MÉLIDIS : Reconnaissance de formes par modélisation mixte intrinsèque/discriminante à base de systèmes d'inférence floue hiérarchisés

Nicolas Ragot

N° d'ordre: 2889

THÈSE

Présentée devant

devant l'Université de Rennes 1

pour obtenir

le grade de : Docteur de l'Université de Rennes 1
Mention Informatique

par

Nicolas Ragot

Équipe d'accueil : IMADOC/IRISA
École doctorale : MATISSE
Composante universitaire : IFSIC/IRISA

Titre de la thèse :

MÉLIDIS :
Reconnaissance de formes par modélisation mixte intrinsèque/discriminante à base de systèmes d'inférence floue hiérarchisés

soutenue le 28 octobre 2003 devant la commission d'examen

M. :	Georges	Stamon	Président, Rapporteur
Mme. :	Bernadette	Bouchon-Meunier	Rapporteur
MM. :	Maurice	Milgram	Examinateurs
	Marc	Parizeau	
	Éric	Anquetil	
	Guy	Lorette	

Remerciements

J'adresse toute ma reconnaissance à Georges Stamon, Professeur à l'Université René Descartes (Paris 5), qui m'a fait l'honneur de présider ce jury, d'avoir accepté la charge de rapporteur, mais aussi pour le soutien et les conseils qu'il a su m'apporter.

Je remercie Bernadette Bouchon-Meunier, Directeur de Recherche du CNRS à l'Université Pierre et Marie Curie (Paris 6), d'avoir accepté la charge de rapporteur et pour l'intérêt qu'elle a porté à ces travaux.

Je remercie Maurice Milgram, Professeur à l'Université Pierre et Marie Curie (Paris 6), ainsi que Marc Parizeau, Professeur à l'Université de Laval (Canada), d'avoir accepté d'examiner ce travail. Merci notamment à Marc Parizeau pour son regard critique et pertinent.

Je souhaite également remercier vivement Guy Lorette, Professeur à l'Université de Rennes 1 et Éric Anquetil, Maître de conférence à l'INSA de Rennes, d'avoir encadré ces travaux, pour leurs conseils précieux et enrichissants ainsi que pour le recul qu'ils ont su m'apporter.

Je tiens plus particulièrement à exprimer toute ma reconnaissance et ma profonde sympathie à Éric Anquetil, sans qui ce travail ne serait pas ce qu'il est. Je tiens à le remercier pour son soutien, sa rigueur et la motivation qu'il a su m'insuffler, mais aussi pour la sincérité et le dévouement avec lesquels il a accompli cette tâche d'encadrement, sans relâche pendant plus de trois ans.

Un grand merci à Claude Labit, Directeur de l'IRISA, qui m'a donné la chance de pouvoir faire ces recherches. Merci à tous le personnel et aux membres de l'IRISA et en particulier à l'équipe IMADOC pour le cadre de travail idéal qui m'a été offert, mêlant dynamisme et bonne humeur.

Enfin, je souhaite exprimer toute ma gratitude à l'ensemble des personnes qui, bien qu'en marge de ce travail, ont contribué largement à son aboutissement. Que ce soit depuis toujours, depuis plusieurs années ou quelques mois seulement, nombreux sont celles et ceux qui ont participé à mon épanouissement et je les en remercie sincèrement.

À mon père,
pour son dévouement

En hommage à ma mère

Table des matières

Introduction **13**

1 Principes généraux de la reconnaissance de formes **17**
1.1 L'homme, l'écrit et la machine 19
1.1.1 La lecture automatique de documents papiers 19
1.1.2 L'écriture électronique orientée stylet 20
1.1.3 Problématique de la reconnaissance de formes manuscrites 22
1.1.4 Discussion 24
1.2 Notions élémentaires sur la reconnaissance de formes 25
1.2.1 Formalisation de la fonction de classification 26
1.2.2 Espace de représentation des entrées 27
1.2.3 Espace de sortie 28
1.2.4 Modélisation et apprentissage automatique à partir de données : une vision centrée sur les connaissances 29
1.2.4.1 Une vision des connaissances dans un système de reconnaissance de formes 30
1.2.4.2 Modélisation implicite des classes par des fonctions discriminantes et modélisation explicite des classes par des prototypes 32
1.2.4.3 Architecture et structuration des connaissances dans un classifieur 33
1.2.4.4 Apprentissage automatique d'un classifieur 34
1.2.4.5 Interprétabilité de la modélisation 35
1.2.5 Fiabilité et notions de rejet 37
1.2.5.1 Le refus de classement 38
1.2.5.2 Rejet de distance 38
1.2.6 Évaluation des performances d'un classifieur 41
1.2.6.1 Taux de reconnaissance et taux d'erreurs 41
1.2.6.2 Rejet et fiabilité 43
1.2.6.3 Autres critères pour caractériser les systèmes de reconnaissance 45
1.3 Différentes approches de reconnaissance 46
1.3.1 Méthodes probabilistes 46

1.3.2 Les k plus proches voisins . . . 47
1.3.3 Les réseaux de neurones de type perceptron . . . 48
1.3.4 Les machines à vecteurs supports . . . 48
1.3.5 Les réseaux de neurones de fonctions à base radiale . . . 50
1.3.6 Les systèmes d'inférences floue . . . 51
1.3.7 Les arbres de décision . . . 52
1.4 Vers une approche générique . . . 53
1.4.1 Le système *ResifCar* . . . 54
1.4.2 Les objectifs de notre approche . . . 56
1.4.2.1 Généricité . . . 56
1.4.2.2 Performances . . . 57
1.4.2.3 Interprétabilité . . . 58
1.4.2.4 Compacité . . . 58
1.4.3 La philosophie de notre méthode . . . 58
1.4.3.1 Architecture générale . . . 59
1.4.3.2 Méthodologie de conception . . . 59
1.4.3.3 Mécanisme de décision . . . 61
1.5 Bilan . . . 62

2 Représentation de connaissances imprécises et architecture structurée par multiples classifieurs 65
2.1 Représentation de connaissances imprécises . . . 67
2.1.1 Théorie des sous-ensembles flous . . . 67
2.1.1.1 Ensembles, sous-ensembles et sous-ensembles flous . . . 67
2.1.1.2 Éléments caractéristiques des sous-ensembles flous . . . 69
2.1.1.3 Opérations sur les sous-ensembles flous . . . 70
2.1.1.4 Normes et conormes triangulaires . . . 73
2.1.2 Raisonnement à partir des sous-ensembles flous . . . 74
2.1.2.1 Variable linguistique . . . 74
2.1.2.2 Propositions floues . . . 75
2.1.3 Règles floues et systèmes d'inférence floue . . . 76
2.1.3.1 Les règles floues . . . 76
2.1.3.2 Raisonnement par déduction . . . 78
2.1.3.3 Systèmes d'inférence floue . . . 78
2.1.4 Cas particuliers en reconnaissance de formes et en soft computing 79
2.2 Vers une approche de combinaison structurée guidée par les données . . . 82
2.2.1 Accroître les performances en combinant plusieurs classifieurs . . . 83
2.2.2 Approches de combinaison fonctionnellement indépendantes . . . 85
2.2.2.1 Indépendance des classifieurs . . . 85
2.2.2.2 Comment obtenir des points de vue différents . . . 86
2.2.2.3 Combinaison de classifieurs avec des sorties hétérogènes 87
2.2.2.4 Différentes méthodes de combinaison . . . 89
2.2.3 Approches de combinaison fonctionnellement dépendantes . . . 91
2.2.3.1 Approches de combinaison correctives . . . 92

2.2.3.2 Approches de combinaison par experts 94
2.2.4 Une nouvelle approche : la combinaison structurée guidée par les données 95
2.2.4.1 Problèmes de conception avec les approches de combinaison usuelles 95
2.2.4.2 Le mode de combinaison proposé 96
2.3 Bilan 99

3 Extraction et structuration des connaissances dans notre modélisation : méthodologie et apprentissage 101
3.1 Les connaissances utilisées pour la modélisation du système *Mélidis* . . . 103
3.1.1 Les connaissances de nature intrinsèque aux classes 103
3.1.1.1 Leurs rôles dans notre approche 103
3.1.1.2 Leurs propriétés 104
3.1.2 Les connaissances de nature discriminante 106
3.1.2.1 Leur rôle dans notre approche 106
3.1.2.2 Leurs propriétés 107
3.1.3 Structuration des connaissances 108
3.1.4 Synthèse 108
3.2 Extraction automatique de sous-ensembles flous 111
3.2.1 Intérêts de la classification non supervisée 111
3.2.2 Les C-moyennes floues 112
3.2.2.1 Principes 112
3.2.2.2 Avantages et inconvénients de l'algorithme 113
3.2.3 Les algorithmes dérivés des C-moyennes floues 114
3.3 Niveau intrinsèque : modélisation des connaissances et décomposition . . 115
3.3.1 Inconvénients des C-moyennes floues pour l'extraction de connaissances intrinsèques 116
3.3.2 Les C-moyennes possibilistes 118
3.3.3 Propriétés de l'algorithme pour une modélisation intrinsèque . . 120
3.3.4 Extraction des modèles intrinsèques aux classes 121
3.3.4.1 Configuration des C-moyennes possibilistes pour la caractérisation des classes 122
3.3.4.2 Définition des modèles intrinsèques aux classes 124
3.3.5 Principes de la décomposition en sous-problèmes 125
3.4 Niveau discriminant : modélisation et structuration des connaissances . . 127
3.4.1 La structure d'arbre de décision pour organiser les connaissances discriminantes 128
3.4.2 Mécanisme d'apprentissage général des arbres de décision 129
3.4.2.1 Le choix d'un attribut pour partitionner 131
3.4.2.2 La stratégie de partitionnement 133
3.4.2.3 Critère d'arrêt et élagage d'un arbre de décision 134
3.4.2.4 Création d'une feuille 135

3.4.2.5 Inconvénients des arbres de décision classiques dans le cadre de notre approche . . . 136
3.4.3 Les arbres de décision flous . . . 136
3.4.4 Une nouvelle approche de construction d'arbres de décision flous pour l'extraction de connaissances discriminantes . . . 137
3.4.4.1 L'algorithme d'apprentissage . . . 137
3.4.4.2 Obtention des modalités floues . . . 138
3.4.4.3 Les C-moyennes floues pour représenter des connaissances discriminantes . . . 140
3.4.4.4 Partitionner par les C-moyennes floues . . . 141
3.4.4.5 Choix d'un sous-espace pour discriminer les classes . . . 145
3.4.4.6 Création des feuilles . . . 147
3.5 Bilan . . . 147

4 Exploitation du modèle pour la décision : combinaison de systèmes d'inférence floue 149
4.1 Mécanisme de décision et complémentarité des connaissances . . . 151
4.2 Intégration des connaissances dans des SIF . . . 151
4.2.1 Module de pré-classification . . . 152
4.2.1.1 Règles à une conclusion . . . 152
4.2.1.2 Règles à conclusions multiples . . . 153
4.2.1.3 Règles optimisées pour la classification . . . 153
4.2.2 Module de classification principale . . . 154
4.3 Processus de décision pour la classification . . . 156
4.3.1 Sélection des connaissances discriminantes et rejet de distance . . . 156
4.3.2 Pré-classification . . . 157
4.3.3 Classification principale . . . 158
4.3.4 Classification finale . . . 159
4.4 Bilan . . . 160

5 Expérimentations 163
5.1 Benchmarks et bases utilisées pour les expérimentations . . . 165
5.1.1 Les benchmarks classiques . . . 165
5.1.2 Problèmes associés à la reconnaissance en-ligne de caractères manuscrits . . . 165
5.1.2.1 La base IRONOFF . . . 166
5.1.2.2 La base UNIPEN . . . 166
5.1.2.3 Type de caractéristiques utilisées pour décrire les tracés 168
5.2 Expérimentations préliminaires sur notre approche des arbres de décision flous . . . 168
5.3 Évaluation des performances générales du système *Mélidis* . . . 170
5.3.1 Benchmarks classiques . . . 171
5.3.2 Reconnaissance en-ligne de chiffres manuscrits . . . 172
5.3.2.1 Reconnaissance omni-scripteurs . . . 172

5.3.2.2 Reconnaissance multi-scripteurs 174
5.3.3 Reconnaissance en-ligne de lettres manuscrites 174
5.4 Étude des différents modules du système 175
5.4.1 Le système à la loupe 175
5.4.2 Apports de la décomposition guidée par les données 177
5.5 Implémentation 178
5.6 Bilan 178

Conclusion et perspectives **181**

Annexes **187**

A Principales distances **189**

B Classification non supervisée **191**
B.1 Les algorithmes de classification non supervisée hiérarchiques 192
B.2 Les algorithmes basés sur une fonction objectif 193

C Algorithmes génétiques **197**
C.1 Principes généraux des algorithmes génétiques 197
C.2 Exemple pour la sélection de caractéristiques 198

Publications personnelles **201**

Bibliographie **203**

Table des figures

1.1 Traitement d'un document « papier » par numérisation et reconnaissance. 20
1.2 Acquisition d'un signal au moyen d'un stylet et d'une tablette tactile. . . 22
1.3 Exemple de variabilité intra-scripteur pour la lettre 'a'. 23
1.4 Exemple de variabilité inter-scripteur pour la lettre 'a'. 23
1.5 Schéma fonctionnel général d'un classifieur f_M. 26
1.6 De la forme à l'entrée du classifieur : le choix d'un espace de représentation. 27
1.7 Deux types de modélisation : description implicite des classes par fonctions discriminantes (a) et description explicite des classes par prototypes (b). 33
1.8 Schéma fonctionnel d'un système de classification gérant le refus de classement. 38
1.9 Représentation de deux modes de gestion du rejet de distance : approche par modélisation discriminante (a) et approche par modélisation explicite par prototypes (b). 39
1.10 Schéma fonctionnel d'un système de classification gérant le rejet de distance par un classifieur spécialisé. 39
1.11 Schéma fonctionnel d'un système de classification gérant le rejet de distance de façon intégrée. 40
1.12 Forme e devant être rejetée et provoquant une erreur si le rejet de distance est modélisé de façon discriminante (a), mais bien gérée par le rejet reposant sur une description des classes par prototypes (b). 40
1.13 Schéma fonctionnel d'un système de classification gérant le rejet de distance en s'appuyant sur la modélisation du classifieur principal. 41
1.14 Répartition des exemples dans la base de test : cas général. 42
1.15 Découpage d'une base d'exemples en fonction du protocole de test utilisé : (a) découpage classique : (b) validation croisée en k-parties. 43
1.16 Répartition des exemples dans la base de test : avec refus de classement. 44
1.17 Répartition des exemples dans la base de test en fonction des décisions du classifieur : avec refus de classement et rejet de distance. 45
1.18 Marge et hyperplan séparateur obtenus par une SVM pour un problème à deux classes ω_1 et ω_2. 49
1.19 Niveaux de descriptions associés au modèle d'un allographe du caractère 'a' dans le système *ResifCar*. 55
1.20 Schéma-bloc du système *Mélidis*. 59

1.21 Structuration des connaissances dans le système *Mélidis*. 61
1.22 Mécanisme de décision du système *Mélidis*. 62

2.1 Fonction caractéristique de E_{grand} 68
2.2 Exemples de fonctions d'appartenance. 68
2.3 Fonction d'appartenance à E^f_{grand} 69
2.4 Principaux éléments caractéristiques d'un sous-ensemble flou A. 71
2.5 Union de deux sous-ensemble flou A et B. 71
2.6 Intersection de deux sous-ensemble flou A et B. 72
2.7 Variable linguistique associée à la taille d'une personne 75
2.8 Schéma de combinaison parallèle 89
2.9 Schéma de combinaison série. 92
2.10 Schéma de combinaison série-parallèle (cascade). 93
2.11 Schéma-bloc du système *Mélidis*, s'appuyant sur une combinaison structurée guidée par les données. 97
2.12 Exemples de problèmes relatif à l'utilisation directe des sorties d'un classifieur pour une combinaison à deux étages. Si le deuxième étage utilise les sorties du premier, les formes noires, qui sont semblables, seront traitées différemment (a). Ce problème est absorbé dans le cas de la combinaison structurée guidée par les données (b). 99

