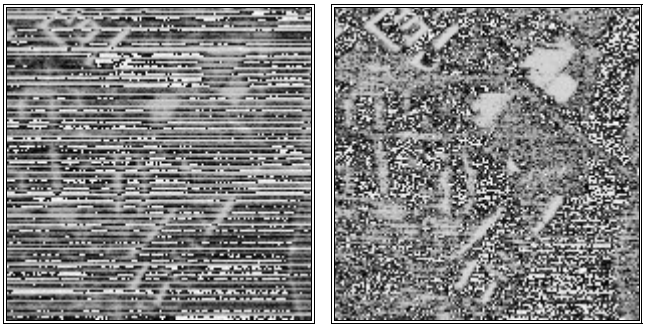
5.3.4. Le cas hyperspectral

5.3.4.1. Présentation

Le format des données étudiées ici est relativement impressionnant, puisqu'elles sont extraites d'une image représentant une fauchée au sol d'environ 60 kilomètres de longueur et de 10 kilomètres de largeur. Ces données sont produites par un satellite dit *hyperspectral* car il échantillonne le spectre des pixels sur 79 canaux, embrassant les longueurs d'onde de 0,498 à 12,668 μm.

La figure suivante présente en (a) l'extrait de l'image hyperspectrale (traitée pour l'impression) utilisée pour l'apprentissage de la classification prévue par l'expert en (b). L'image experte a été produite à l'aide d'une classification non supervisée par k-moyennes (ang. *K-Means*). Dans le chapitre 3, nous avions écrit que nous retirons la plupart des bandes de l'image, pour des raisons de bruit. Cependant nous nous sommes permis de modifier cette image uniquement pour des raisons d'affichage. Les résultats présentés ici ont été obtenus à partir de la totalité de l'image, dont les 34 bandes bruitées parmi lesquelles les bandes 42 (c) et 59 (d). L'image (c) comporte un bruitage technique, dû à un dysfonctionnement du capteur pour cette longueur d'onde précise, et le bruit observé en (d) est dû à la mauvaise pénétration dans l'atmosphère des signaux envoyés par les émetteurs : la notion de *fenêtre spectrale*, bien connue des experts en télédétection, exprime les limites de capture du spectre imposées par les interférences atmosphériques.

(a) (b)



(c) (d)

Fig. 28 : Matériel utilisé pour l'étude de cas n° 3

L'algorithme a été lancé avec les paramètres suivants :

Tableau 10 : Paramètres pour l'étude de cas n° 3

Paramètre	Valeur	Signification
NbIndividus	1500	1500 règles vont évoluer simultanément, et la meilleure d'entre elles sera à chaque fois gardée pour la population de règles suivante.
NbGenerations	250	On limite l'apprentissage à 250 générations maximum, dans le cas où le critère du palier serait trop long.
LongueurPalier	20	Si la performance n'évolue pratiquement pas durant 20 cycles, l'algorithme est stoppé.
DiffPalierMin	10^{-4}	
PourCroisement	80%	80% d'individus seront croisés (voir la partie 4.3.3.)
PourMutation	5%	5% d'individus seront mutés (voir la partie 4.3.4.)
PourcElimNouv	1%	1% des meilleurs individus subsisteront (voir 4.3.6.)

5.3.4.2. *Résultats*

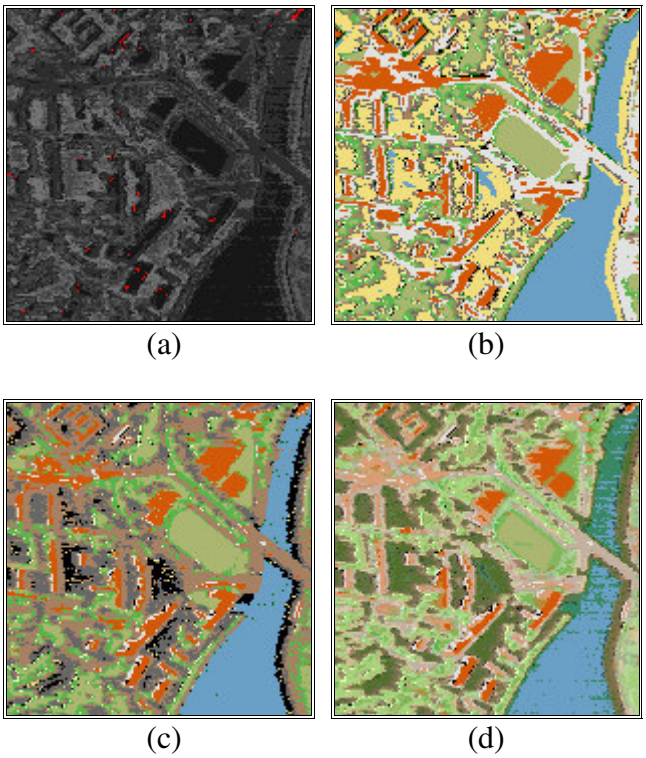
Tableau 11 : Récapitulatif de l'étude de cas n° 3

Durée du traitement (temps CPU)	2 j., 1 h., 9 min., 17 s.
Nombre de classes apprises	11
Performance générale	86,06 % de pixels correctement classifiés
Performance pour la classe *eau*	98,54 %
Performance pour la classe *ombre*	86,59 %
Meilleure classe reconnue	EAU
Plus mauvaise classe reconnue	SOL GRAVIER, 76% (peu présente)

Mentionner la totalité des règles obtenues, comprenant chacune 79 conditions n'aurait pas de sens ici. Les intervalles définis par l'algorithme sont de la même forme que ceux vus auparavant, avec un domaine de définition plus large (non plus 8 mais 16 bits). Voici pour l'exemple les intervalles spectraux I_0, I_{42} et I_{59} permettant de caractériser un pixel de végétation dense, sur les bandes 0, 42 et 59 (sachant que seule la bande 0 n'est pas bruitée) :

$$I_0 = [860;1327]$$
$$I_{42} = [0;65535]$$
$$I_{59} = [0;65535]$$

Toutes ces règles obtenues peuvent alors être appliquée sur l'image d'apprentissage. La figure suivante montre en (a) le degré de recouvrement des règles et en (b) et (c) les classifications obtenues selon la méthode de sélection des règles, respectivement en considérant la meilleure note ou le centre des intervalles. Enfin, en (d) nous présentons une classification moyennant les teintes des classes superposées.