3.1 Illustration de la description de trois classes par des prototypes intrinsèques dans un espace à deux dimensions. Ces prototypes ne tiennent pas compte des valeurs non typiques. 104
3.2 Illustration de la décomposition du problème à partir de la description intrinsèque des classes par des prototypes : sur la base des classes (a) ; sur la base des sous-classes (b). 105
3.3 Classes à discriminer (ω_1 Vs. ($\omega_2 + \omega_3$)) dans un sous-problème issu de la description intrinsèque de la classe ω_1. 106
3.4 Fermeture du 'a' comme critère de discrimination par rapport au 'u'. . . 107
3.5 Structuration des connaissances dans la modélisation du système *Mélidis*. 109
3.6 Résumé du processus d'apprentissage de la modélisation du système Mélidis. 110
3.7 Forme et comportement des fonctions d'appartenance issues des *C-moyennes floues* lorsque deux regroupements sont recherchés : en une dimension (a) ; en deux dimensions après projection des lignes de niveaux ($\mu = 0,9$ à $\mu = 0,5$). Les degrés d'appartenance tendent vers $0,5$ lorsqu'on est à égale distance des centroïdes, que ce soit près de ceux-ci (individu 'x') ou loin (individu 'y') (b). 116
3.8 Problèmes liés à la définition relative des fonction d'appartenance dans les *C-moyennes floues* : la donnée 'x' est du bruit mais elle déforme et déplace les sous-ensembles flous. 117
3.9 Forme d'une fonction d'appartenance issue des *C-moyennes possibilistes* en une dimension. 120

3.10 Caractérisation intrinsèque par les *C-moyennes possibilistes*. 121
3.11 Immunité au bruit des prototypes issus des *C-moyennes possibilistes*. . . 121
3.12 Différence entre les fonctions d'appartenance exponentielles et les fonctions de type « cloche » utilisées. 123
3.13 Exemple de modèle intrinsèque pour une classe constituée de trois sous-classes. 125
3.14 Exemple du mécanisme de décomposition à partir des modèles intrinsèques de trois classes. 126
3.15 Exemple d'arbre de décision (partiel) pour la discrimination de caractères latins. 129
3.16 Structure des arbres de décisions flous utilisés. 139
3.17 Frontière discriminante résultant d'une modélisation par les *C-moyennes floues*. 140
3.18 Exemple de visualisation d'un partitionnement en deux dimensions par les C-moyennes floues. Le contexte de discrimination est représenté à la fois par les données locales au nœud (base B constituée ici d'exemples de 2 classes : '$\times$' et '$\diamond$') et l'espace $\{F_{k1} \times F_{k2}\}$ permettant la discrimination. Les deux modalités floues sont représentées par les lignes de niveaux associées aux fonctions d'appartenance. 144

4.1 Principe du mécanisme de décision du système *Mélidis*. 152
4.2 Chaque chemin de l'arbre qui va de la racine à une feuille est traduit par une règle. 155

5.1 Exemples de chiffres manuscrits de la base IRONOFF. 166
5.2 Exemples de chiffres manuscrits de la base UNIPEN. 167
5.3 Exemples de lettres manuscrites de la base UNIPEN. 167
5.4 Réduction relative des erreurs par rapport à la pré-classification (%). . . 177

A.1 Illustration des principales distances de Minkowski. 190

B.1 Exemple de dendogramme résultant d'une classification non supervisée hiérarchique. 193

C.1 Exemple de codage des individus dans un algorithme génétique utilisé pour rechercher un sous-espace d'attributs : un individu est représenté par 6 gènes à valeurs booléennes correspondant aux 6 attributs de l'espace complet. L'individu (a) représente l'espace complet, et l'individu (b) représente un sous-espace de 3 attributs. 199
C.2 Exemple de la création de deux fils à partir de deux individus (a) et (b) en utilisant un croisement à un point. 199
C.3 Exemple de mutation d'un individu sur le gène correspondant à l'attribut Att4. 199

Introduction

La diffusion de la technologie à l'ensemble de la population est une tendance particulièrement notable de nos jours. Elle se fait largement au travers des moyens informatiques qui se propagent jour après jour dans les différentes couches de la société. Pour que cette diffusion soit la plus efficace possible, l'ensemble des sciences [1] concernées cherchent à simplifier les interactions entre l'homme et les machines pour les rendre accessibles à tout type d'utilisateur et en toutes circonstances.

En effet, la prise en main d'un système ou d'une application requiert bien souvent un certain temps d'apprentissage. De plus, même lorsque celle-ci est assurée, il subsiste généralement des difficultés importantes pour piloter certaines applications de façon intuitive et efficace. On peut par exemple citer les problèmes d'ergonomie liés à la saisie de formules mathématiques ou encore à l'envoie de messages sur des téléphones portables. Cette difficulté du dialogue entre l'homme et la machine provient d'une part de la souplesse et de la variété des modes d'interactions que nous sommes capable d'utiliser (geste, écriture, parole, etc.) et d'autre part de la rigidité de ceux classiquement offerts par les systèmes informatiques. Une partie de la recherche actuelle vise donc à concevoir des applications mieux adaptées aux différentes modalités de communication couramment employées par l'homme. Il s'agit de doter les systèmes informatisés de fonctionnalités permettant d'appréhender les informations que l'homme manipule lui-même tous les jours.

D'une façon générale, la nature des informations à traiter est très riche. Il peut s'agir de textes, tableaux, images, paroles, sons, écriture, gestes, etc. De par les contextes géographiques, sociaux, professionnels, applicatifs et personnels, la façon de représenter ces informations et de les transmettre est très variable. Il suffit de considérer par exemple la variété des styles d'écriture, que ce soit entre les différentes langues et même pour une même langue. De plus, à cause de la sensibilité des capteurs et des médias utilisés pour acquérir et transmettre les informations, celles qui sont finalement traitées diffèrent très souvent des originales. Elles sont donc caractérisées par des imprécisions, soit intrinsèques aux phénomènes dont elles sont issues, soit liées à leurs modes de transmission. Leur traitement [47] nécessite donc la mise en œuvre de systèmes complexes d'analyse et de décision. Cette complexité est un facteur limitant important dans le contexte de la diffusion des moyens informatiques. Cela reste vrai malgré la croissance des puissances de calcul et l'amélioration des systèmes de traitement, puisque parallèlement, les

1. informatiques, physiques, cognitives, etc.

recherches s'orientent vers la résolution de tâches de plus en plus difficiles et vers l'intégration de ces applications dans des systèmes nomades peu chers et ne possédant donc que de faibles ressources informatiques. Aussi, l'élaboration de ces systèmes nécessite presque toujours un compromis entre les fonctionnalités souhaitées et les capacités que la machine « réceptacle » possède. Le plus bel exemple à l'heure actuelle est sans doute l'intégration d'outils d'assistance variés et la prise en compte de modalités complexes comme l'écriture manuscrite sur des biens de consommation courants comme les téléphones portables ou les assistants personnels.

Dans cette thèse, nous nous intéressons plus particulièrement aux systèmes de reconnaissance de formes qui constituent bien souvent un maillon essentiel des systèmes de traitement de l'information. Leur rôle consiste à analyser une forme présentée en entrée afin de l'identifier.

Dans ce domaine en général, mais aussi dans des sous-domaines plus précis comme celui de la reconnaissance de formes manuscrites qui est à l'origine de notre démarche, la « Quête du Graal » est sans aucun doute d'arriver à concevoir un mécanisme qui permette de résoudre automatiquement la plupart des problèmes que l'on est amené à rencontrer. De nombreuses approches « génériques » ont donc été élaborées, en s'appuyant sur différents cadre d'études tels que les statistiques, la théorie des langages, la biologie. Mais aucune d'elles n'est réellement « la solution » pour tous les problèmes. Elles possèdent chacune leurs avantages et leurs inconvénients, qui les rendent plus ou moins efficaces dans un contexte donné. Il semble donc particulièrement important de pouvoir bénéficier d'une méthodologie permettant de choisir ou de concevoir un système de reconnaissance en fonction du problème à traiter et de son contexte d'utilisation [76]. Cependant, une telle « grille de correspondance » n'existe pas encore à l'heure actuelle. Les concepteurs doivent donc souvent adapter les systèmes existants ou en créer de nouveaux en fonction de leurs besoins et des contraintes applicatives. Cette tâche s'avère très souvent fastidieuse parce que les classifieurs ont longtemps été considérés comme des « boîtes noires », des composants logiciels, qui à une entrée devait faire correspondre une sortie. En conséquence de quoi il est souvent difficile de les paramétrer correctement et de comprendre pourquoi ils fonctionnent mal dans certains cas.

Nous pensons donc qu'il est nécessaire, pour un classifieur donné, de pouvoir identifier quels sont les éléments qui sont à l'origine de défaillances afin d'essayer de les corriger. Pour cela, le système ne doit plus être considéré comme une « boîte noire » mais plutôt comme un ensemble de modules et de connaissances que le concepteur peut manipuler et comprendre. Ce point de vue est assez proche de celui qui est adopté en extraction de connaissances à partir de données (ECD) [58, 56]. Cependant, dans ce dernier cas, l'ensemble de la chaîne de traitement est orientée vers l'utilisateur, afin de produire des informations et des connaissances sur le phénomène (représenté par les données) pour qu'il puisse les exploiter. Ici les connaissances qui nous intéressent sont avant tout destinées au concepteur, pour la compréhension du système en lui-même.

L'approche que nous présentons tente de remplir un certain nombre d'objectifs qui sont rarement satisfaits dans un même système de reconnaissance de formes. Ainsi, le

classifieur doit être suffisamment *générique* pour pouvoir s'adapter automatiquement ou être adaptable à différents types de problèmes de complexités variées. Pour obtenir de bonnes *performances*, il doit pouvoir être *fiable* et posséder de bonnes propriétés de *robustesse* face à l'imprécision et à la variabilité des formes. En outre, en vue de son usage dans différents contextes applicatifs, on souhaite qu'il soit *compact* afin de faciliter son intégration dans des systèmes embarqués. Enfin, pour faciliter son optimisation et son adaptation aux différents contextes d'utilisation, on souhaite que l'architecture soit modulaire et compréhensible pour le concepteur, c'est-à-dire « *transparente* » en quelque sorte.

Notre méthode repose sur la structuration d'un ensemble de connaissances représentant les propriétés des formes dans leur espace de représentation. L'architecture du système correspond plus précisément à une organisation originale autour de *connaissances intrinsèques aux classes* et de *connaissances discriminantes entre classes*. Ces différentes natures de connaissances ont été caractérisées précisément de façon à identifier leurs propriétés et les fonctions qu'elles pouvaient remplir dans un système de reconnaissance. L'architecture du système a ensuite été conçue autour de ces connaissances, de façon à les extraire, les représenter et les structurer en fonction de leur nature. Ainsi, les connaissances intrinsèques aux classes sont représentées par des prototypes flous qui permettent au système de se focaliser sur les formes des différentes classes ayant des propriétés similaires et donc plus difficiles à différencier. Une nouvelle forme d'arbres de décision flous est ensuite utilisée sur le résultat de cette focalisation pour extraire de façon fine et robuste les connaissances discriminantes permettant de distinguer les formes des différentes classes. Le système est donc entièrement guidé par les données et centré sur la complémentarité de ces deux natures de connaissances, à la fois sur le plan fonctionnel, structurel mais pour aussi pour la décision.

Tout au long de ces travaux, nous nous sommes appuyé sur l'expérience acquise dans le cadre de la reconnaissance de formes manuscrites et notamment sur les mécanismes du reconnaisseur dédié *ResifCar* [13, 14] qui présente des propriétés intéressantes pour la résolution de notre problématique. Ce sont eux qui ont permis d'aboutir au système de reconnaissance *Mélidis* (ModÉLIsation Intrinsèque/DIScriminante) que nous présentons ici.

Dans le chapitre 1, nous présentons les concepts élémentaires de la reconnaissance de formes en partant de la problématique particulièrement riche de la reconnaissance de formes manuscrites. L'ensemble de l'exposé s'appuie sur la notion de connaissances dans un classifieur, leurs liens avec les données et leur intérêt pour le concepteur. Nous terminons ce chapitre en donnant nos objectifs ainsi que les grandes lignes de notre approche.

Le chapitre 2 décrit ensuite le cadre général sur lequel se fonde notre méthode : la représentation et le raisonnement à base de connaissances imprécises grâce à la théorie des sous-ensembles flous et l'utilisation de multiples classifieurs pour pouvoir traiter des problèmes complexes et structurer un système de reconnaissance. Nous en profitons pour introduire précisément l'architecture du système de reconnaissance *Mélidis*

et ses différences par rapport aux autres méthodes reposant sur une combinaison de classifieurs.

Le chapitre 3 est dédié à la modélisation du système et à son apprentissage automatique. L'intérêt d'utiliser à la fois des connaissances intrinsèques aux classes et des connaissances discriminantes entre classes y est développé, ainsi que les moyens utilisés pour les extraire, les représenter et les structurer en fonction de leur nature et de nos objectifs. Nous proposons notamment une nouvelle approche par arbres de décision flous particulièrement adaptée pour la représentation des connaissances discriminantes.

Dans le chapitre 4, nous décrivons comment cette modélisation est exploitée pour extraire un ensemble de règles de décision formalisées par des systèmes d'inférence floue (SIF). Nous présentons ensuite l'ensemble du mécanisme de décision et notamment comment ces SIF sont combinés pour profiter des avantages de la structuration et de la complémentarité des deux types de connaissances utilisées.

Le système de reconnaissance *Mélidis* a été validé par un ensemble d'expérimentations permettant de mettre en évidence l'intérêt de son architecture structurée autour des connaissances intrinsèques et discriminantes. L'ensemble des propriétés du système (performances, généricité, compacité, complémentarité des connaissances) ont notamment été validées à partir de benchmarks classiques et de problèmes plus complexes comme ceux liés à la reconnaissance de caractères manuscrits, en particulier au travers de la base UNIPEN. Les résultats de ces tests sont reportés dans le chapitre 5.

Finalement, nous terminons ce mémoire en concluant sur notre travail ainsi qu'en proposant un certain nombre de perspectives qu'il reste à étudier.

Chapitre 1

Principes généraux de la reconnaissance de formes

La reconnaissance de formes est un domaine majeur de l'informatique dans lequel les recherches sont particulièrement actives. Il existe en effet un très grand nombre d'applications qui peuvent nécessiter un module de reconnaissance, notamment dans les systèmes de traitement visant à automatiser certaines tâches de l'homme. Parmi celles-ci, la reconnaissance de l'écriture manuscrite et plus généralement la reconnaissance de formes manuscrites est un problème délicat à traiter car il regroupe à lui seul une bonne partie des difficultés généralement rencontrées en reconnaissance de formes. Il constitue donc un bon point de départ pour l'élaboration d'une architecture générique pour la reconnaissance de formes.

Dans ce chapitre, après avoir introduit le contexte et la problématique de la reconnaissance de formes manuscrites, nous présentons les principes fondamentaux régissant le fonctionnement des systèmes de reconnaissance. Tout au long de la description l'accent est mis sur la nature de la modélisation. C'est en effet elle qui confère aux systèmes leurs propriétés et les rend plus ou moins aptes à traiter certains problèmes. Nous terminons ce chapitre par une présentation de l'approche proposée dans cette thèse dont l'objectif est de fournir une architecture pour la reconnaissance qui soit générique et souple et avec laquelle le concepteur puisse interagir pour optimiser le système dans son contexte d'utilisation.

1.1 L'homme, l'écrit et la machine

L'écriture est une modalité de communication utilisée par l'homme depuis bien longtemps pour transmettre les informations à travers l'espace et le temps. C'est aussi devenu un complément indispensable à la modalité orale, à la fois pour assurer la pérénité des informations (fonction de stockage) mais aussi parce qu'elle offre des possibilités de description mieux adaptées à certaines tâches. Ainsi, l'écrit est devenu une étape incontournable pour la représentation d'agencements spatiaux par le biais de schémas et de diagrammes, pour les raisonnements mathématiques avec symboles et formules, pour la composition musicale avec les partitions, etc.

Aujourd'hui, l'ère numérique offre des possibilités de diffusion, de stockage et d'assistance pour le traitement de l'information bien supérieures à celles classiquement offertes par le support papier. Cependant, l'usage de ce dernier reste très largement répandu, pour des raisons essentiellement pratiques, économiques et culturelles. Afin que l'homme puisse bénéficier à la fois des avantages du numérique et de l'écrit, il est indispensable que les systèmes informatisés puissent absorber cette modalité. Cela passe par deux processus de traitements : la lecture automatique de documents papiers et l'écriture électronique à l'aide d'un stylet. Bien que les mécanismes de lecture/écriture soient parfaitement maîtrisés par l'homme, ils n'en sont pas moins complexes et leur mise en œuvre sur des machines pose un ensemble de problèmes. Parmi ceux-ci, la reconnaissance de formes manuscrites (symboles, lettres, mots, etc.) constitue un maillon central délicat à aborder. Mais la richesse de sa problématique le rend aussi particulièrement intéressant pour la reconnaissance de formes en général.

Dans ce paragraphe, nous présentons dans un premier temps le contexte de la reconnaissance de formes manuscrites dans le cadre du traitement automatique de documents papiers ainsi que dans celui de la saisie avec un stylet. Nous décrivons alors plus précisément l'ensemble des problèmes associés et montrons comment ceux-ci peuvent se retrouver dans les problèmes de reconnaissance de formes en général.

1.1.1 La lecture automatique de documents papiers

Même si actuellement un effort important est fait pour que le numérique soit de plus en plus utilisé comme support de l'information graphique, le papier est loin d'avoir « dit son dernier mot ». Il reste un média très accessible, peu cher et donc très employé. De plus, il existe une quantité très importante d'ouvrages et de documents qui sont conservés dans de nombreux endroits tels que les administrations ou les centres culturels. De par le format même du « papier », le stockage, la conservation et l'accès à ces documents sont généralement coûteux et difficiles. Mettre ceux-ci sous une forme numérique représente alors une solution avantageuse.

La numérisation est une première étape dans ce sens mais elle n'est pas suffisante. Il faut aussi pouvoir indexer les documents afin de faciliter l'accès et la recherche de

ceux-ci [12]. Pour cela, le contenu (et notamment le contenu textuel) doit être au moins partiellement accessible sous un format directement exploitable par le système informatique. La numérisation est alors couplée à une phase de reconnaissance *hors-ligne* des caractères qui travaille directement à partir d'une image du document (cf. figure 1.1). Lorsque le document d'origine est imprimé, le mécanisme de reconnaissance est appelé *OCR* (Optical Character Recognition). S'il est manuscrit, la dénomination est *ICR* (Intelligent Character Recognition).

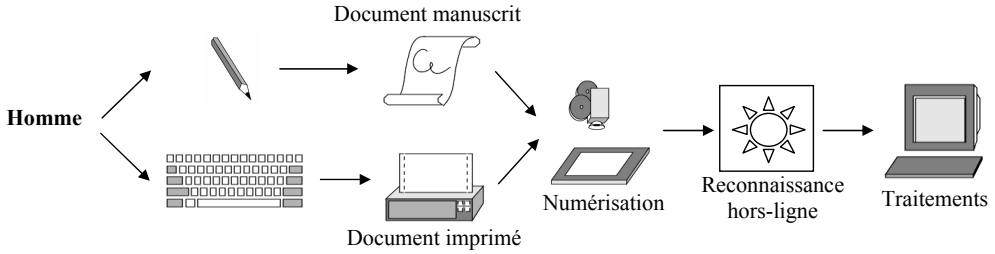

FIG. 1.1 – *Traitement d'un document « papier » par numérisation et reconnaissance.*

Il existe à l'heure actuelle un certain nombre d'OCR qui fonctionnent de façon tout à fait satisfaisante sur les documents imprimés (pour peu que les conditions de numérisation soient correctes). Mais le traitement des documents manuscrits tels que les documents d'archives, les feuilles de soins, les notes manuscrites, pose toujours des problèmes, bien que les ICR soient de plus en plus courantes.[1] Cette difficulté liée à la modalité elle-même est encore accrue par les problèmes résultants de la qualité du support. En effet, lors de la phase de numérisation, le grain du papier, la présence de quadrillage, de taches ou encore la dégradation liée à l'âge, ajoutent à l'image des informations inutiles et souvent gênantes, complexifiant le traitement automatique du document [40, 42].

1.1.2 L'écriture électronique orientée stylet

Parallèlement au traitement automatique de documents papiers, la prise en compte de la modalité écrite par les systèmes informatiques s'est aussi développée au travers de nouveaux modes d'interaction.

Traditionnellement, les ordinateurs ne proposent à l'utilisateur que le clavier et la souris comme seuls outils de communication. Hors ceux-ci ne sont pas toujours faciles à maîtriser et ils sont très mal adaptés pour certaines tâches. D'une manière générale, ils sont difficilement utilisables dans le cadre de la prise de notes, telle que nous la

1. Pour des exemples d'ICR, le lecteur peut se référer à :
http://www.recogniform.com/icr.htm
http://www.readsoft.com/icr.shtml
http://www.haifa.il.ibm.com/projects/image/ocr/icr.html
http://www.recoscript.com/english/src-main.htm
http://www.abbyy.co.uk/products/kofax.htm

pratiquons quotidiennement. Plus particulièrement, la saisie rapide de texte nécessite une bonne maîtrise du clavier et donc un apprentissage qui est en général assez long et fastidieux. De plus, pour la saisie de formes graphiques, la souris est très mal adaptée, introduisant notamment des imprécisions dans les gestes. Les éditeurs proposent alors une assistance par des systèmes de menus permettant de sélectionner les formes à tracer. Mais ce mode de fonctionnement reste peu intuitif et fatiguant à l'usage. Il suffit pour s'en convaincre de considérer les problèmes liés à la saisie de formules mathématiques par exemple. Enfin, le clavier et la souris sont encombrants et donc inadéquats pour un usage sur les systèmes nomades tels que les assistants de poches ou encore les téléphones portables.

Pour faire face à ces difficultés, des interfaces stylet grand public commencent à voir le jour. Celles-ci peuvent se décliner sous différentes formes, mais toutes reposent sur l'utilisation d'un outil semblable au stylo et d'un support assimilable à la traditionnelle feuille de papier. Sans être nécessairement exhaustif, voici les formes d'interaction stylet les plus courantes :

- le stylet est un stylo ordinaire mais muni d'une caméra qui envoie les images acquises au système chargé d'exploiter l'information. Dans ce cas, le support reste la feuille de papier. Cette dernière peut posséder un quadrillage spécial qui permet de repérer la position du stylo de façon absolue [2] ;
- une tablette (voir un écran) tactile enregistre les mouvements de la pointe d'un stylet à sa surface (*Wacom* [3], *Palm Pilot*, *Pocket PC*, *Smart Phone*, etc.) ;
- une tablette (voir un écran) possède un capteur non tactile capable de repérer la position d'un stylet adapté. Le plus souvent, les capteurs sont électromagnétiques (*Tablette PC*).

Dans ces modes d'interaction, l'information initiale correspondant au geste de l'utilisateur est acquise au moyen de capteurs et transformée sous un format numérique que peut utiliser le système informatique. La tendance actuelle consiste à s'orienter vers la génération d'une encre électronique [4]. Cette encre est composée des informations sur le tracé qui a été effectué sur le support (papier ou tablette). Il s'agit au minimum de la séquence des points acquis à intervalles de temps réguliers (fréquence d'échantillonnage) par les capteurs de la tablette ou du stylo. Ces points, représentés par leur position spatiale (coordonnées en x et en y dans le repère associé au support), sont ordonnés dans le temps. Ce signal contient donc la dynamique du tracé. Il est qualifié de *en-ligne*.

En fonction du type d'interface, l'encre électronique peut aussi contenir des informations supplémentaires telles que la pression du stylet sur la tablette, son orientation en trois dimensions (notamment pour voir si le stylet est penché et dans quel sens),

2. Le système *Anoto* (cf. http://www.anoto.com) en est un exemple. Il est intégré dans le pack *Logitech io* (cf. http://www.logitech.com) qui utilise le cahier spécial *Easybook d'Oxford* (cf. http://www.cahiers-oxford.com)

3. http://www.wacom.com

4. Le lecteur peut se référer aux travaux du *W3C* sur ce sujet : http://www.w3.org/TR/inkreqs/

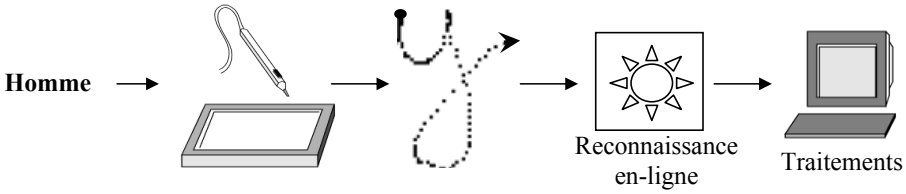

FIG. 1.2 – *Acquisition d'un signal au moyen d'un stylet et d'une tablette tactile.*

la couleur de l'encre courante, etc. Cette encre électronique peut être conservée telle qu'elle. Un certain nombre d'opérations d'édition élémentaires sont alors possible telle que la sélection, le déplacement, la suppression, etc. Cependant, pour que des traitements plus avancés comme l'insertion des données dans un traitement de textes ou une base de données soient possibles, les symboles doivent être identifiés par un système de reconnaissance *en-ligne* (cf. figure 1.2).

1.1.3 Problématique de la reconnaissance de formes manuscrites

Qu'il s'agisse de reconnaissance en-ligne ou hors-ligne de formes manuscrites, la problématique associée est particulièrement riche et complexe.

Tout d'abord, il existe une grande variété de problèmes à traiter en fonction des différents contextes applicatifs. Les besoins vont de la reconnaissance de chiffres à la reconnaissance de mots en passant par la reconnaissance de lettres, symboles, figures géométriques, gestes d'éditions, etc. Même si la complexité de ces différents problèmes est variable, celle-ci reste bien souvent très importante.

Ainsi, le nombre de classes de formes à reconnaître varie selon le problème de une dizaine pour les chiffres à plusieurs dizaines de milliers pour le chinois, rendant la modélisation d'autant plus délicate.

De plus, les formes sont très variables. Cette variabilité est essentiellement de deux ordres : *intra-scripteur* et *inter-scripteur*. La variabilité intra-scripteur résulte de l'imprécision même du geste graphique et de l'incapacité pour une personne de reproduire à l'identique une même forme. Elle peut aussi être une conséquence de l'évolution de l'écriture du scripteur au cours du temps, ou encore du contexte de saisie. Les formes tracées seront en effet différentes selon que le support possède ou non des repères graphiques (grilles, lignes, etc.), que la saisie est faite de façon isolée ou non (lettres seules, en début, milieu ou fin de mot par exemple) ou encore que l'environnement direct du scripteur est plus ou moins propice à l'action même (bureau Vs. métro). La figure 1.3 présente un exemple de variabilité intra-scripteur pour la lettre 'a'.

La variabilité inter-scripteur est quant à elle liée aux différents styles d'écriture que chacun d'entre nous développe de façon personnelle et qui font qu'il existe différents allographes[5] pour un même type de forme (cf. figure 1.4).

5. Un allographe est une représentation concrète de la forme, du caractère.

FIG. 1.3 – *Exemple de variabilité intra-scripteur pour la lettre 'a'.*

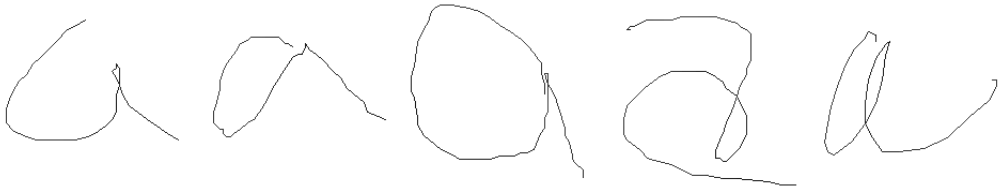

FIG. 1.4 – *Exemple de variabilité inter-scripteur pour la lettre 'a'.*

Un autre facteur de complexité résulte de la présence de bruit dans le signal correspondant aux formes à reconnaître. Ce bruit est un ensemble d'artefacts directement liés à la modalité écrite et aux médias utilisés. Ainsi, pour la reconnaissance hors-ligne, nous avons déjà évoqué le fait que la mauvaise qualité du support était préjudiciable pour la lecture automatique du document. Mais le bruit peut aussi venir du capteur optique, s'il est de médiocre qualité ou partiellement défectueux. Dans le cas de la saisie en-ligne, les capteurs représentent aussi une part importante de la détérioration du signal, que ce soit à cause d'effets de parallaxes liés à l'écran tactile ou à l'inconfort de la surface de contact. Des « maladresses » de l'utilisateur peuvent aussi introduire dans le signal des données supplémentaires qui perturbent les systèmes de reconnaissance. Il peut s'agir par exemple de posés et levés de crayons inutiles, de traits doublés, etc. Même si certains prétraitements comme le filtrage permettent de limiter son impact, la présence de bruit est rarement négligeable dans tout signal et c'est donc un point important que les systèmes de reconnaissance doivent prendre en compte.

Outre cette complexité relative au problème de reconnaissance en lui-même, le cadre applicatif impose aussi certaines contraintes qui rendent les traitements encore plus difficiles. La première est certainement la nécessité d'obtenir de bonnes performances. La notion de « bonnes » performances varie évidemment d'un problème à l'autre et en fonction du contexte d'utilisation mais la recherche des meilleurs taux de reconnaissance est systématiquement considérée comme un objectif de première importance. Ce point revêt une importance toute particulière lorsque le système de reconnaissance constitue un maillon fort d'une chaîne de traitement et que le bon fonctionnement de l'ensemble dépend de ses performances. C'est aussi le cas lorsque le confort de l'utilisateur en dépend directement, comme par exemple pour la saisie de texte sur des assistants personnels ou des téléphones portables.

En dehors des performances brutes, il est aussi fréquent de devoir imposer une certaine fiabilité au système de reconnaissance pour qu'il soit exploitable dans des conditions réelles [65]. Dans le cas de traitement automatique de chèques [178] ou d'adresses postales [65] par exemple, la fiabilité sera même au moins aussi importante que les capacités de reconnaissance en elles-mêmes. En effet, il coûtera moins cher de ne pas traiter un document parce que le classifieur « pense » qu'il risque de se tromper, plutôt que de commettre une erreur et de traiter un chèque avec un mauvais montant ou encore d'envoyer une lettre à une mauvaise destination. En dehors de ce « refus de classement » lié à une ambiguïté, la fiabilité d'un système peut aussi être accrue en prenant la décision de rejeter explicitement certaines formes identifiées comme incorrectes parce qu'elles sont trop dégradées ou n'appartiennent « visiblement » pas aux classes de l'espace de sortie. Par exemple, dans une application traitant des formules mathématiques, trois classifieurs peuvent collaborer : un pour reconnaître les chiffres, un autre pour les lettres et un dernier pour les symboles mathématiques. Si tous utilisent la même entrée ou bien s'ils sont organisés en série (du plus simple au plus complexe par exemple), chaque classifieur doit pouvoir rejeter les formes qui lui parviennent et qui ne correspondent pas à celles pour lesquelles il a été conçu.