(a) (b)

(c) (d)

Fig. 29 : Classifications du Stade Vauban en hyperspectral

5.3.4.3. *Discussion*

L'approche que nous avons emprunté ici est tout à fait différente que dans les deux premières études de cas. En effet, nous n'avons pas eut à notre disposition des classifications expertes suffisamment acceptables pour notre apprentissage. Les experts interrogés nous ont répondu que l'image était de bien trop mauvaise qualité. Nous nous sommes donc rabattu sur des classifications automatiques, réalisées au sein même de notre équipe. Le principe de l'algorithme des k-moyennes est d'agréger les pixels sous forme de *cluster*s, dont le centre est la moyenne de ses membres. Les centroïdes sont recalculés après chaque itération, qui peuvent alors incorporer de nouveaux membres. Nous conseillons au lecteur de se reporter à l'annexe A pour découvrir d'autres méthodes non supervisées. La méthode implantée dans le cadre des travaux sur Samarah, décrits dans [Wemmert, 2000], nous convenait parfaitement.

Le principal intérêt de cet exemple est d'illustrer la robustesse de nos classifieurs face à un bruit réel (et non de laboratoire), intégré à l'image dès son processus de fabrication et qu'il est donc très difficile de supprimer. Une première remarque s'impose : comme dans l'étude précédente, la présence de bandes éliminées est tout aussi visible dans les règles, concordant exactement avec les bandes bruitées. Ensuite, il ne faut pas négliger la légère baisse de la performance générale de nos règles de classification. Elle s'explique pourtant : pour certains pixels, il y a une mauvaise adéquation entre la réalité hyperspectrale, et l'image experte produite par des outils automatiques, dont le rendu est loin de l'expertise dite de "vérité terrain" dans laquelle chaque pixel est vérifié. Il est clair que ce processus très long et très coûteux ne peut être appliqué dans notre cas, qui par sa complexité aurait tout de même nécessité quelques attentions particulières. Cette mauvaise adéquation ne dénote pas uniquement la déficience des procédés de segmentation classiques, mais aussi, plus grave, l'incapacité de ce type de règles à modéliser correctement ce type de données. L'exploration de l'espace de

définition des règles par l'algorithme génétique inclus l'exploration de minima et maxima locaux, mais malgré tous ces efforts, il n'a pas été possible de construire des règles satisfaisantes, au vu de l'expert. Il faut alors songer à changer de représentation, ce qui sera - nous l'espérons - l'objet de futurs travaux.

Expliquons nous sur de telles conclusions : en observant de près la Fig. 29 (a), on remarque qu'il y a très peu de pixels non classifiés. Par contre, beaucoup de règles se recouvrent, gâchant la classification : on éprouve là les limites de la structure de nos règles. Le temps nous a malheureusement cruellement manqué, à ce stade, pour en définir d'autres.

En réalité la situation est loin d'être aussi dramatique. L'eau tout d'abord apparaît comme une étendue unie, contrairement à ce que laisse apercevoir l'examen attentif des bandes de l'image prises une par une ou ce que révèlent d'autres tests de classification non supervisée. Ces examens trahissent sur les données brutes, une multitude d'ondelettes, visibles sur la Fig. 29 (a), correspondant en réalité à des variations assez fortes des valeurs de réflectance dans cette région. Ceci explique pourquoi l'algorithme noie certaines classes, en tenant d'unifier l'amplitude des valeurs de la classe *eau*. Enfin, on remarque que les classifieurs permettent toujours un découpage correct des détails d'une image : nous pouvons voir distinctement en (b) le tracé plutôt fin des routes et des terrains d'entraînement, ainsi que les différents éléments des immeubles : au milieu les toits, puis les murs en béton, les allées d'accès et les haies environnantes. Ces détails nous confirment que nos classifieurs intègrent parfaitement la multiplicité des classes apprises.

5.4. Conclusion

Par ces différents exemples, nous avons illustré la capacité de nos classifieurs à intégrer des données à la fois hétérogènes et complexes (dimensions, nombre de bandes). L'apprentissage est sans doute plutôt long, mais cela vient de la taille des images, de l'algorithmique génétique, et de notre désir de privilégier des paramètres plus *sûrs* que *rapides*. Les résultats obtenus sont d'ailleurs assez bons, au moins par rapport à l'expertise à notre disposition, et peuvent conduire dans certains cas à leur amélioration (voir l'ombre des pins de la Fig. 25 (c)) : ceci est dû à la capacité de notre structure de règle à découper précisément l'espace spectral des pixels.

Néanmoins, nous avons vu que le cas hyperspectral révélait quelques limitations à cette structuration : nous pensons que des règles fondées sur des centres spectraux entourés de nuages de probabilité captant tel ou tel pixel seraient plus à même de modéliser ce type d'image. La structure utilisée a été malgré tout capable de créer des règles mutuellement exclusives et maximales, classifiant la totalité des pixels, alors que cette condition n'est pas évidente à produire car elle n'est pas décrite explicitement dans le modèle.

Enfin nous terminerons par citer la bonne corrélation entre les résultats que nous obtenons et certaines études statistiques de l'image (spectrogramme, exclusion des bandes bruitées sans qu'il a été nécessaire de les signaler). La réalité du terrain, même implicite, est donc reproduite fidèlement dans les règles de classification conçues par notre algorithme.

6. CONCLUSIONS ET PERSPECTIVES

Travaux et contributions scientifiques

Au cours de notre projet, nous avons mené à bien une étude des concepts entrant d'une part dans la définition d'un système capable de découvrir des règles décrivant les classes thématiques contenues dans une image de télédétection et d'autre part, dans la définition d'un mécanisme d'intégration de ces connaissances dans un système de classifieurs. Ces règles sont à la fois pratiques et simples pour l'utilisateur, et suffisamment génériques pour permettant de réitérer le processus de nombreuses fois, sans avoir à recommencer l'apprentissage. A cette intention, nous avons validé certaines étapes clés.