Une autre contrainte liée au cadre applicatif concerne les possibilités d'embarquement du système. De plus en plus, les applications informatiques tendent à être intégrées sur des machines de poche dont les ressources sont limitées. Les systèmes conçus doivent donc être compacts et peu gourmands en mémoire et processeur. Ces contraintes de coûts restent valables malgré la croissance des puissances de calcul, les progrès de la miniaturisation et l'amélioration des techniques de classification, puisque parallèlement, les recherches s'orientent vers le traitement de problèmes de plus en plus difficiles et vers leur intégration dans des systèmes nomades peu chers et donc avec de faibles ressources.

Enfin, il ne faut pas oublier que dans la plupart des applications, le contexte d'utilisation est amené à évoluer. Ainsi, au fil du temps, les spécificités de l'écriture d'un même scripteur peuvent changer. Une même application peut aussi être utilisée par de nouveaux scripteurs dont les styles n'étaient pas prévus au départ. Une autre forme d'évolution découle de l'ajout ou de la modification de certaines fonctionnalités qui peuvent remettre en cause tout ou partie du système. Tous ces changements imposent de pouvoir maintenir et adapter les modules de reconnaissance.

Ce sont l'ensemble de ces problèmes qui font de la reconnaissance de symboles manuscrits un domaine de recherche complexe encore aujourd'hui.

1.1.4 Discussion

L'ensemble des difficultés présentées ci-dessus dans le cadre de la reconnaissance de formes manuscrites ne sont pas entièrement spécifiques à ce problème. Même si elles ne transparaissent pas toujours de la même manière ni avec la même force, elles sont au contraire assez générales et se retrouvent dans de nombreux autres problèmes de classification et de reconnaissance de formes.

Ainsi, les problèmes de variabilités sont aussi présents de façon presque identique dans la problématique de la reconnaissance de la parole par exemple. D'une façon plus générale, le fait d'avoir plusieurs représentations possible pour une même classe (variabilité inter-scripteur en manuscrit) se retrouve dans de nombreux domaines tels que la botanique où il existe différentes variétés d'une même plante ne différant que par certains critères (couleur, longueur des feuilles, etc.). Il en sera de même en zoologie, mais aussi pour des problèmes de diagnostique et de décision où plusieurs hypothèses peuvent aboutir à la même conclusion, pour l'indexation d'images, etc. En fait, tout dépend du niveau de description auquel on se situe mais dans la plupart des problèmes non triviaux, les classes peuvent être représentées par différentes modalités. On parlera de *classes multimodales*. En dehors de cette première forme de variabilité, la variabilité intra-scripteur peut aussi être généralisée par ce que l'on appellera la *variabilité intra-classe*[6]. Celle-ci est notamment liée à l'évolution des conditions d'acquisition des formes. Il peut s'agir des conditions climatiques dans le cas de capteurs optiques extérieurs, de l'évolution d'une forme au cours du temps, de la précision et du réglage des capteurs.

D'une façon similaire, les problèmes de bruits se retrouveront dans la plupart des applications où la forme présentée au système est issue d'un capteur et plus généralement d'un ensemble de mesures, que celles-ci soient faites automatiquement ou non.

Enfin, l'ensemble des autres difficultés (contraintes de fiabilité, d'embarquement et d'adaptation) sont, comme nous l'avons déjà souligné, essentiellement liées aux cadres applicatifs. Dès lors qu'une exploitation dans des conditions réelles d'utilisation est envisagée, ceux-ci peuvent intervenir à différents degrés. Il nous semble donc important de ne pas les négliger.

À travers ce point de vue, on se rend compte qu'en cherchant à élaborer une approche qui permette de répondre à la problématique générale de la reconnaissance de formes manuscrites, celle-ci ne devient pas nécessairement spécifique à ce problème. Par la mise en œuvre de processus de modélisation et de décision adaptés, elle doit au contraire pouvoir être plus générique et être capable d'appréhender un grand nombre de problèmes de reconnaissance de formes différents et ce dans plusieurs contextes d'utilisation. C'est un des objectifs du travail présenté ici.

1.2 Notions élémentaires sur la reconnaissance de formes

Dans la partie précédente, nous avons vu que pour pouvoir traiter la modalité écrite, les systèmes informatiques avaient besoin d'un module de reconnaissance capable de retrouver l'identité associée à un tracé manuscrit, que celui-ci soit représenté par une image dans le cas de traitements hors-ligne ou bien par un signal issu d'un stylet et d'une tablette pour le traitement en-ligne. Dans cette partie, les principes généraux liés à la reconnaissance de formes sont décrits de façon formelle. Cette présentation

6. La *variabilité inter-classe* dénote quant à elle ce qui permet de distinguer les classes. Plus elle sera grande et plus il est facile (en théorie en tout cas) de discriminer les classes. Lors du choix d'un espace pour représenter les formes, on cherchera à maximiser cette variabilité.

s'appuie de temps à autre sur le cadre applicatif lié au manuscrit à la fois pour illustrer les concepts mais aussi pour mettre en avant un certain nombre d'écueils. Cependant, cela ne constitue pas une restriction à ce cadre d'utilisation.

1.2.1 Formalisation de la fonction de classification

Un classifieur peut être vu comme une *fonction de décision* f qui associe à une entrée e décrite dans un espace E une sortie s d'un espace S :

$$\begin{array}{l} f \ : E \to S \\ \forall e \in E, \exists s \in S \quad f(e) = s \end{array} \tag{1.1}$$

Les entrées du système représentent les formes à reconnaître et les sorties les classes ou catégories auxquelles elles appartiennent. En reconnaissance de formes manuscrites par exemple, une entrée sera représentée par un tracé effectué par l'homme. Le rôle du classifieur consiste à identifier celle-ci afin de retrouver la classe à laquelle elle appartient parmi celles de l'espace de sortie. Cela revient à déterminer dans l'espace des formes les régions associées à chaque catégorie de S. Ces régions sont appelées *régions de décision* et la frontière entre deux catégories, est appelée *frontière de décision*.

Pour faire cette identification, le classifieur utilise la *fonction de décision* f pour confronter l'entrée à une *modélisation* M du problème. Pour marquer le lien entre la modélisation et la fonction de décision nous noterons cette dernière f_M par la suite. Cette notation souligne le fait que pour nous il existe bien deux entités distinctes (fonction de décision et modélisation), même si en pratique elles ne sont pas toujours séparables. Cette distinction facilitera la suite des explications. La figure 1.5 illustre le schéma fonctionnel d'un tel classifieur.

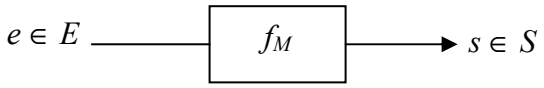

FIG. 1.5 – *Schéma fonctionnel général d'un classifieur* f_M.

On distingue trois étapes dans la mise en œuvre d'un système de reconnaissance. La première étape correspond à la conception du classifieur, c'est-à-dire à l'élaboration de la modélisation M et à son association avec la fonction de décision f. Cette étape peut être faite par un expert ou par un apprentissage automatique à partir d'un ensemble de données du problème appelé *ensemble* ou *base d'apprentissage*. C'est la phase d'*apprentissage*.

La deuxième étape consiste à évaluer les performances du système à partir d'un ensemble de données, la *base de test*, pour déterminer ses capacités de généralisation. Elle est couramment appelée phase de *généralisation*.

Finalement, si le système est intégré dans un cadre applicatif réel, il rentre dans sa phase d'*exploitation*.

Dans les paragraphes suivants, nous décrivons plus précisément chacune des trois entités caractérisant le système de classification : les entrées, les sorties et la modélisation.

1.2.2 Espace de représentation des entrées

D'une façon générale, l'entrée d'un classifieur est une forme qui correspond à une observation de l'environnement extérieur au système. Elle est acquise au moyen d'un capteur et présentée au système sous la forme d'un signal (temporel ou statique). Par exemple, en reconnaissance de formes manuscrites, le système informatique dans lequel est inclus le classifieur est en relation avec l'extérieur par le biais de la modalité écrite. Les informations, issues de la main de l'homme, sont donc transmises au système par un capteur, soit tactile (tablette et stylet) soit optique (OCR/ICR). Une fois le signal acquis, celui-ci peut être présenté au classifieur tel quel (signal brut) ou bien subir un ensemble de modifications, allant de simples prétraitements jusqu'à l'extraction de caractéristiques complexes (cf. figure 1.6).

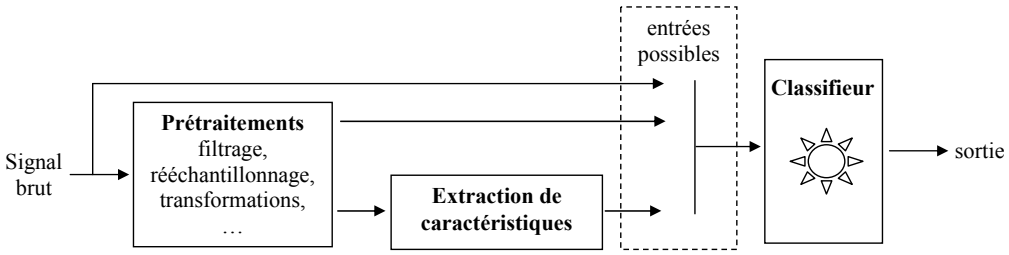

FIG. 1.6 – *De la forme à l'entrée du classifieur : le choix d'un espace de représentation.*

Dans le cas où l'entrée du système est le signal en lui-même, l'espace de représentation E dépend directement du capteur. Il peut s'agir d'une matrice de pixels, d'une séquence de points ordonnée dans le temps, etc. Ce mode de fonctionnement peut poser plusieurs problèmes. Il impose notamment de pouvoir traiter des signaux dans lesquels la quantité d'information utile peut varier (en fonction de la taille de la forme, de la durée de l'acquisition, de la précision du capteur). C'est pourquoi le signal brut subit très souvent une série de prétraitements. L'objectif de ceux-ci est d'obtenir une description de la forme qui soit la plus stable possible, c'est-à-dire qu'entre différentes acquisitions d'un même type de forme, les signaux obtenus doivent être les plus semblables possible. Ces prétraitements sont de natures diverses et variées et dépendent fortement du cadre applicatif. Il peut s'agir de segmentation, de rééchantillonnage, d'élimination d'informations redondantes voir inutiles (bruit), de transformations géométriques (pour

obtenir une invariance par rapport aux rotations, translations) etc. Ce type d'approche a l'avantage de réduire au minimum les prétraitements, ce qui peut être important dans un système où le facteur temps est important. Cependant, la quantité de données à traiter pour une forme peut vite devenir très importante (matrice de pixels par exemple) et dans celle-ci, l'information utile pour l'identification des classes n'est pas toujours directement accessible. C'est donc le classifieur qui doit compenser cette faiblesse. [135] est un exemple de système fonctionnant sur ce mode de représentation.

Une seconde possibilité, très souvent utilisée en pratique bien que nécessitant des traitements supplémentaires, consiste à extraire du signal un certain nombre de caractéristiques, ou attributs, décrivant la forme. Celles-ci sont très souvent numériques (quantité de pixels noirs, centre de gravité, position de points remarquables, etc.) mais peuvent aussi dans certains cas être symboliques. L'objectif est d'obtenir une description à la fois réduite et la plus pertinente possible, c'est-à-dire contenant le maximum d'informations pour pouvoir différencier les classes, et donc faciliter le travail du classifieur. À ce niveau, une sélection et une hiérarchisation des caractéristiques peut être faite, soit automatiquement, soit *a priori* afin de n'employer que les plus pertinentes dans un contexte donné. D'autres connaissances *a priori* peuvent aussi être introduites telles que la manière de mettre en relation différentes parties de la forme pour représenter une information importante (position relative de deux courbes, position d'une lettre par rapport à la ligne de base, etc.). On parlera dans ce dernier cas de caractéristiques de haut niveau.

On notera que, quel que soit l'espace de représentation des formes, il s'agit d'un des points clés de tout système de reconnaissance puisque d'une certaine façon il représente la « matière première ». Si l'information qu'elle contient est insuffisante ou trop variable pour un même type de forme, le classifieur ne pourra en aucun cas fonctionner correctement.

Dans la suite, nous ne considérerons que le cas le plus général sur le mode de représentation d'une entrée e, à savoir qu'elle est décrite par un vecteur $[e^1, \ldots, e^k, \ldots, e^n]^T$ dans un espace de représentation E à n dimensions (n étant fixe). Pour des raisons de simplicité, nous noterons ce vecteur de la même façon que la forme à laquelle il correspond (e). Chaque dimension k représente une caractéristique particulière, prenant elle même ses valeurs dans son propre espace de définition. Dans notre cas, ces valeurs sont supposées numériques, ce qui est le le plus fréquent dès lors que le système est en relation avec l'environnement extérieur par le biais de capteurs.

De plus, nous ne nous intéressons pas directement au problème de l'extraction de ces caractéristiques. Il s'agit en effet, d'un domaine de recherche à part entière [83, 133].

1.2.3 Espace de sortie

Les classes de S correspondent à l'identité des formes par rapport au problème de classification considéré. Elles sont représentées par leurs étiquettes (ou labels) $\{\omega_1, \ldots, \omega_S\}$. Les classes sont généralement connues *a priori* par expertise du problème. Par exemple, pour la reconnaissance de chiffres manuscrits, les 10 classes sont représentées

par les étiquettes suivantes : $\{'0','1','2','3','4','5','6','7','8','9'\}$. Cependant, il arrive que cet espace de sortie ne soit pas initialement connu. Il devra alors être déterminé automatiquement par apprentissage (cf. paragraphe 1.2.4.4).

Lors de la formalisation du classifieur (cf. équation (1.1)), nous avons dit que le résultat de la fonction de décision était une classe $s \in S$. En pratique, la sortie peut prendre différentes formes, en fonction de l'information supplémentaire qu'elle apporte en plus de la classification. Quatre types de sorties sont généralement distingués :

- le type classe : les classifieurs fournissent en sortie une seule classe s : celle qu'ils jugent la plus adaptée à l'entrée présentée (c'est le cas précédemment utilisé) ;
- le type ensemble : ce type de sortie est sans doute moins fréquent car plus difficilement exploitable. Il fournit comme résultat, un ensemble de classes, sans aucune indication sur leur priorité par rapport à la fonction de classification ;
- le type rang : il s'agit d'une extension du type ensemble où, cette fois-ci, un ordre de préférence est donné pour toutes les classes ou pour une partie d'entre elles ;
- le type mesure : là encore, il s'agit d'une extension du précédent type puisque les classes ne sont plus simplement ordonnées mais que cet ordre est quantifié par une mesure (mesure de confiance, score, probabilité, etc.).

Dans la suite, nous considérons le cas général où la sortie s d'un classifieur f_M est un vecteur $[s_1, \ldots, s_s, \ldots, s_{\mathcal{S}}]$ où chaque élément s_s est un indicateur sur l'adéquation de la forme présentée en entrée par rapport à la classe $\omega_s \in S$. Selon le cas, ce vecteur peut être booléen (type classe et ensemble) ou numérique (type rang et mesure).

1.2.4 Modélisation et apprentissage automatique à partir de données : une vision centrée sur les connaissances

Une des principales difficultés en reconnaissance de formes est de choisir et de concevoir la fonction de décision f et la modélisation M sur laquelle elle repose.

Parfois, des connaissances *a priori* sur le problème de classification sont disponibles, suite à des expertises. Elles peuvent alors permettre de choisir le type de classifieur (et donc de modélisation) le plus adapté à la situation. Ces connaissances peuvent aussi apporter au concepteur du système des informations précises sur la configuration du reconnaisseur. Il arrive même qu'elles soient suffisantes pour que l'élaboration de la modélisation et de la fonction de décision puisse se faire entièrement de façon « manuelle », aboutissant ainsi à une conception *a priori*, le plus souvent dédiée au problème.

Cependant, la plupart du temps, très peu de connaissances directement exploitables sont disponibles pour l'élaboration du système. Seul un ensemble d'observations a pu être recueilli. Ces données sont souvent difficiles et complexes à analyser pour en extraire les informations nécessaires à l'élaboration du système. Des mécanismes spécifiques doivent alors être mis en œuvre pour en extraire automatiquement les connaissances qui serviront à l'élaboration du modèle.