La première étape consistait en l'étude théorique d'un classifieur spécialisé dans le traitement d'images hyperspectrales permettant d'intégrer toutes les informations nécessaires au traitement de données d'une complexité aussi importante. Pour obtenir un tel classifieur adapté à ce domaine d'application, il a fallu imaginer une structure capable d'apporter une solution efficace au problème posé, tout en gardant une certaine simplicité (ergonomie, résultats présentés, etc.) vis-à-vis de l'utilisateur. Nous avons alors considéré différents formalismes de règles, depuis les représentations binaires jusqu'aux représentations acceptant des valeurs réelles, que nous avons retenues. Nous avons donc conçu des règles symboliques à partir de conjonctions de disjonctions d'intervalles, garantissant du même coup l'uniformisation des valeurs spectrales rencontrées. Puis nous avons montré que les fonctions de base de ces règles de classification étaient à même de saisir les particularités de l'environnement étudié. La représentation des classifieurs, quant à elle, n'est pas une structure plate, mais une hiérarchie de classifieurs indépendants.

Ces classifieurs, nouveaux dans leur genre, nécessitaient l'emploi d'un système de brassage adéquat. La deuxième étape nous amena donc à étendre les systèmes de classification habituels pour réaliser deux objectifs. D'une part, il fallait y intégrer un algorithme génétique permettant un apprentissage plus performant et plus robuste, dans un milieu souvent hostile aux classifications traditionnelles. D'autre part, il fallait étudier puis réaliser des opérateurs adaptés à la génération de classifieurs évolutifs, à la fois pour *exploiter* ceux existants (opérations de croisements : intersection, union, découpages d'intervalles) mais encore pour *explorer* l'espace spectral de recherche (opérations de mutation : ajout, translation d'intervalles). Ces opérateurs devaient également assurer les mécanismes de convergence et de validation (fonction *fitness*) des classifieurs, susceptibles d'interagir entre eux pour délimiter les frontières des classes dans lesquelles chacun d'eux s'est spécialisé.

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 spectrales. Ensuite il autorise l'intégration future de règles plus puissantes de façon harmonieuse avec le reste du logiciel. Enfin, la prise en compte d'autres opérateurs génétiques n'y est pas limitée, comme nous l'avons montré avec les fonctions de mutation dans la partie 4.3.4.

La troisième et dernière étape était de constituer une base traduisant les informations pertinentes de ces classifieurs sous forme de règles de décision faciles à interpréter. La base de règles définie introduit une stratégie de sélection permettant la classification de pixels non traités par l'expert ou concernés par plusieurs règles, permettant ainsi l'attribution sans équi-

voque de chaque élément de l'image à la classe qui lui est le plus proche. Une fois la base construite, l'utilisateur a le moyen de visualiser ces règles, de les hiérarchiser (ajout, fusion et morcellement de nœuds) et de les exporter sous un format courant (en HTML). S'il préfère obtenir des images classifiées, un module se charge d'appliquer la base sur ses images puis de les exporter dans l'un des formats proposés.

Apports pratiques et résultats en télédétection

Les résultats présentés dans le dernier chapitre de ce rapport sont très encourageants, par rapport aux difficultés posées par le contenu très riche de telles images et par la qualité médiocre de l'expertise. Cette dernière peut en effet succomber, par son caractère automatique, aux mêmes problèmes de bruit, de pixels mixtes, de bordures mélangeant toutes les combinaisons de classes possibles, que notre système de classifieurs. Cependant nous avons prouvé, par les expérimentations, que notre système se révélait être très robuste et pouvait même dans certains cas distinguer des classes fusionnées par l'expert, comme l'ombre des pins de la Fig. 25 (c).

L'étude de cas a permis de dégager plusieurs points intéressants. La structure proposée, malgré sa simplicité et le nombre de classes, reproduit fidèlement la réalité. Les règles délaissent les bandes inexploitables et sont à la fois adaptées aux grands espaces (terrains de sport) comme aux plus petits détails (arbres, traînées d'ombres, bordures des immeubles). Des travaux récents dans notre équipe ([Novak, 2000]) tentaient de recouvrir des valeurs tridimensionnelles du spectre d'un pixel dans un seul objet mathématique (hypercube, hypersphère ou hyperellipse). Malheureusement, ceux-ci s'appliquent mal dans notre cas, le large spectre de réflectance dans le cas hyperspectral se caractérisant par de très forts pics abrupts ainsi que d'autres morceaux inutilisables (bruités). Notre structure, quand à elle, permet la modélisation d'une sorte de tube spectral, beaucoup mieux adapté à la granularité de ce type de spectre, en proposant une contrainte simple et efficace pour chaque longueur d'onde.

Finalement, nous avons écrit un prototype 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. Enfin, il faut noter l'apport de ce programme en informatique et en technologie de traitement d'images au travers des manipulations réalisables (découpage spatial et spectral, *browser* de bandes, ...) et de l'analyse spectrale (spectrogramme, pipette instantanée, ...) pour le traitement de grosses quantités d'information.

Ce prototype a été présenté avec succès au laboratoire Image et Ville de Strasbourg, avec lequel nous avons tissé des liens depuis longtemps, et a été retenu comme un outil de classification intéressant dans le cadre d'un projet plus vaste, le projet européen TIDE ([NET 10]).

Perspectives

Nous sommes 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 faut étendre notre formalisme de représentation pour y inclure des concepts sémantiquement plus riches comme la forme des objets, leur texture, ... Nous pourrions alors proposer à l'utilisateur un moyen d'introduire sa propre représentation des règles, en fonction des connaissances préalables qu'il a du domaine d'étude.

Ce formalisme très ouvert exigera des opérateurs génétiques appropriés, capables de regrouper des régions de pixels ou de découvrir des relations entre les séquences de valeurs. L'inclusion au préalable de toutes ces méta-connaissances améliorera les performances actuelles de notre algorithme et renforcera notre savoir et notre expérience en télédétection.

D'autres points seraient à perfectionner comme l'optimisation des temps de réponse, qui restent essentiels pour un outil destiné à un usage intensif, face à la complexité croissante de la structure des règles et des images analysées. Pour le moment, nous n'utilisons qu'une source de données homogène. L'intégration d'une technologie permettant le traitement de plusieurs stratégies de règles différentes et de plusieurs représentations d'images différentes (images multi-sources ou multi-échelles) serait alors la bienvenue.

Parallèlement, une autre piste de recherche, tout à fait accessible à partir de la base génétique actuelle, conduirait à produire des règles qui ne seraient non plus capables d'abstraire les *valeurs* des pixels sous forme d'intervalles ou autre, comme ici, mais les *variables* d'un calcul permettant de décrire les classes. A l'image de l'indice de végétation (IVN), équation généralisable permettant de connaître la proportion de flore à l'aide d'une simple division, nous nous proposons d'étudier une méthode originale capable de générer de telles formules automatiquement, sous réserve d'un ensemble d'apprentissage suffisamment vaste.

Ces formules, nouvelles jusqu'alors, pourraient être utilisées dans divers domaines nécessitant la classification rapide d'images et ne seraient pas limitées à la télédétection. Après l'étude d'images vidéos représentant l'évolution saisonnière de la végétation ou de la pollution, les images médicales pourrait être aussi concernées : en effet, elles sont légèrement plus simples, car la résolution n'est souvent pas un problème majeur et il n'y a qu'un nombre restreint de classes à reconnaître. Par contre l'erreur acceptée est beaucoup plus faible (tumeur, ...) ce qui nécessite des outils décrivant ces classes au médecin sous une forme compréhensible dont il saisit le processus de production, et non une image classifiée, ce que nous serions capable de faire. L'étude qui en découlerait devrait engendrer des méthodes suffisamment génériques pour être appliquées à un domaine quelconque, tant que la connaissance factuelle s'exprime sous forme de matrices.

ANNEXE A

Les méthodes non supervisées

Le but de cette annexe est double. Premièrement, elle permet d'introduire le lecteur à un ensemble de méthodes très importantes dans le cadre de la télédétection et sur lesquelles notre équipe a à la fois publié et réalisé de nombreux travaux. Deuxièmement, les méthodes non supervisées étant l'exacte symétrie des méthodes supervisées, elles les complètent souvent, notamment dans notre cas puisque certaines images expertes présentées dans l'étude de cas du chapitre 5 ont été produites par les algorithmes de [Wemmert, 1999].

Ce type de méthodes ne fait pratiquement aucune hypothèse sur les résultats à obtenir. Dans certains cas il faudra malgré tout définir le nombre maximal de classes à obtenir (c'est par exemple le cas des nuées dynamiques, [Demiriz, 1999] ou des cartes de [Kohonen, 1988]).

Au contraire de l'apprentissage supervisé, il s'agit ici de trouver un ensemble de regroupements (aussi appelées *clusters*) dans lequel une distribution de probabilité peut être déterminée, décrivant la place de certains objets, et permettant de prédire la localisation de ces objets dans une autre partie de l'environnement. Les tâches typiques dans ce domaine sont le regroupement de documents graphiques ou textuels. Le but de ces méthodes est d'extraire une *structure* (dans le sens d'un agencement particulier et significatif de pixels) parmi un assortiment de pixels, par exemple dans le cas de la télédétection de détecter les structures caractéristiques d'une ville (à savoir des bâtiments, ...), de faire de la détection de contours ou des mesures de surface de cette ville, etc. Il s'agit aussi de trouver une représentation concise des données, en tirant parti de la redondance des objets présents dans l'images (formes, couleurs, taille des surfaces, ...), pour arriver à décrire ces ensembles de données de manière plus succincte et plus précise. Cette description rejoint la notion abstraite de concept vue plus haut, inhérent au domaine d'application.

Les méthodes adaptées aux réseaux de neurones

Les algorithmes d'apprentissage utilisant des réseaux de neurones modifient les poids du réseau pour obtenir des sorties homogènes à partir d'exemples sources (images, données brutes) présentant des similitudes. On n'impose au réseau ni son architecture, ni de connaissances a priori sur les associations désirées entre un exemple et la sortie attendue du réseau. Cependant, on lui présente suffisamment d'exemples pour qu'il puisse découvrir lui-même des relations qui ont du sens. Les travaux effectués au sein du LSIIT par [Hammadi, 1995] et [Novak, 2000] sur des méthodes de classification connexionnistes non supervisées tendent à créer un système coopératif qui sélectionne les attributs pertinents des objets recherchés, classe ces objets et évalue les résultats obtenus.

Les **cartes auto-organisatrices** de [Kohonen, 1988] constitue un exemple connu d'apprentissage non supervisé. Bien entendu d'autres méthodes existent, comme les **réseaux élastiques** (voir la résolution du problème du voyageur de commerce sur [NET1]) et les **gaz neuronaux** ([Martinetz, 1991], [NET 2]), notamment les grilles extensibles de Fritzke, l'apprentissage compétitif dur, l'apprentissage Hebbien, ... Ces réseaux ont été étudiés en détails pour

l'application à la télédétection et constituent l'une des méthodes de résolution de ces problèmes proposées par l'équipe AFD du LSIIT ([Novak, 2000]).

Résumons le principe général de l'algorithme de Kohonen pour des cartes 2D :

Une carte de Kohonen 2D est un réseau de neurones comprenant deux couches entièrement connectées entre elles (réseau connexe), dont celle de sortie représente une matrice de neurones à deux dimensions. La particularité de ces réseaux est de pouvoir traiter les problèmes intégrant une dimension spatiale comme dans le cas de la télédétection. Pour garantir une forte dépendance entre deux neurones voisins, on définit le *voisinage* d'un neurone i appelé *centroïde* par un ensemble de neurones se trouvant à une distance $V_i(t)$ qui diminue au cours du temps. Grâce à cette dépendance, deux entrées analogues affecteront des valeurs similaires aux poids des connexions de deux neurones voisins. Les cartes de Kohonen permettent de surcroît une *réduction de la dimensionalité*, c'est-à-dire qu'un espace à 3 dimensions ou plus (l'espace spectrale des données de l'image) est projeté sur un espace à 2 dimensions constitué par le réseau, d'où une simplification notoire de la représentation de ces données. D'autre part, le fait qu'un même neurone traduit l'information d'un ensemble de points qui sont proches dans l'image, et donc de points semblables, produit ce que l'on appelle un *effet de clusterisation*. Voici l'algorithme présenté de manière condensée :

ALGORITHME A.1
L'apprentissage des cartes de Kohonen (source [NET 6])

_____

Etape 1 : On initialise les poids du réseau de manière aléatoire. On initialise un paramètre d'apprentissage α.