Dans les paragraphes suivants, nous abordons les principes liés à l'élaboration du classifieur et à la modélisation du problème. Ceux-ci sont présentés en se basant sur une

vision personnelle relative à la notion de *connaissances* dans un système de reconnaissance. Il s'agit en effet pour nous d'un point clé à l'origine de notre démarche.

1.2.4.1 Une vision des connaissances dans un système de reconnaissance de formes

Dans notre étude, la notion de connaissances est abordée d'une manière un peu particulière par rapport à l'usage courant qui en est fait en informatique. Ce terme étant largement utilisé dans la suite de l'exposé, il convient ici d'en donner une définition précise.

Le terme *connaissance* fait partie de notre vocabulaire courant. Sa définition n'en reste pas moins très générale, laissant de nombreuses possibilités d'interprétation. On trouve par exemple dans le dictionnaire [7] la définition suivante relative à la connaissance dans le domaine de l'éducation, c'est-à-dire selon le sens le plus courant :

Définition 1 (Connaissances en éducation) *Ensemble des notions et des principes qu'une personne acquiert par l'étude, l'observation ou l'expérience et qu'elle peut intégrer à des habiletés.* □

On notera que dans cette définition la connaissance s'*acquiert* et qu'elle peut s'*intégrer* à des habiletés, c'est-à-dire être utile pour l'accomplissement de tâches particulières. Ce sont là des notions fondamentales sur l'origine et la fonction des connaissances qui se retrouvent dans l'usage informatique du terme. Ainsi, toujours dans le même dictionnaire, la définition appliquée à l'informatique et plus particulièrement au domaine de l'intelligence artificielle est la suivante :

Définition 2 (Connaissances en intelligence artificielle) *Ensemble de faits, événements, règles d'inférence et heuristiques permettant à un programme de fonctionner intelligemment.*

Note(s) :

On distingue traditionnellement, en intelligence artificielle, la connaissance et le raisonnement. Bien que cette distinction ne soit certainement pas si nette dans le cerveau humain, la séparation a été très fructueuse pour la mise en œuvre de programmes « intelligents ». Les questions liées à la connaissance sont celles de son acquisition (apprentissage), de sa représentation et de son utilisation (raisonnement). □

Cette définition fait écho à la précédente puisque les mêmes notions liées à l'acquisition et à la fonction des connaissances se retrouvent. Un autre point important est aussi soulevé : celui de leur représentation. Celui-ci résulte directement de la volonté de doter le système informatique de capacités permettant de manipuler ces connaissances

7. Définition récupérée à partir du *Grand Dictionnaire Terminologique* : http://www.granddictionnaire.com.

c'est-à-dire de « raisonner ». Il faut donc trouver un formalisme adapté à la machine. Le choix de ce mode de représentation n'est pas anodin. C'est en effet lui qui sert de « passerelle » entre l'environnement extérieur, c'est-à-dire l'utilisateur, et celui du système. Si les connaissances manipulées et extraites par le système informatique doivent représenter des concepts eux-mêmes utilisés par l'utilisateur, alors elles doivent être interprétables et compréhensibles par lui.

C'est par exemple le cas en *extraction de connaissances à partir de données* (ECD) [56, 57, 58] où, à partir d'une masse de données et d'informations à l'origine inexploitables, le système de traitement doit extraire un ensemble de connaissances informantes pour l'utilisateur. Ainsi, dans [56], les auteurs associent les connaissances aux motifs qui sont déduits à partir des données et à leur intérêt :

> « *we can consider a pattern to be knowledge if it exceeds some interestingness threshold, [...] knowledge in this definition is purely user oriented and domain specific and is determined by whatever functions and thresholds the user chooses.* »

Les motifs sont eux-mêmes définis de la manière suivante :

> « *pattern is an expression in some language describing a subset of the data or a model applicable to the subset. Hence in our usage here, extracting a pattern also designates fitting a model to data; finding structure from data; or, in general, making any high-level description of a set of data.* »

Dans cette définition, les notions précédentes sur l'acquisition, la représentation et le rôle des connaissances se retrouvent à nouveau. Cependant, celles-ci sont enrichies de deux manières. L'acquisition est ici connectée à la recherche de motifs qui sont intimement liés à la description des données. De plus, ces motifs, ou connaissances, deviennent dépendants de leur usage, c'est-à-dire de l'intérêt qu'ils représentent par rapport à ce que l'utilisateur souhaite en faire.

À travers toutes ces définitions, il transparaît que la notion de connaissances, et notamment son extraction et sa représentation, sont intimement dépendantes de son usage et du type d'utilisateurs potentiels. Dans le cadre spécifique de la reconnaissance de formes, les connaissances sont avant tout destinées au classifieur en lui-même. C'est lui qui les intègre dans son architecture afin de pouvoir effectuer sa tâche de reconnaissance. Les connaissances ne se situent donc pas au même niveau qu'en ECD. Nous essayons ici d'en donner une définition pour ce contexte particulier. L'objectif de cette définition est avant tout d'éclairer le lecteur sur ce qui est réellement sous entendu dans ce manuscrit lorsque ce terme est utilisé sans autres précisions.

Définition 3 (Connaissances dans un classifieur) *Ensemble des éléments du système de reconnaissance sur lesquels repose la modélisation, qui sont soit issus de connaissances a priori injectées par un expert, soit issus d'une extraction automatique à partir de données et qui permettent à la fonction de décision d'identifier les formes du problème.* □

Dans cette définition, les connaissances peuvent être de simples paramètres, des exemples de formes réelles ou fictives, des éléments structurels des formes, etc. Ainsi, dans un classifieur neuronal de type perceptron par exemple (cf. paragraphe 1.3.3), les connaissances manipulées par le système correspondent essentiellement à l'ensemble des poids reliant les neurones des différentes couches. L'ensemble de ces connaissances, intégrées dans la structure connexionniste, permettent de remplir l'objectif de classification en déterminant des frontières de décision. Dans un classifieur neuronal de fonctions à base radiales (cf. paragraphe 1.3.5), les connaissances correspondront aux zones de l'espace de représentation décrites par ces fonctions (forme et position des fonctions).

Cette définition se limite donc volontairement aux connaissances que l'on pourrait qualifier d'élémentaires car ce sont elles qui vont nous intéresser plus particulièrement. Cependant, il est bien entendu que d'autres types de connaissances peuvent être distinguées, selon le niveau de description auquel on se place. On pourrait par exemple parler de connaissances structurantes permettant d'organiser les connaissances élémentaires (de même nature ou bien de natures différentes). Au plus haut niveau, le classifieur, c'est-à-dire la fonction de décision f_M, pourrait aussi être considéré comme une connaissance mais destinée cette fois à un utilisateur extérieur (humain ou non). Mais dans ce manuscrit, le terme de connaissance dans un classifieur renverra systématiquement à la notion de connaissances élémentaires.

L'intérêt de considérer un système de reconnaissance comme étant un ensemble de connaissances est qu'il devient nécessaire de s'interroger sur la nature, le rôle, la représentation et l'organisation de celles-ci par rapport aux objectifs que le système doit remplir. Ce sont en effet ces éléments qui confèrent au classifieur ses propriétés et qui peuvent le rendre apte ou au contraire inadapté pour la résolution de certains problèmes. Il s'agit donc d'un fil conducteur important pour l'élaboration d'une nouvelle approche de reconnaissance, comme c'est le cas dans cette étude.

Cette terminologie trouve aussi tout son intérêt dès que l'homme est amené à manipuler ces connaissances. Nous avons déjà cité plus haut le cas typique de l'ECD où un utilisateur exploite le système pour améliorer sa compréhension d'un phénomène. Mais il faut aussi considérer le cas du concepteur du classifieur qui peut être amené à le maintenir, l'adapter ou l'optimiser. L'interprétabilité du système devient alors un point important. Nous y reviendrons au paragraphe 1.2.4.4.

1.2.4.2 Modélisation implicite des classes par des fonctions discriminantes et modélisation explicite des classes par des prototypes

En général, deux types de modélisation peuvent être distingués selon leur objectif : la modélisation par description implicite des classes qui opère par des fonctions discriminantes (*modélisation discriminante*) et la modélisation par description explicite des classes au moyen de prototypes (*modélisation explicite*). Dans la première, le but est de partitionner l'espace de représentation des formes en recherchant les frontières discriminantes permettant de séparer les classes (cf. figure 1.7 (a)). Les classes sont donc comparées les unes aux autres afin de rechercher ce qui les différencie et à s'appuyer

dessus pour déterminer des frontières séparatrices définies par des *fonctions discriminantes*. Dans ce cas, la modélisation M correspond à l'ensemble de ces fonctions et leurs paramètres représentent les connaissances élémentaires du classifieur.

Dans la modélisation explicite, l'objectif n'est pas de trouver les frontières séparant les classes mais plutôt de décrire ces dernières en recherchant des prototypes (aussi appelés modèles ou encore patrons) les caractérisant dans l'espace de représentation (cf. figure 1.7 (b)). Ce sont ces prototypes qui constituent les connaissances du système. Cette modélisation est souvent utilisée pour mesurer la similarité entre une forme en entrée et les différentes classes modélisées. Elle est aussi utile pour obtenir une description explicite et compréhensible de la structure des classes. [8] Cependant, elle peut aussi servir comme connaissance élémentaire pour effectuer une modélisation discriminante des classes. Il suffit pour cela de mettre en concurrence les modèles de chaque classe par le biais d'une fonction de décision.

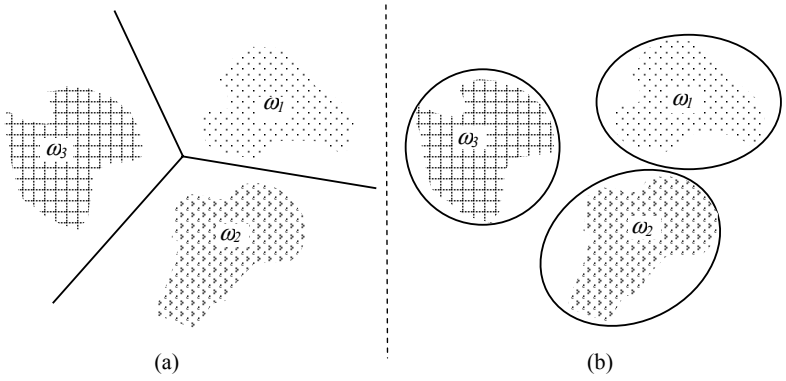

FIG. 1.7 – *Deux types de modélisation : description implicite des classes par fonctions discriminantes (a) et description explicite des classes par prototypes (b).*

Bien que la modélisation discriminante soit la plus souvent utilisée en reconnaissance de formes, ces deux modes de représentation possèdent des propriétés très spécifiques qui s'avèrent fort utiles dans le processus de reconnaissance. Nous verrons quelques exemples un peu plus loin dans ce chapitre (cf. paragraphe 1.2.5.2) et plus particulièrement dans le chapitre 3.

1.2.4.3 Architecture et structuration des connaissances dans un classifieur

Selon les approches, les connaissances élémentaires constituant la modélisation M peuvent être mises en relation de différentes manières, entre elles et par rapport à la fonction de décision. La modélisation est donc plus ou moins structurée.

Lorsqu'elle existe, la structuration se fait notamment selon deux axes. La *structuration horizontale* consiste à utiliser des connaissances de même nature, mais sur des

8. Cela ne signifie pas que toute modélisation explicite est compréhensible.

parties différentes du problème. Ainsi, la description de la structure sous-jacente aux classes en déterminant des sous-classes est un exemple de structuration horizontale. Chaque sous-classe représente un ensemble de propriétés communes à la classe et se situe au même niveau de description que les autres. Les classes décrites de cette façon sont appelées *classes multimodales*.

La *structuration verticale* se traduit quant à elle par une une forme de dépendance entre les connaissances, imposant un ordre sur leur utilisation et donc une forme de hiérarchisation sur différents niveaux. Cette dépendance peut résulter par exemple d'une description sur différents niveaux de détails comme dans une modélisation multi-échelle, ou encore pour une description structurelle hiérarchique de formes (une voiture possède des roues qui possèdent elles-mêmes une jante sur laquelle sont fixés des boulons...). Dans ce schéma, une fonction discriminante représente aussi une structuration verticale des connaissances puisque ces dernières (les paramètres de la fonction) sont dépendantes les unes des autres pour la réalisation de la discrimination.

Ces principes liés à la structuration des connaissances élémentaires entre elles sont aussi valables pour la structuration d'ensemble de connaissances structurées. Ainsi, l'utilisation d'un ensemble de fonctions discriminantes pour modéliser un problème à plusieurs classes est un autre exemple de structuration horizontale. Par extension, les systèmes de reconnaissance reposant sur l'utilisation de multiples classifieurs reposent eux aussi sur ces principes de structuration mais à une plus grande échelle. Nous aborderons ce sujet dans la deuxième partie du chapitre 2 (cf. paragraphe 2.2).

Dans tous les cas, cette structuration, communément appelée *architecture* du classifieur, est un aspect fondamental de toute approche de reconnaissance. En effet, elle permet de conférer au système des propriétés importantes. Ce sont les points sur lesquels nous allons particulièrement insister tout au long de ce document.

1.2.4.4 Apprentissage automatique d'un classifieur

Le mécanisme d'apprentissage a pour objectif d'extraire d'une base d'apprentissage l'ensemble des connaissances nécessaires à la modélisation du problème et à les structurer si besoin est. Les données $\{e_j \ / \ j = 1, \ldots, N\}$ de la base sont elles-mêmes, suivant les auteurs et les contextes, appelées observations, échantillons, individus ou encore exemples. Quand elles ont été expertisées au préalable pour leur attribuer l'étiquette d'une des classes du problème, cette information peut être utilisée lors de l'apprentissage qui est alors qualifié de *supervisé* (ou avec professeur). Chaque individu e_j est décrit dans $E \times S$ par $[e_j^1, \ldots, e_j^k, \ldots, e_j^n, e_j^{(n+1)}]^T$ où $e_j^{(n+1)} \in S$ représente la classe de e_j considérée comme exacte (même s'il peut arriver qu'il y ait des failles dans le processus d'expertise visant à attribuer ces étiquettes). Dans le cas d'un *apprentissage non supervisé* (sans professeur), l'information $e_j^{(n+1)}$ n'est pas disponible *a priori* et l'apprentissage consiste à identifier une structure dans les données en regroupant celles qui possèdent des propriétés similaires. Ce type de modélisation s'effectue par des *algorithmes de classification non supervisée* (cf. annexe B).

L'induction est un processus d'apprentissage dérivé du mode de pensée de l'homme. Il repose sur le principe de la généralisation, et vise à établir une loi générale (connaissance complexe) à partir d'un ensemble d'observations (connaissances simples). Cette démarche est typiquement celle utilisée en sciences physiques où les scientifiques entreprennent une série d'expérimentations pour récupérer des observations (mesures) et pour pouvoir ensuite établir une loi générale ou encore un modèle, plus ou moins complexe. En classification et en reconnaissance de formes, l'induction est donc un mécanisme qui permet d'établir un ensemble de règles de classification générales à partir d'une base d'exemples.

Une des limites de l'induction et des méthodes d'apprentissage à partir de données réside en général dans leur principe même : comme la modélisation est établie à partir d'un ensemble d'observations, celles-ci doivent être suffisamment représentatives pour que les capacités de généralisation du modèle soient efficaces. C'est pourquoi en reconnaissance de formes, la composition de la base d'apprentissage constitue souvent un frein pour l'élaboration des classifieurs, parce que les classes sont représentées de façon hétérogène ou encore parce que des données sont mal étiquetées. Aussi, le mécanisme d'apprentissage doit être capable d'extraire les connaissances les plus pertinentes, tout en éliminant celles qui ne résultent que de la composition particulière de la base et qui ne reflètent pas la réalité. Si la modélisation s'appuie trop sur les « détails » de la base d'apprentissage, le classifieur deviendra trop spécialisé et n'aura pas de bonnes capacités de généralisation. Ce phénomène est appelé *sur-apprentissage*. Il peut être limité dans certaines approches en utilisant pendant l'apprentissage une *base de validation* dédiée à l'évaluation des capacités de généralisation du classifieur.

Un autre inconvénient de l'apprentissage à partir d'exemples résulte de l'acquisition de nouvelles observations qui peuvent remettre en cause la modélisation préalablement établie. Il est alors particulièrement important que la modélisation puisse être adaptée et modifiée pour prendre en compte la variation de son contexte d'utilisation. Sinon, un nouvel apprentissage complet du système doit être effectué, sans nécessairement de garanties sur le résultat.

1.2.4.5 Interprétabilité de la modélisation

Même si le rôle premier d'un classifieur est l'identification de formes (ou d'une façon plus générale d'un objet, d'une situation) la compréhension du modèle et des connaissances qui le constituent se révèle souvent utile, voire nécessaire. Le diagnostique médical en est un exemple puisqu'au-delà de l'identification de la pathologie, le médecin souhaite aussi savoir quels sont les facteurs à l'origine de celle-ci et à quels degrés ils interviennent. Ces éléments sont indispensables d'une part pour vérifier l'hypothèse du système de reconnaissance et d'autre part pour pouvoir soigner plus efficacement la maladie. D'une façon générale, un classifieur est considéré comme « *opaque* » lorsque sa modélisation n'est pas interprétable et que la façon dont il est parvenu à se résultat ne peut être expliquée *a posteriori*. On parle aussi de « boîte noire ». Dans le cas contraire, il est qualifié de « *transparent* ».

D'une façon plus précise, nous relevons au moins deux niveaux d'interprétabilité selon le point de vue adopté. Dans le cas usuel, l'interprétabilité de la modélisation est considérée du *point de vue de l'utilisateur*. Cela correspond à l'exemple précédent et plus généralement au cadre de l'ECD dans lequel on souhaite que le système puisse apporter des informations à l'utilisateur sur le domaine étudié. Les constituants du modèle, leurs relations ainsi que leur rôle doivent alors être clairement identifiables et posséder une signification précise par rapport au problème tel que l'homme le perçoit. Le système manipule donc les mêmes concepts que l'utilisateur dans un formalisme compréhensible par ce dernier.

En plus de cette définition classique, nous introduisons ici un deuxième niveau d'interprétabilité correspondant au *point de vue du concepteur* du système de reconnaissance. Dans ce cas de figure, l'expert du système n'est pas toujours un expert du domaine comme le serait l'utilisateur. De plus, il existe bien souvent une certaine forme d'abstraction du problème initial par la description des données dans l'espace de représentation E. Il est alors difficile de pouvoir aboutir à une modélisation entièrement interprétable au sens où nous l'avons décrite ci-dessus. En revanche, la modélisation du système de reconnaissance peut être suffisamment explicite pour que la nature, la fonction ainsi que le comportement de ses différents constituants puissent être compréhensibles par le concepteur dans le cadre du processus de reconnaissance. L'intérêt est alors évident. Si le concepteur du système est capable d'identifier les différentes briques, la manière dont elles se comportent ainsi que leurs interactions, il lui devient possible d'identifier plus aisément les éventuels problèmes afin d'essayer de les résoudre. De même, si le système nécessite des propriétés précises et un comportement particulier liés à son contexte d'utilisation (ce qui est souvent le cas en pratique), il devient plus facile de les obtenir en modifiant la modélisation et en y injectant de nouvelles connaissances de façon supervisée. C'est donc tout un ensemble d'optimisations guidées par un expert (le concepteur) qui deviennent possibles. On peut illustrer ce principe par l'exemple du mécanicien travaillant sur une voiture de course. En fonction du comportement de la voiture sur un circuit donné, des pièces spécifiques seront changées par d'autres, plus adaptées, et un ensemble de réglages seront effectués de façon à s'adapter au mieux aux conditions de courses. Bien entendu, cela n'est possible que si le mécanicien maîtrise l'ensemble des éléments de la voiture, ce qui passe par une connaissance et une compréhension de ceux-ci.

Pour que ce niveau d'interprétabilité puisse être atteint, il est essentiel que l'architecture repose sur une *modélisation structurée*, en s'appuyant notamment sur l'un et/ou l'autre des axes horizontaux et verticaux. Cette structuration doit notamment permettre d'introduire une certaine modularité dans l'architecture ainsi que différents niveaux de descriptions qui faciliteront la compréhension et le traitement, notamment pour des problèmes complexes. De plus, si les connaissances du modèle sont directement connectées aux propriétés des formes telles qu'elles transparaissent dans l'espace de représentation E, la modélisation gagne encore en interprétabilité car elle conserve un lien explicite avec le référentiel initial (E). Cela rejoint la notion de motif définie dans [56] dans le cadre de l'ECD (cf. paragraphe 1.2.4.1). Par exemple, dans un espace

de représentation numérique, ce type de modélisation s'appuiera sur les agencements « naturels » des formes dans E (forme et position des nuages de points), c'est-à-dire sur des zones de connaissances explicites. Elle est notamment opposée aux modélisations décrivant des régions de E sans rapport direct avec les zones de connaissances explicites, ainsi qu'à celles reposant sur un niveau d'abstraction supplémentaire par rapport à E comme dans les systèmes type « boîtes noires ».

1.2.5 Fiabilité et notions de rejet

Comme nous l'avons déjà mentionné, dans le cadre de la reconnaissance de symboles manuscrits et plus généralement pour la reconnaissance de formes, la fiabilité des réponses du système est souvent un élément primordial dès que ce dernier est intégré dans une chaîne de traitement complète comme par exemple en combinaison de classifieurs [81, 9, 20, 169].

Cette fiabilité du reconnaisseur passe par la mise en œuvre de mécanismes permettant de détecter des formes susceptibles de provoquer une erreur de classification. Ces mécanismes se traduisent par l'ajout de nouveaux types de décisions à l'espace de sortie qui sont employées par le reconnaisseur lorsque celui-ci est incapable d'attribuer à l'entrée une étiquette de S suffisamment pertinente. Deux cas de figure principaux sont à l'origine d'une telle situation :

- la fonction de décision considère que plusieurs classes de l'espace de sortie peuvent être attribuées de manière à peu près identique à une même entrée. Plutôt que de lui assigner une étiquette de façon arbitraire parmi celles qui sont possibles, le reconnaisseur exprime son incertitude et son incapacité à traiter ce cas en prenant la décision de ne pas classer la forme. Celle-ci devra être traitée soit par un autre module dédié à la résolution de tels conflits, soit directement par un acteur humain. Nous appelons ce cas de figure le *refus de classement* et on lui associe la décision ω_{Ramb} pour signifier que c'est une ambiguïté qui est à l'origine de cette décision ;[9]
- aucune classe de S ne correspond à la forme en entrée. Il peut s'agir d'une forme d'un type différent de celui pour lequel le classifieur a été conçu (une lettre pour un reconnaisseur de chiffres par exemple) ou encore d'une forme du bon type mais trop dégradée pour pouvoir être identifiée. Lorsqu'il se trouve dans cette situation, qualifiée de *rejet de forme inconnue* ou encore de *rejet de distance*, le classifieur prend une décision spécifique notée ω_{Rdist}.

Dans beaucoup d'approches de classification ces deux notions sont regroupées sous le terme général de *rejet* [154] qui est alors associé à une seule classe ω_R générale. Cela s'explique soit parce qu'un seul des deux cas est traité, soit parce que d'un point de vue applicatif la distinction entre les deux n'est pas primordiale. Cependant, dans la suite, nous distinguons systématiquement ces deux notions parce qu'elles reposent sur des concepts fondamentalement différents. Le refus de classement traduit une certaine

9. On trouve aussi parfois le terme de *rejet d'ambiguïté* ou encore de *rejet de confusion* pour parler de cette situation. Cependant, le terme de *rejet* est en général associé au deuxième cas de figure, le *rejet de forme inconnue*. C'est pour éviter toute ambiguïté que nous employons le terme de refus de classement.

forme de défaillance du classifieur : il refuse de classer une forme parce qu'il est incapable de l'identifier alors qu'il devrait pouvoir le faire. Au contraire, le rejet de distance correspond à un comportement parfaitement voulu : la capacité du système à identifier les éléments perturbateurs (bruit). La conséquence directe de cette distinction est que leur mise en œuvre nécessite des traitements spécifiques pour chacun d'eux. De plus, si le système est inclus dans une chaîne de traitement, ces deux décisions devront être traitées spécifiquement par des modules différents. Enfin, en les considérant séparément, on récupère des informations différentes sur le fonctionnement du système qui peuvent s'avérer très utiles en vue d'optimisations et d'adaptations futures.

1.2.5.1 Le refus de classement

D'un point de vue fonctionnel, le refus de classement peut être vu comme une fonction de décision f_{Ra}. Celle-ci utilise comme entrée la sortie de f_M et donne une réponse dans $S \cup \{\omega_{Ramb}\}$, où ω_{Ramb} représente la décision associée au refus de classement (cf. figure 1.8).

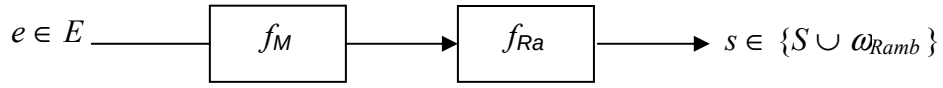

FIG. 1.8 – *Schéma fonctionnel d'un système de classification gérant le refus de classement.*

Pour que f_{Ra} puisse opérer convenablement, il faut que la sortie du classifieur initial soit suffisamment informante. Autrement dit, un système avec une sortie de type classe sera insuffisant ici pour pouvoir gérer le refus de classement. Le plus souvent, le système f_M est défini de façon à produire une sortie de type mesure. Le refus de classement peut alors être implémenté de différentes manières. Généralement, il est réalisé par la détermination d'un ou plusieurs seuils relatifs aux deux sorties (les deux classes) les plus activées par f_M [20, 81, 73].

1.2.5.2 Rejet de distance

Pour qu'un reconnaisseur puisse gérer le rejet de distance, il faut qu'il puisse déterminer si une forme ne correspond à aucune des classes de S. Ces formes inconnues seront considérées comme appartenant à une classe fictive nommée ω_{inc}. Ce problème peut être abordé de deux façons : soit en s'appuyant sur une modélisation discriminante, soit en considérant au contraire une modélisation explicite par prototypes. Un exemple illustratif de ces deux principes est donné par la figure 1.9. Les formes appartenant aux classes ω_1, ω_2 et ω_{inc} sont localisées explicitement dans l'espace de représentation et la zone de rejet de distance déterminée par le système de reconnaissance est hachurée (décision ω_{Rdist}).

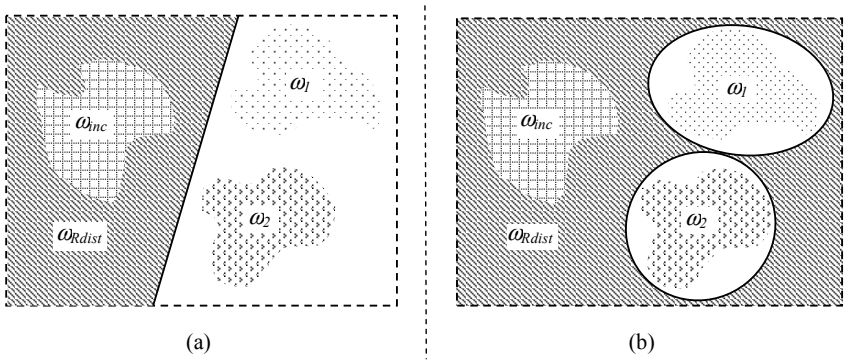

FIG. 1.9 – *Représentation de deux modes de gestion du rejet de distance : approche par modélisation discriminante (a) et approche par modélisation explicite par prototypes (b).*

Modélisation du rejet de distance par une frontière discriminante :
L'approche discriminante consiste à déterminer une fonction de décision f_{Rd} modélisant une frontière discriminante entre les formes de S et les autres formes que le système peut être amené à rencontrer (et qu'il doit donc rejeter). Il peut s'agir comme nous l'avons déjà dit de formes n'appartenant pas à S ou bien de formes trop dégradées, déformées. La modélisation s'effectue généralement à partir d'une base dite de rejet (ou de rebut) dont les échantillons sont étiquetés comme appartenant à ω_{inc}.

Pour élaborer la modélisation Rd, deux possibilités sont envisageables. Soit celle-ci est déterminée complètement indépendamment de M, soit elle y est directement intégrée. Dans le premier cas (envisagé par exemple dans [20]), un classifieur spécialisé effectue une modélisation discriminante f_{Rd} entre la classe ω_{inc} et la classe ω_S qui regroupe l'ensemble des classes de S. Du point de vue fonctionnel, si une entrée e n'est pas rejetée (la décision est ω_S), elle est présentée en entrée du système f_M pour être classée (cf. figure 1.10).

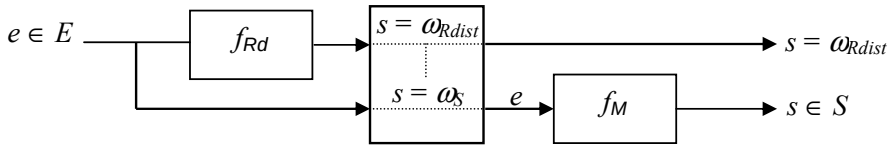

FIG. 1.10 – *Schéma fonctionnel d'un système de classification gérant le rejet de distance par un classifieur spécialisé.*

Dans le deuxième cas, le rejet de distance est directement intégré dans le processus de modélisation de M. Pour bien signifier la différence, celui-ci est noté Md (cf. figure 1.11). Cette dernière solution rend en général le processus d'apprentissage plus complexe et plus difficile à maîtriser, d'autant plus que le type de modélisation utilisé pour construire la modélisation M n'est pas forcément adapté pour la modélisation du rejet de distance.

L'inconvénient de cette vision discriminante du rejet de distance est que sa fiabilité

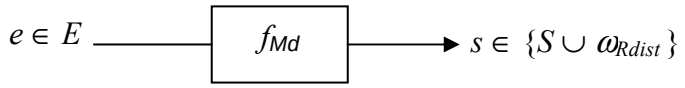

FIG. 1.11 – *Schéma fonctionnel d'un système de classification gérant le rejet de distance de façon intégrée.*

dépend presque entièrement de la pertinence de la base de rebut. Celle-ci doit être suffisamment représentative des formes à rejeter ce qui n'est pas aisé dans le cas de formes dégradées ou mal segmentées. De plus, il est souvent difficile de prévoir à l'avance tous les types de formes à rejeter. Cette méthode peut donc s'avérer gênante dans le cas d'une évolution du système ou de l'environnement d'utilisation. La figure 1.12 illustre un exemple où une forme e, présentée au système, n'est pas rejetée et donc classée, alors qu'elle ne correspond vraisemblablement à aucune classe de S (a). Cette forme est en revanche bien rejetée si le rejet de distance est géré en utilisant une modélisation explicite des classes (b). C'est ce que nous décrivons dans le paragraphe suivant.