Etape 2 : Tant que les poids des neurones n'ont pas convergé, répéter les étapes 3 à 7 :

Etape 3 : Pour chaque vecteur d'entrée x, on calcule :

Etape 4 : Pour chaque centroïde j, on calcule $D(j) = d(x, w_j)$, une distance entre les poids des connexions du centroïde j et le vecteur x.

Etape 5 : On détermine le centroïde j qui minimise $D(j)$.

Etape 6 : On ajuste les poids des neurones se trouvant dans le voisinage $V_j(t)$ du centroïde j de la manière suivante :

$$w_{ij}^{new} = w_{ij}^{old} + \alpha(x_i - w_{ij}^{old})$$

Etape 7 : Modification des paramètres d'apprentissage : on diminue $V_i(t)$ et α.

D'autres méthodes importantes

La plupart de ces travaux, comme nous allons le voir, ont été effectué au sein de notre équipe AFD (Apprentissage et Fouille de Données), dirigé par J.J. Korczak [NET 4].

- **Le regroupement** (**clustering**) utilisé par exemple pour la reconnaissance de motifs (ang. *pattern recognition*). L'algorithme de regroupement partitionne les entrées en un nombre fixe de groupes (*clusters*) de manière à ce que les entrées d'un même groupe sont proches (similaires) les unes des autres, par respect d'une certaine métrique dans l'espace des entrées, et présentent des différences plus grossières si elles sont de deux groupes différents. Une étude de détection de motif dans des images est présentée dans [Campbell, 1997].

- L'apprentissage par **formation de concepts** a été abondamment étudié et décrit par [Ketterlin, 1995], en particulier dans le cadre de la télédétection. Ce sont des méthodes non supervisées dites d'*agrégation de données,* qui permettent de créer une *représentation conceptuelle* de l'espace de données. Ce sont des classifieurs générant une structure hiérarchique qui établissent les similarités entre deux individus, ce qui permet d'intégrer de nouveaux individus appartenant à la même catégorie. Les individus sont regroupés en classes selon leurs similarités donc, mais sont aussi isolés selon leurs dissimilarités pour obtenir des classes plus précises. On peut citer :

    - La **méthode Unimem** [Lebowitz, 1987] permet de créer une classification *hiérarchique*, *incrémentale descendante*. Hiérarchique et descendante parce que chaque objet est ajouté à l'arbre créé en partant de la racine (le concept le plus général), puis en descendant dans les sous-objets, en retirant à chaque fois de sa description les attributs exprimés par chaque objet rencontré). Incrémentale parce qu'elle permet d'intégrer de nouveaux objets, sans avoir à re-parcourir l'ensemble des objets déjà classifiés.

    - La **méthode Cobweb** [Fisher, 1987] ne considère plus la description d'un objet comme étant formée d'un seul attribut (stratégie *monothétique*), mais comme un ensemble d'attributs associés chacun à une distribution de probabilité, indiquant la probabilité qu'un objet a d'avoir cet attribut (stratégie *polythétique*). Le principal avantage de cette méthode est qu'elle autorise la classification d'objets portant des attributs marginaux, qui ne seraient pas traités sinon.

    - L'un des travaux de N. Louis, publiés dans [Korczak, 1999] est de construire une arborescence qui synthétise une hiérarchie d'objets donnée par un algorithme ou un procédé de classification comme K-Means, Unimem ou Cobweb et ses dérivés.

- Une approche originale consiste à mêler toutes ces méthodes non supervisées, de façon à construire **une méthode hybride** d'apprentissage. L'intérêt est de faire partager les connaissances de plusieurs méthodes (nécessitant souvent des images de types différents) et d'échanger leurs résultats, alors que ce sont des algorithmes relativement différents, tant sur le plan des paramètres demandés en entrée que des bilans obtenus en sortie. Nous pouvons citer les travaux de notre laboratoire à ce sujet dans [Wemmert, 2000], qui ont permis d'aboutir à un logiciel (SAMARAH), développé par P. Gançarski et C. Wemmert. Un supplément d'informations se trouve dans la publication de [Wemmert, 1999].

- D'autres projets sont en cours ou terminés, par exemple le projet **CORTEX - Classification d'environnement** (modèles neuronaux basés sur des données géographiques

permettant le positionnement d'antenne émettrices pour la téléphonie mobile) ou les **Modèles hiérarchiques causaux pour l'analyse multi-images** (production de classifications non supervisées pour analyser entre autre des données multirésolution/multispectrale fournies par le satellite Landsat), tous deux projets à l'INRIA.

ANNEXE B

La création d'un spectrogramme

Cette annexe est utile dans la mesure où elle présente une méthode déterministe d'extraction d'informations sur la probabilité de répartition des valeurs de réflectance des pixels. La construction d'un spectrogramme est intéressante comme pré-traitement et comme source d'indications pour l'initialisation de l'algorithme génétique présenté au chapitre 4.

Un spectrogramme indique pour une classe et une bande donnée la répartition des valeurs de réflectance des pixels appartenant à cette classe. Voici tout d'abord les principales étapes de la création de ce spectrogramme :

Supposons que pour chaque classe, on effectue les deux étapes suivantes :

- Déterminer l'ensemble des pixels de l'image brute étiquetés de cette classe par l'expert,

- Pour chaque pixel de cet ensemble et pour chaque bande, assombrir sur une colonne d'un pixel de large un point en fonction de la valeur de ce pixel pour la bande donnée. Un point noir en haut de la colonne correspond à une valeur forte (souvent 255 ou 65535), un point noir en bas de la colonne correspond à une valeur faible (0). Plusieurs points superposés deviennent de plus en plus noirs. La colonne est donc un ensemble de pixels allant du blanc (RGB 255,255,255) au noir (RGB 0,0,0).