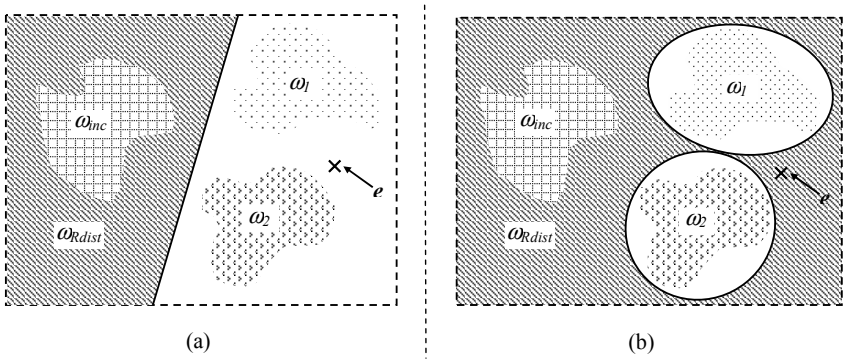

FIG. 1.12 – *Forme e devant être rejetée et provoquant une erreur si le rejet de distance est modélisé de façon discriminante (a), mais bien gérée par le rejet reposant sur une description des classes par prototypes (b).*

Modélisation du rejet de distance en s'appuyant sur une description explicite des classes :
Cette approche utilise une modélisation explicite des classes pour rejeter toute forme ne correspondant pas à cette description. Ce mécanisme peut être rapproché des méthodes plus générales consistant à élaborer des mesures de confiance (ou de pertinence) d'une forme par rapport au modèle M du classifieur [154, 173, 65]. La vision est cependant légèrement différente puisque la mesure de confiance n'est pas déterminée directement par rapport au problème de reconnaissance en lui-même (et notamment par rapport à la description des classes) mais par rapport au modèle du classifieur (quel que soit sont type) et à ses capacités de reconnaissance. Les deux points de vue convergent cependant si l'on considère la finalité de fiabilité du mécanisme de décision.

L'idée ici consiste à localiser et à décrire les zones de connaissances correspondant aux classes de S dans l'espace de représentation. Ces zones représentent d'une certaine façon les « valeurs » typiques que peuvent prendre les échantillons des différentes classes. La description de classes multimodales, telle que nous l'avons évoquée dans le paragraphe 1.2.4.3, est donc un exemple de modélisation possible pour ce problème. Une fois la modélisation Rd établie, une fonction de décision permettant de comparer une forme en entrée à celle-ci doit être choisie. Il peut s'agir d'une distance adéquate, d'une mesure de similarité, de ressemblance, etc. Si une forme ne correspond à aucune des classes, elle est rejetée. Ce type de décision peut être fait en déterminant des seuils qui peuvent éventuellement être appris à partir d'une base de rebut.

L'avantage de cette méthode est qu'elle ne modélise que l'information nécessaire pour le rejet, en s'appuyant sur des connaissances supposées stables. On remarquera en effet qu'une description trop fine des classes peut devenir préjudiciable, entraînant un rejet important. Un autre intérêt de cette approche est que la modélisation M du classifieur peut lui-même tirer avantage de cette modélisation. C'est ce que nous montrerons par la suite dans les chapitre 3 et 4. La conséquence directe de ce dernier point est que le schéma fonctionnel dans cette approche peut être vu de deux façons différentes : soit f_{Rd} effectue sa propre description explicite des classes, indépendamment des connaissances utilisées par M (idem figure 1.10), soit M intègre déjà les connaissances descriptives nécessaires à f_{Rd} qui les utilise directement pour prendre sa décision (cf. figure 1.13).

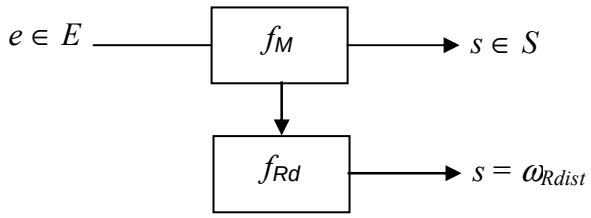

FIG. 1.13 – *Schéma fonctionnel d'un système de classification gérant le rejet de distance en s'appuyant sur la modélisation du classifieur principal.*

1.2.6 Évaluation des performances d'un classifieur

Nous décrivons dans ce paragraphe les critères les plus souvent employés pour évaluer la performance des systèmes de reconnaissance en phase de généralisation. L'objectif est d'obtenir une estimation la plus fidèle possible du comportement du système dans des conditions réelles d'utilisation. Pour cela, des critères classiques comme les taux de reconnaissance et taux d'erreurs sont presque systématiquement utilisés. Mais d'autres critères comme la fiabilité ou la complexité apportent aussi des informations utiles.

1.2.6.1 Taux de reconnaissance et taux d'erreurs

Les taux de reconnaissance et d'erreurs permettent d'évaluer la qualité du classifieur f_M par rapport au problème pour lequel il a été conçu. Ces taux sont évalués grâce à une

base de test qui contient des formes décrites dans le même espace de représentation E que celles utilisées pour l'apprentissage. Elles sont aussi étiquetées par leur classe réelle d'appartenance afin de pouvoir vérifier les réponses du classifieur. Pour que l'estimation du taux de reconnaissance soit la plus fiable possible, il est primordial que le reconnaisseur n'ait jamais utilisé les échantillons de cette base pour faire son apprentissage (la base de test ne doit avoir aucun individu en commun avec la base d'apprentissage et les éventuelles bases de validation). De plus, cette base de test doit être suffisamment représentative du problème de classification.[10]

En général, quand les échantillons étiquetés à disposition sont suffisamment nombreux, ils sont séparés en deux parties disjointes et en respectant les proportions par classes de la base initiale. Une partie sert pour former la base d'apprentissage et l'autre pour former la base de test (cf. figure 1.15 (a)). Le découpage le plus courant est de $2/3$ pour l'apprentissage et le $1/3$ restant pour la base de test. Les performances en terme de taux de reconnaissance sont alors déterminées en présentant au classifieur chacun des exemples e_j de la base de test et en comparant la classe donnée en résultat $f_M(e_j) = s$ à la vraie classe de e_j, c'est-à-dire $e_j^{(n+1)}$. En considérant que la base de test contient N individus et que sur ceux-ci $N_{corrects}$ sont biens classés par le système, le *taux de reconnaissance* est simplement défini par :

$$\tau_{reco} = \frac{N_{corrects} \cdot 100}{N} \tag{1.2}$$

Le *taux d'erreur* τ_{err} est défini à partir du nombre d'individus N_{err} mal classés c'est-à-dire les individus pour lesquels $f_M(e_j) = s$ avec $s \neq e_j^{(n+1)}$:

$$\tau_{err} = \frac{N_{err} \cdot 100}{N} \tag{1.3}$$

La répartition des exemples bien et mal classés est illustré sur la figure 1.14.

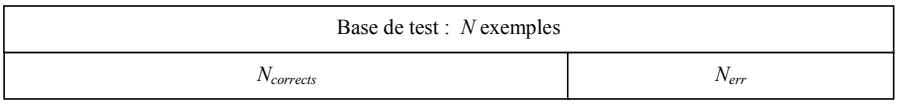

FIG. 1.14 – *Répartition des exemples dans la base de test : cas général.*

Il arrive aussi que le nombre total d'échantillons disponible soit insuffisant pour pouvoir faire à la fois une base d'apprentissage et une base de test. De plus, même si les taux de reconnaissance et d'erreurs peuvent fournir une estimation des capacités de généralisation du classifieur, ils ne permettent pas d'évaluer la stabilité de la méthode d'apprentissage par rapport à des variations sur le contenu de la base d'apprentissage.

10. Dans [72], les auteurs donnent une méthode pour déterminer la taille d'une base de test statistiquement significative dans le contexte de la reconnaissance de caractères et de formes.

Le mécanisme de *validation croisée* [175] permet de répondre aux deux problèmes précédents. Il consiste à diviser la base contenant l'ensemble des échantillons en k sous-bases $b_i,\ i = 1, \ldots, k$ disjointes et de tailles équivalentes. Si chaque sous-base contient la même proportion d'échantillons de chaque classe que la base initiale, la validation croisée est dite *stratifiée* [96]. k étapes apprentissage/validation sont ensuite faites. À l'étape i, l'apprentissage est effectué sur l'union des bases $b_j,\ j \neq i$ et le test sur b_i (cf. figure 1.15 (b)). Le taux de reconnaissance est alors égal à la moyenne des taux obtenus à chaque étape. La stabilité de la méthode d'apprentissage peut être évaluée en calculant l'écart type du taux de reconnaissance sur ces k étapes. Bien entendu, il est aussi possible d'obtenir une meilleure estimation en répétant plusieurs validations croisées et en prenant les résultats moyens.

À noter qu'il existe différentes formes de validation croisée [96]. Citons par exemple le cas où $k = N_{tot}$, avec N_{tot} le nombre total d'échantillons disponibles. Cette méthode de validation est appelée le *leave-one-out*. C'est elle qui donne en général la meilleure approximation du taux de reconnaissance réel du classifieur. Son inconvénient majeur est qu'elle demande beaucoup de temps de calcul.

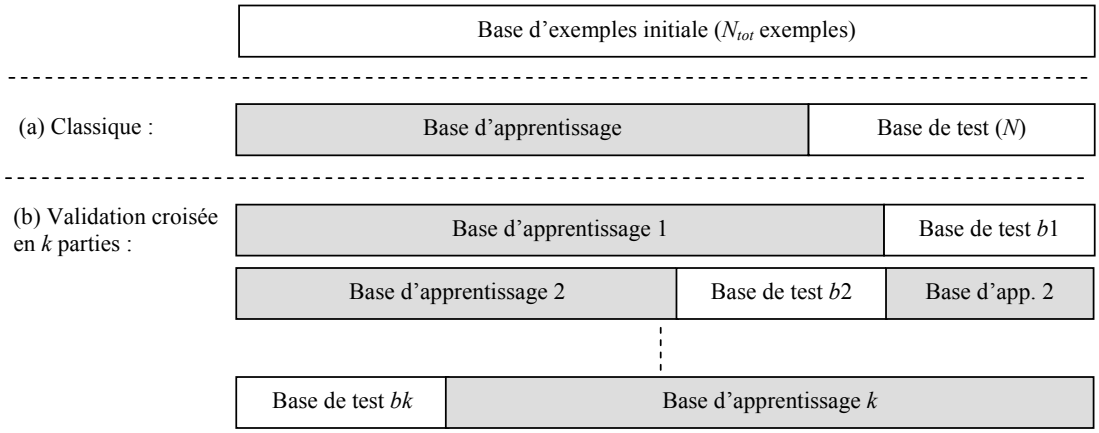

FIG. 1.15 – *Découpage d'une base d'exemples en fonction du protocole de test utilisé : (a) découpage classique : (b) validation croisée en k-parties.*

1.2.6.2 Rejet et fiabilité

Si le classifieur gère le refus de classement, le *taux de refus de classement* lié aux ambiguïtés (τ_{Ramb}) est défini en fonction du nombre d'échantillons N_{Ramb} de la base de test pour lesquels la décision ω_{Ramb} est prise (cf. figure 1.16) :

$$\tau_{Ramb} = \frac{N_{Ramb} \cdot 100}{N} \tag{1.4}$$

Si le classifieur gère en outre le rejet distance, plusieurs critères sont à évaluer afin de déterminer la validité du rejet et son impact sur la classification. Pour cela, la base

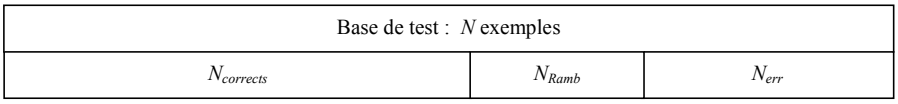

FIG. 1.16 – *Répartition des exemples dans la base de test : avec refus de classement.*

de test doit contenir une base de rebut (différente de celle éventuellement utilisée pour la modélisation de Rd) composée de N_R échantillons d'étiquette ω_{inc} (cf. figure 1.17).

Deux types d'erreurs de rejet sont généralement distinguées. Les erreurs de faux rejet correspondent aux N_{fr} individus de S rejetés (décision ω_{Rdist}) parmi les N_S individus étiquetés dans S ($N_S + N_R = N$). Le *taux de faux rejet* τ_{fr} correspondant est défini par :

$$\tau_{fr} = \frac{N_{fr} \cdot 100}{N_S} \tag{1.5}$$

Les erreurs de fausses acceptations correspondent aux N_{fa} individus de ω_{inc} reconnus comme appartenant à une classe de S alors qu'ils auraient dû être rejetés. Le *taux de fausse acceptation* τ_{fa} correspondant est donné par :

$$\tau_{fa} = \frac{N_{fa} \cdot 100}{N_R} \tag{1.6}$$

Le taux d'erreur total lié au rejet (de distance), τ_{errRej}, est la combinaison de ces deux types d'erreurs :

$$\tau_{errRej} = \frac{(N_{fr} + N_{fa}) \cdot 100}{N} \tag{1.7}$$

Les erreurs restantes sont indépendantes du rejet et liées à une confusion du classifieur parmi les $\mathcal{S}$ classes de S. S'il y a $N_{errConf}$ échantillons de ce type, le *taux d'erreur de confusion* $\tau_{errConf}$ est défini par :

$$\tau_{errConf} = \frac{N_{errConf} \cdot 100}{N_S} \tag{1.8}$$

Il est aussi intéressant d'évaluer le taux de vrai rejet τ_{Rdist} qui détermine l'aptitude du système à bien détecter les formes de type inconnu. Il est défini à partir de $N_{correctsRebut}$ qui est le nombre d'échantillons ayant l'étiquette ω_{inc} et pour lesquels la décision prise par le classifieur est ω_{Rdist}. τ_{Rdist} vaut alors :

$$\tau_{Rdist} = \frac{N_{correctsRebut} \cdot 100}{N_R} \tag{1.9}$$

En plus des critères précédents, le critère de fiabilité τ_{fiab} du classifieur est souvent utilisé lorsque celui-ci implémente le rejet en général (refus de classement et/ou rejet de distance). Il est défini comme étant le taux de reconnaissance sur les individus non rejetés, soit :

$$\tau_{fiab} = \frac{N_{correctsw} \cdot 100}{N - N_{Ramb} - N_{Rdist}} \tag{1.10}$$

où $N_{correctsω}$ est le nombre d'échantillons bien classés appartenant à ω_i, $(i = 1, \ldots, \mathcal{S})$, $N_{Rdist} = N_{fr} + N_{correctsRebut}$ est le nombre d'échantillons pour lesquels le système prend la décision ω_{Rdist} et $N_{Ramb} = N_{Rambω} + N_{RambRebut}$ est le nombre d'échantillons pour lesquels le système prend la décision ω_{Ramb}.

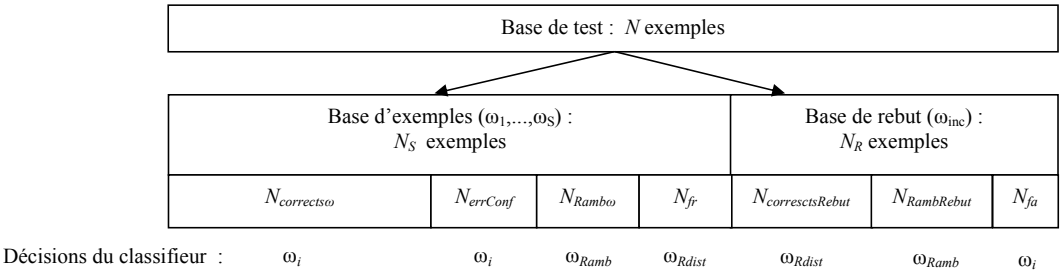

FIG. 1.17 – *Répartition des exemples dans la base de test en fonction des décisions du classifieur : avec refus de classement et rejet de distance.*

NOTE :

En fonction des cadres applicatifs, la terminologie et les critères utilisés peuvent varier. Ainsi, on trouve assez souvent les termes de *vrai négatifs* pour désigner $N_{correctsRebut}$, de spécificité (*specificity*) pour τ_{Rdist}, de vrai positifs pour $N_{correctsω}$, de sensibilité (*sensitivity*) pour $(N_{correctsω} \cdot 100)/N_S)$, de *faux positifs* pour N_{fa} et de *faux négatifs* pour N_{fr} [38].

On remarquera aussi le lien avec le cas général (cf. figure 1.14) qui peut s'établir par : $N_{corrects} = N_{correctsω} + N_{correctsRebut}$ et $N_{err} = N_{errConf} + N_{fr} + N_{fa}$.