Notre prototype propose une visualisation graphique de spectrogrammes permettant d'appréhender pour chaque classe la réponse spectrale la plus fréquente des pixels, en affichant en noir les valeurs les plus fréquentes, et en dégradé de gris les valeurs les moins fréquentes. Voici ce que l'on obtient, de manière agrandie, dans le cas de l'image SPOT de la première étude de cas du chapitre 5 (3 bandes) :

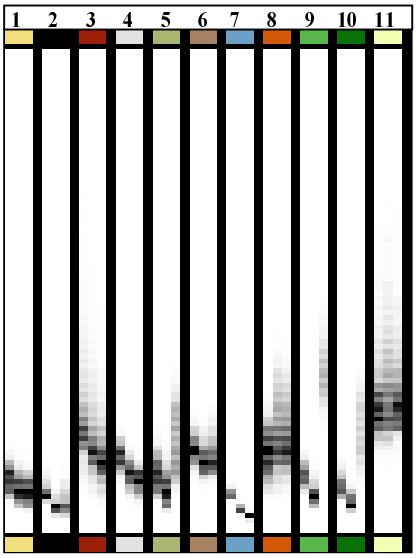


Fig. B.1 : Spectrogramme de l'image SPOT

Chaque colonne représente une classe (11 en tout), dont la couleur représentative a été placée en légende en haut. Pour chaque classe, il y a 3 bandes, donc 3 colonnes de pixels. La hauteur de chaque colonne correspond aux différentes valeurs que peuvent prendre chaque pixel (256 ou 65536 selon le codage des données brutes). Ici cette hauteur est de 100. En effet on "projette" la taille théorique réelle, ceci pour deux raisons :

- L'image serait trop grande et inexploitable si elle avait une hauteur de 65536 plus les quelques pixels de la légende.

- Le cumul d'un certain nombre de pixels pendant la projection procure une information plus consistante (des plages de valeurs permettent de mieux voir la tendance des valeurs de ces pixels, qu'une information sur des valeurs individuelles, à cause de la densité de ces plages).

D'une manière générale, un ensemble de pixels noirs correspond à un critère discriminant pour cette règle, alors que de longues traînées en gris clair nous informent que la bande est inutile.

Le spectrogramme indique clairement les bandes dans lesquelles les règles doivent prévoir des disjonctions d'intervalles, et les bandes dans lesquelles cela est inutile.

Examinons plus précisément les colonnes n° 2 et n° 7. Elles correspondent respectivement à l'ombre et à l'eau. Voici les règles associées, découvertes par notre système de classifieurs :

- Ombre : { [6;64] } { [3;21] } { [11;42] } (note : 95,38%)
- Eau : { [6;38] } { [0;27] } { [1;15] } (note : 95,58%)

L'étude statistique de l'expert avait confondu eau et ombre. Ici, 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 (à cause du spectre plus diffus des régions couvertes par 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. Bien entendu dans le cas de l'eau ces intervalles pourraient être encore plus fins. Mais ceci n'est pas la faute des opérateurs génétiques : que les intervalles restent tels quels ou soient plus affinés, ceci ne change rien à l'affaire car ils suffisent pour discriminer les classes, compte tenu du peu d'influence des autres classes dans cette plage de valeurs. C'est pourquoi la fonction d'évaluation ne les a pas éliminés pour les remplacer par d'autres intervalles.

Le spectrogramme est toujours le même pour une image brute et une image experte données. De plus il se calcule très rapidement (moins d'une seconde), et donne une idée générale des règles que l'on va obtenir, ce qui en fait son intérêt comme pré-traitement, comme validation par l'utilisateur des règles produites ou comme indice dans la création d'un pool initial de règles.

REFERENCES

[Agrawal, 1993] R. Agrawal, T. Imielinski, A. Swami, *Mining Associations between Sets of Items in Massive Databases*, Proc. of Int. Conf. on Management of Data, Washington D.C.

[Banzhaf, 1999] W. Banzhaf, J. Daïda, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, R. E. Smith, *Dimensionally Aware Genetic Programming, Proc. of the Genetic and Evolutionary Computation Conference*, pp. 1069-1076.

[Barto, 1985] A.G. Barto, P. Anandan, *Pattern-recognizing Stochastic Learning Automata*, IEEE Transactions on Systems, Man and Cybernetics, 15, pp. 360-375.

[Barto, 1986] A.G. Barto, P. Anandan, C. Anderson, *Cooperativity in Networks of Pattern Recognizing Stochastic Learning Automata*, Narenda, New York, Plenum Press.

[Blanzieri, 1998] E. Blanzieri, *Learning Algorithms for Radial Basis Function Networks : Synthesis, Experiments and Cognitive Modelling*, Ph. D. Thesis, University and Polytechnic, Turin.

[Blekas, 1997] K. Blekas, A. Likas, A. Stafylopatis, *A Fuzzy Neural Network Approach to Classification Based on Proximity Characteristics of Patterns*, 9th IEEE Int. Conf. on Tools with Artificial Intelligence (ICTAI'97), California.

[Blickle, 1995] T. Blickle, L. Thiele, *Computer Engineering and Communication Networks Lab*, TIK-Report, Second Edition, Swiss Federal Institute of Technology, Zurich.

[Bock, 1999] H. Bock, E. Diday, *Analysis of Symbolic Data*. Data analysis and knowledge organization.

[Boser, 1992] B. E. Boser, I. Guyon, V. N. Vapnik, *A Training Algorithm for Optimal Margin Classifiers, Proc. of the Fifth Annual Workshop on Computational Learning Theory*, 5, pp. 144-152.

[Campbell, 1997] J. G. Campbell, C. Fraley, F. Murtagh, A. E. Raftery, *Linear Flaw Detection in Woven Textiles using Model-Based Clustering*, Pattern Recognition Letters, 18, pp. 1539-1548.

[Cao, 2001] H. Cao, *IMPUTE: A SAS Application System for Missing Value Imputations*, Institute for Social Research, University of Michigan.

[Chavent, 1998] M. Chavent, *A Monothetic Clustering Method*, Pattern Recognition Letters.

[Clark, 1989] P. Clark, T. Niblett, *The CN2 Induction Algorithm*, Machine Learning, 3, pp. 261-283.

[DAIS, 2001] *Proceedings of the Final Results Workshop on DAISEX (Digital AIrborne Spectrometer EXperiment)*, ESTEC, Noordwijk.

[DeJong, 1975] K. A. DeJong, *Analysis of the Behavior of a Class of Genetic Adaptive Systems*, Ph.D. Dissertation (C.C.S.), University of Michigan.

[DeJong, 1988] K. A. DeJong, *Using Genetic Algorithms to Learn Task Programs: the Pitt Approach*, Machine Learning.

[Demiriz, 1999] A. Demiriz, K. P. Bennett, M. J. Embrechts, *Semi-supervised Clustering using Genetic Algorithms*, Rensselaer Polytechnic Institute.

[Fisher, 1987] D. H. Fisher, *Knowledge Acquisition via Incremental Conceptual Clustering*, Machine Learning 2, pp. 139-172.

[Fodor, 1988] J. A. Fodor, Z. W. Pylyshyn, *Connectionism and Cognitive Architecture: A Critical Analysis*, Cognition, 28, pp. 3-71.

[Fong, 2000] C. K. Fong, S. Y. Yuen, *A Genetic Algorithm with Coverage for Object Localization*, Proc. Int. Symposium on Intelligent Multimedia, Video & Speech Processing, University of Hong Kong.

[Fu, 2000] X. Fu, L. Wang, *Linguistic Rule Extraction from a Simplified RBF Neural Network*, Nanyang Technological University, Singapore.

[Furuhashi, 1993] T. Furuhashi, K. Kakaoka, K. Morikawa, Y. Uchikawa, *Controlling Excessive Fuzziness in a Fuzzy Classifier System, Proc. of the Fourth Int. Conf. on Genetic Algorithms*, pp. 635.

[Ganascia, 1987] J. G. Ganascia, *AGAPE et CHARADE : deux techniques d'apprentissage symbolique appliquées à la construction de bases de connaissances*, Thèse soutenue à l'Université de Paris-Sud.

[Gluc, 1987] M. A. Gluck, R. F. Thompson, *Modeling the Neuronal Substrate of Associative Learning and Memory : a Computational Approach*, Psychological Review, vol. 94, pp. 176.

[Goldberg, 1989a] D. E. Goldberg, B. Korb, K. Deb, *Messy Genetic Algorithms: Motivation, Analysis, and First Results*, Complex Systems 3, pp. 493-530.

[Goldberg, 1989b] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Reading, Addison-Wesley.

[Greene, 1994] D. P. Greene, S. F. Smith, *Using Coverage as a Model Building Constraint in Learning Classifier Systems*, Evolutionary Computation, Vol. 2, No. 1, pp. 67-91.

[Hammadi, 1995] F. Hammadi-Mesmoudi, *Classifieur neuronal d'images de télédétection*, Thèse soutenue à l'Université Louis Pasteur de Strasbourg.

[Hebb, 1949] D. O. Hebb, *The Organization of Behaviour*, Wiley.

[Herbrich, 2002] R. Herbrich, *Learning Kernel Classifiers*, The MIT Press.

[Holland, 1962] J. H. Holland, *Outline for a Logical Theory of Adaptive Systems*, J ACM, 9(3), pp. 297-314.

[Holland, 1975] J. H. Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor, University of Michigan Press.

[Holland, 1985] J. H. Holland, *Properties of the Bucket Brigade, Proc. of an Int. Conf. on Genetic Algorithms*, Hillsdale.

[Horn, 1994] J. Horn, D. E. Goldberg, K. Deb, *Implicit Niching in a Learning Classifier System : Nature's Way*, University of Illinois at Urbana-Champaign.

[Kaelbling, 1996] L. P. Kaelbling, L. M. Littman, A. W. Moore, *Reinforcement Learning: A Survey*, Journal of Artificial Intelligence Research, vol. 4, pp. 237-285.

[Ketterlin, 1995] A. Ketterlin, *Découverte de concepts structurés dans les bases de données*, Thèse soutenue à l'Université Louis Pasteur de Strasbourg.

[Kohonen, 1988] T. Kohonen, *Self-Organization and Associative Memory*, Springer-Verlag, Berlin, 2nd Edition.

[Korczak , 1999] J. Korczak, N. Louis, *Synthesis of Conceptual Hierarchies Applied to Remote Sensing, Proc. of SPIE, image and signal processing for remote sensing IV*, Barcelona.

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

[Lebowitz, 1987] M. Lebowitz, *Experiments with Incremental Concept Formation : UNIMEM*, Machine Learning 2, pp. 103-138.

[Martinetz, 1991] T. Martinetz, K. Schulten, *A Neural-gas Network Learns Topologies*, North-Holland, Amsterdam.

[Mascarilla, 1995] L. Mascarilla, *Apprentissage de connaissances pour l'interprétation des images satellite*, Thèse soutenue à l'Université Paul Sabatier de Toulouse, 1995, pp. 27-28.

[Michalewicz, 1996] Z. Michalewicz, M. Schoenauer, *Evolutionary Computation, Control and Cybernetics*, pp. 307-338.

[Michalski, 1983] R. S. Michalski, *A Theory and Methodology of Inductive Learning,* [in] *Machine Learning : An Artificial Intelligence Approach*, Volume I, pp. 83-134.

[Minsky, 1969] M. Minsky, S. Papert, *Perceptrons : An Introduction to Computational Geometry*, The MIT Press.

[Mitchell , 1982] T. M. Mitchell, *Generalization as Search,* Artificial Intelligence, Vol. 18, pp. 203-226.

[Novak, 2000] J. P. Novak, *Méthodes neuronales pour la segmentation d'images de télédétection, et l'apprentissage de concepts*, Thèse soutenue à l'Université Louis Pasteur de Strasbourg.

[Palma, 1997] S. Di Palma, J. Da Rugna, D.A. Zighed, *Apprentissage supervisé polythétique : une évaluation*. Cinquièmes Rencontres de la Société Francophone de Classification, SFC'97, Lyon.

[Powell, 1985] M. Powell, *Radial Basis Functions for Multivariable Interpolation,* IMA Conf. on Algorithms for the Approximation of Functions and Data, RCMS Shrivenham, UK, pp. 143-167.

[Quinlan, 1986] J. R. Quinlan, *Induction of Decision Trees*, Machine Learning, 1: pp. 81-106.

[Quinlan, 1993] J. R. Quinlan, *C4.5 : Programs for Machine Learning*, Morgan Kaufmann, San Mateo.

[Rendon, 1997] M. V. Rendon, *Reinforcement Learning in the Fuzzy Classifier System*, Institut Technologique des Etudes Supérieures de Monterrey.

[Richards, 1995] R. A. Richards, *Zeroth-Order Shape Optimization utilizing a Learning Classifier System*, version en ligne sur [NET 14].

[Richards, 2001] R. A. Richards, *Classifier Systems & Genetic Algorithms*, p. 4.

[Riolo, 1988] R. L. Riolo, *Empirical Studies of Default Hierarchies and Sequences of Rules in Learning Classifier Systems*, Ph.D. Dissertation, Computer Science and Engineering Department, University of Michigan.

[Robertson, 1988] G. G. Robertson, R. L. Riolo, *A Tale of Two Classifier Systems*, Machine Learning, 3, 139-159.

[Rosenblatt, 1962] F. Rosenblatt, *Principles of Neurodynamics: Perceptron and the Theory of Brain Mechanisms*, Washington, DC: Spartan Books.

[Rumelhart, 1986] D. E. Rumelhart, G. E. Hinton, R. J. Williams. *Learning Internal Representations by Error Propagation*. in D. E. Rumelhart, J. L. McClelland (eds.), *Parallel Distributed Processing : Explorations in the Microstructure of Cognition - Volume 1 : Foundations*, Bradford Book, The MIT Press.

[Srikant, 1995] R. Srikant, R. Agrawal, *Mining Generalized Association Rules, Proc. of 21st Int. Conf. on very Large Databases*, Zurich.

[Srikant, 1996] R. Srikant, R. Agrawal, *Mining Sequential Patterns : Generalizations and Performance Improvements, Proc. of Extended Data Base Technology*, Avignon.

[Sutton, 1998] R. S. Sutton, A. G. Barto, *Reinforcement Learning: An Introduction*, The MIT Press.

[Sutton, 2001] A. Sutton, *Négociation entre agents dans un système de classification hybride non supervisée (cas de la télédétection)*, Mémoire de DEA, Université Louis Pasteur de Strasbourg.

[Thrun, 1993] S. Thrun, *Extracting Provably Correct Rules from Artificial Neural Networks*, University of Bonn, Dept. of Computer Sciences.

[Watkins, 1989] C. J. C. H. Watkins, *Learning from Delayed Rewards*, Ph.D. Dissertation, Cambridge University.

[Wemmert, 1999] C. Wemmert, P. Gançarski, J. Korczak, *Un système de classification hybride et son application à la télédétection*, [in] Actes des septièmes rencontres de la Société Francophone de Classification, pp. 273-280, Nancy.

[Wemmert, 2000] C. Wemmert, *Classification hybride distribuée par collaboration de méthodes non supervisées*, Thèse soutenue à l'Université Louis Pasteur de Strasbourg.

[Whitley, 1993] D. Whitley, *An Executable Model of a Simple Genetic Algorithm*, D. Whitley (ed.), Foundations of Genetic Algorithms II, Morgan Kaufmann, San Mateo, pp. 45-62.

[Williams, 1989] R. J. Williams, D. Zipser, *A Learning Algorithm for Continually Running Fully Recurrent Neural Networks*, Neural Computation, 1, pp. 270-280.

[Wilson, 1999] S. W. Wilson, *State of XCS Classifier System Research*, Second International Workshop on Learning Classifier Systems (IWLCS-99), Orlando.

Documents électroniques

Ces pages ont été consultées et consultables durant tout le premier semestre 2002.

[NET 1] Une applet pour résoudre le problème du voyageur de commerce par les réseaux élastiques :
http://nuweb.jinr.dubna.su/~filipova/tsp.html

[NET 2] Une applet pour visualiser graphiquement les méthodes d'apprentissage compétitif (gaz neuronaux, apprentissage Hebbien, ...) :
http://diwww.epfl.ch/mantra/tutorial/french/competitive/html/

[NET 3] Learning Kernel Classifiers, un livre en ligne sur la théorie des classifieurs :
http://www.learning-kernel-classifiers.org/table_of_contents.htm

[NET 4] La liste des projets de notre équipe :
http://hydria.u-strasbg.fr/projets.php3?language=en

[NET 5] Une description des systèmes de classifieurs. Contient aussi un développement sur la théorie évolutionniste (histoire de la vie artificielle, darwinisme, ...) :
http://www.math-info.univ-paris5.fr/~latc/va/va35.html

[NET 6] Une description simplifiée de l'algorithme de Kohonen :
http://www.dice.ucl.ac.be/~verleyse/elec3190/exercices/seance5.pdf

[NET 7] Stratégies de remplacement des classifieurs :
http://www.stanford.edu/~buc/SPHINcsX/bkhm063.htm

[NET 8] Informations sur le programme SPOT :
[NET 8a] http://spot4.cnes.fr/spot4_fr/index.htm
[NET 8b] http://www.spotimage.fr/accueil/system/introsat/payload/welcome.htm
[NET 8c] http://www.ac-creteil.fr/svt/Teledec/Ima_spot/ima_spot.htm

[NET 9] Laboratoire Image et Ville Strasbourg :
http://imaville.u-strasbg.fr/

[NET 10] Projet TIDE :
http://www.istitutoveneto.it/tide/
http://tide.u-strasbg.fr/

[NET 11] Méthode et logiciel SIPINA :
http://www.math-appli-uco.fr/etudiants/projets/sipina/

[NET 12] F. Heitz, responsable du projet "Pôle Image" de Strasbourg :
http://picabia.u-strasbg.fr/lsiit/perso/heitz.htm

[NET 13] Descriptions de certains indices (IVN, ...) :
http://www.cnerta.educagri.fr/infogeo/teldetec/leonardo/leoprot/courfr5.htm

[NET 14] Document en ligne de Robert A. Richards (1995) sur les systèmes de classifieurs :
http://www.stanford.edu/~buc/SPHINcsX/book.html

[NET 15] Description des algorithmes AQR et CN2 :
http://polya.u-strasbg.fr/~alain/IA/App/node3.html

[NET 16] Satellite d'observation de la Terre (SPOT, CASI) :
http://www.ccrs.nrcan.gc.ca/ccrs/eduref/tutorial/chap2/c2p12f.html

[AUTRES] (sites non cités dans le rapport, mais m'ayant aidés)
http://www.iro.umontreal.ca/labs/lbit/HTML/IFT3330/Induction-HTML/node5.html
http://www.inria.fr/rapportsactivite/RA96/temis/node5.html
http://www.inria.fr/rapportsactivite/RA2000/cortex/contr_cnet.html
http://www.unige.ch/ses/geo/cours/td/seminaire/chap6/lec6.html
http://www.sigmaplus.fr/DataEngine/Htm/de_page.htm
http://www.neurosciences.com/Technologies/french_comment_les_r%E9seaux_de_neurones_.htm