1.2.6.3 Autres critères pour caractériser les systèmes de reconnaissance

En dehors des critères de performances et de fiabilité, d'autres éléments sont parfois à prendre en considération, notamment pour la mise au point et l'exploitation du système. Il s'agit d'une tâche complexe et de longue haleine. En effet, selon les approches de classification, il existe généralement un certain nombre de paramètres (liés à l'architecture, à la fonction de décision, à la méthode d'apprentissage) à régler pour obtenir des performances optimales. À moins de posséder des connaissances *a priori* permettant de les obtenir directement ou de bénéficier des avantages d'une modélisation modulaire ou interprétable, il faut souvent procéder par des essais successifs, ce qui est long et fastidieux. Pouvoir caractériser les capacités du processus d'apprentissage en terme de *complexité algorithmique* ou de *durée d'apprentissage* est un élément qui revêt alors une importance certaine, bien qu'il faille la relativiser par rapport aux avancées technologiques qui font sans cesse croître les puissances de calcul.

Lors de l'exploitation du système dans des conditions réelles d'utilisation, d'autres critères peuvent aussi devenir très importants. Ainsi, s'il existe des contraintes sur la vitesse de traitement, comme par exemple pour des applications interagissant directement avec l'utilisateur, le *temps de reconnaissance* devient un critère important. Si l'on

considère l'embarquement du système sur des machines aux ressources limitées (Tablette PC, PDA voire smart phone), la *taille mémoire* du modèle ou d'une façon plus générale le *nombre de paramètres* nécessaires à la modélisation sont des critères de compacité importants à prendre en compte.

1.3 Différentes approches de reconnaissance

Dans ce paragraphe, nous passons en revue quelques exemples de systèmes et de méthodes de reconnaissance couramment utilisés. L'objectif n'est pas de décrire les algorithmes eux-mêmes (le lecteur intéressé pourra pour cela se référer à la littérature) mais plutôt d'énoncer leurs propriétés en fonction des éléments présentés précédemment et notamment de l'architecture. Cela nous permettra de présenter et d'illustrer de façon plus explicite les objectifs et les choix relatifs à notre approche de reconnaissance dans le paragraphe suivant (cf. paragraphe 1.4). Les approches sont présentées dans un ordre qui se veut « croissant » par rapport à la structuration et à l'interprétabilité de la modélisation et donc des connaissances.

Nous nous sommes concentré ici sur les méthodes reposant sur un apprentissage supervisé, permettant d'établir directement une relation entre l'espace des entrées et celui des sorties. Les approches de classification non supervisée utilisées en reconnaissance de formes sont généralement intégrées dans des architectures plus complexes telles que celles mettant en œuvre la compétition de modèles de classes et la combinaison de classifieurs que nous décrivons dans le paragraphe 2.2. Ce sont des outils essentiels pour extraire automatiquement une structure à partir des données et constituent une des bases de notre approche. Nous y reviendrons donc plus en détails dans le chapitre 3. Une présentation générale de ces algorithmes est aussi fournie dans l'annexe B.

1.3.1 Méthodes probabilistes

Les méthodes probabilistes considèrent le problème de classification comme étant un processus stochastique dont les probabilités décrivent le caractère aléatoire. L'objectif de la modélisation est d'établir les probabilités *a posteriori* des classes, c'est-à-dire la probabilité d'obtenir la classe ω_j sachant que la forme e a été observée en entrée du système. Cette probabilité est déterminée à partir de la règle de Bayes :

$$P(\omega_j|e) = \frac{p(e|\omega_j)P(\omega_j)}{p(e)}, \tag{1.11}$$

où $p(e|\omega_j)$ est la densité de probabilité conditionnelle de e, $P(\omega_j)$ est la probabilité *a priori* de la classe ω_j et $p(e)$ est la densité de probabilité de e définie par :

$$p(e) = \sum_{i=1}^{C} p(e|\omega_i)P(\omega_i). \tag{1.12}$$

La difficulté de cette approche réside essentiellement dans la détermination des densités $p(e|\omega_j)$. Celles-ci peuvent être considérées comme étant définies par un ensemble de

paramètres Ω_j [11] à estimer à partir des échantillons de la base d'apprentissage. Il existe plusieurs approches pour parvenir à l'estimation de ces paramètres. Les plus connues sont la méthode Bayésienne et la méthode du maximum de vraisemblance. La différence entre les deux est d'origine conceptuelle. La méthode du maximum de vraisemblance considère Ω_j comme un ensemble de paramètres fixes à estimer à partir des échantillons. La méthode Bayésienne la considère comme une variable aléatoire dont la densité de probabilité *a posteriori* est estimée à partir des données. Cette méthode est beaucoup plus complexe mais fournit en général une meilleure estimation, notamment quand des connaissances *a priori* fiables peuvent être injectées dans la modélisation. Cependant, la méthode du maximum de vraisemblance donne en pratique des résultats presque aussi bons (cela est d'autant plus vrai que la base d'apprentissage est importante) avec une complexité nettement moindre. Pour plus d'informations sur ces méthodes, le lecteur peut se référer à [49].

La décision Bayésienne offre un cadre théorique et un formalisme solides et bien définis qui font toute sa réputation. Cependant, elle repose sur un ensemble de contraintes et d'hypothèses liées au cadre probabiliste qui, si elles s'avèrent fausses, provoquent des dysfonctionnements importants dans le système. De plus, l'abstraction mathématique sur laquelle repose la modélisation n'est pas toujours facile à interpréter ni à manipuler, d'autant plus que le nombre de classes et leurs interactions sont importants.

1.3.2 Les k plus proches voisins

L'approche par les k plus proches voisins (k-PPV) possède au moins deux avantages : elle est très simple et elle ne nécessite pas d'apprentissage. La modélisation utilisée pour la classification est simplement la base d'apprentissage elle-même qui constitue l'ensemble des connaissances du système. Il n'y a donc pas d'architecture à proprement parler. La règle de décision utilisée consiste à attribuer à une entrée e la classe représentée majoritairement dans son voisinage. Ce voisinage est défini par les k plus proches individus de e parmi ceux se trouvant dans la base d'apprentissage. Les seuls paramètres à déterminer sont donc k ainsi que la métrique utilisée pour comparer les individus et trouver les plus proches voisins.

Dans plusieurs cadres applicatifs, cette approche a montré des résultats intéressants. Cependant, elle possède plusieurs inconvénients. Son efficacité dépend directement de la pertinence de la base d'apprentissage et notamment de sa densité dans les différentes régions de l'espace des données. La présence de bruit (données mal étiquetées) est aussi fortement handicapante. Autre inconvénient, la base d'apprentissage entière doit être stockée, ce qui nécessite en général une place mémoire importante. [12] Enfin, la recherche des plus proches voisins est coûteuse et cela d'autant plus que la métrique [13] utilisée est complexe.

11. Il existe aussi des méthodes d'estimations non paramétriques par des noyaux, des histogrammes, des fenêtres (Parzen), etc.).

12. Des techniques de réduction de la base d'apprentissage sont donc souvent nécessaires.

13. Les distances les plus courantes sont reportées dans l'annexe A.

1.3.3 Les réseaux de neurones de type perceptron

Les classifieurs de type réseaux de neurones viennent de recherches orthogonales entre les sciences informatiques et la biologie. Plus précisément, ceux-ci ont été définis à partir d'un modèle de neurone artificiel simulant le mode de fonctionnement du neurone biologique [125]. Du point de vue de la classification, ils reposent sur une modélisation discriminante. Un neurone permet de définir une fonction discriminante linéaire g dans l'espace de représentation E des formes. Cette fonction réalise une combinaison linéaire du vecteur de caractéristiques de la forme e présentée en entrée :

$$g(e) = w^t e + w_0, \tag{1.13}$$

où w est un vecteur de poids de la combinaison linéaire et w_0 est le biais. Ainsi, $g(e) = 0$ définit un hyperplan permettant de séparer E en deux régions de décision. En déterminant les bonnes valeurs de w et w_0, celles-ci peuvent être associées à deux classes pour faire leur discrimination. Pour des problèmes à $\mathcal{S}$ classes, $\mathcal{S}$ fonctions discriminantes sont établies. L'extension à des réseaux de type perceptron multi-couches (PMC) [23] permet d'obtenir des frontières de décision de complexité quelconques[14]. L'apprentissage des poids, est généralement fait par des méthodes comme la rétro-propagation du gradient de l'erreur.

Les PMC ont été très largement utilisés en classification et en reconnaissance de formes, notamment pour leur bonnes performances et leur simplicité. Ils possèdent en outre des propriétés intéressantes. Par exemple, ils offrent la possiblité de faire de l'apprentissage « en-ligne », c'est-à-dire à chaque fois qu'une nouvelle donnée est disponible. En reconnaissance en-ligne de symboles manuscrits, cela peut permettre de faire de l'adaptation au scripteur. En revanche, ils ont aussi leurs points faibles. Ainsi, l'algorithme de rétro-propagation du gradient de l'erreur risque de converger vers un minimum local, donnant une solution sous-optimale. La détermination de l'architecture et des paramètres du réseau est aussi un inconvénient majeur. En effet, avant d'entreprendre l'apprentissage, il faut déterminer le nombre de couches cachées du classifieur ainsi que le nombre de neurones de chacune de ces couches. Aussi, sans connaissances *a priori*, il faut rechercher l'architecture optimale par essais successifs. Un autre inconvénient majeur réside dans le manque d'interprétabilité et de modularité de sa modélisation. Bien que celle-ci soit assez fortement structurée en couches, les liaisons inter-neurones sont tellement complexes qu'il est difficile de déterminer le rôle et l'impact des différents poids (connaissances) et neurones dans le résultat.

1.3.4 Les machines à vecteurs supports

Les machines à vecteurs supports (Support Vectors Machines (*SVM*)) [1, 166] sont des techniques de classification assez récentes qui montrent de très bonnes performances dans de nombreux domaines. Elles reposent elles aussi sur une modélisation discriminante qui s'appuie sur la détermination de *supports* qui sont des formes remarquables de

14. Les PMC sont des approximateurs universels.

la base d'apprentissage. Deux notions clés sont à la base de leur fonctionnement. La première, qui rend les SVM très efficaces en généralisation, repose sur le principe de *marge de classification*. Dans le cas d'un problème à deux classes linéairement séparables, la discrimination optimale est obtenue par un hyperplan qui maximise sa distance (la marge) avec les individus des deux classes les plus proches. Pour obtenir cette marge, une SVM recherche les formes de la base d'apprentissage (les supports) qui sont les plus proches de cet hyperplan, c'est-à-dire les plus difficiles à classer (cf. figure 1.18).

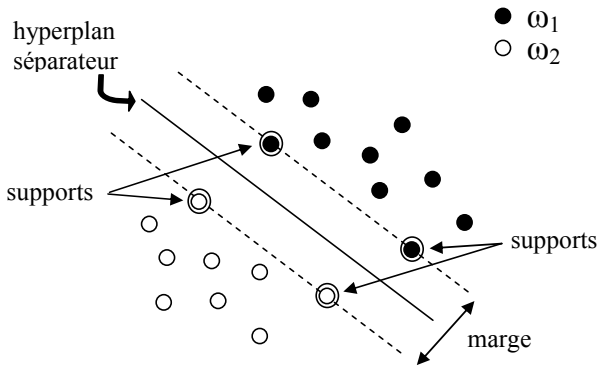

FIG. 1.18 – *Marge et hyperplan séparateur obtenus par une SVM pour un problème à deux classes ω_1 et ω_2.*

La deuxième notion importante est la possibilité d'obtenir des fonctions de décision non linéaires. Pour cela, l'idée consiste à effectuer une transformation non linéaire ϕ de l'espace E dans un espace de plus grande dimension (éventuellement infinie) qui permet de se ramener au cas linéaire. En pratique, pour éviter d'avoir à travailler directement dans un espace de taille éventuellement infinie, les SVM permettent de rester dans l'espace courant E en utilisant des fonctions noyau $K(e, supp_i)$ définies à partir des supports $supp_i$. Ces noyaux peuvent être de plusieurs formes (polynômes, fonctions à base radiales, etc.) mais doivent satisfaire plusieurs critères pour être valides (cf. [110, 34, 131]). Dans le cas de noyaux à base radiale, les SVM s'apparentent aux réseaux de neurones de fonctions à base radiale présentés en 1.3.5. Cependant, la manière dont sont déterminés les supports produit une modélisation et un comportement différents [43, 153] qui peut tourner à l'avantage des réseaux de neurones de fonctions à base radiales [43] quand la taille de la base d'apprentissage devient importante. En effet, les SVM s'appuyant sur les supports difficiles à classer (proches de la marge) produisent des frontières de décision très « découpées » et peuvent ainsi perdre de leur pouvoir de généralisation.

D'une façon générale, les SVM ont aussi quelques inconvénients. Le principal réside certainement dans la difficulté de choisir la fonction noyau la mieux adaptée au problème considéré. De plus, l'algorithme est à l'origine conçu pour traiter des problèmes à deux classes. L'extension aux problèmes multi-classes n'est pas encore clairement établie. Bien souvent, pour des raisons calculatoires, la résolution s'effectue en se ramenant à

des problèmes de classification binaires par des techniques de type « une classe contre les autres » ou « une classe contre une autre » [79, 124]. Un autre problème réside dans la difficulté de l'algorithme à travailler sur des bases d'apprentissage de taille importante. Des méthodes pour diviser la base en sous-bases (*chunking* [166]) ou encore des méthodes de prétraitements pour réduire le nombre d'échantillons [54] doivent alors être utilisées. Enfin, comme la modélisation repose sur les formes les plus difficiles à classer, le nombre de vecteurs supports et donc de paramètres nécessaires est souvent très important. Cet effet est encore accru lorsque des problèmes à plusieurs classes sont traités en les ramenant à des problèmes de classification à deux classes [79]. Cet inconvénient ne fait qu'accroître le manque d'interprétabilité de la modélisation lié à la résolution du problème dans un espace de très grande dimension.

1.3.5 Les réseaux de neurones de fonctions à base radiale

Les réseaux de neurones de fonctions à base radiale (Radial Basis Function Networks : RBFN) [32, 128] s'appuient aussi sur la détermination de *supports* dans l'espace de représentation E. Cependant, à la différence des SVM, ceux-ci peuvent aussi correspondre à des formes fictives. Ils sont associés à une zone d'influence définie par une fonction à base radiale. La fonction discriminante g d'un RBFN à une sortie est définie à partir de la distance de la forme en entrée à chacun des supports et de la combinaison linéaire des fonctions à base radiale correspondantes :

$$g(e) = w_0 + \sum_{i=1}^{Nn} w_i \cdot \phi(d(e, supp_i)), \tag{1.14}$$

où $d(e, supp_i)$ est la distance (au sens d'une métrique à définir) entre l'entrée e et le support $supp_i$, $\{w_0, \ldots, w_{Nn}\}$ sont les poids de la combinaison, ϕ est la fonction à base radiale.

L'apprentissage de ce type de modèle peut se faire en une ou deux étapes. Dans le premier cas, une méthode de type gradient est utilisée pour ajuster l'ensemble des paramètres en minimisant une fonction objectif basée sur un critère comme les moindres carrés. Dans le deuxième cas, une première étape consiste à déterminer les paramètres liés aux fonctions à base radiale (position des supports et zones d'influence associées). Pour déterminer les centres, des méthodes de classification non supervisée sont souvent utilisées. Les poids de la couche de sortie peuvent ensuite être appris par différentes méthodes comme la pseudo-inverse. Les RBFN possèdent alors plusieurs avantages par rapport aux PMC. Tout d'abord, il est possible de déterminer l'architecture (nombre et paramètres des fonctions à base radiale) indépendamment des paramètres de la combinaison linéaire. L'apprentissage des poids consiste alors à résoudre un problème d'optimisation linéaire plus rapide que les méthodes à base de gradient et sans les problèmes de minima locaux.

Du point de vue de la modélisation, celle-ci est beaucoup plus structurée que celle d'un PMC ou d'une SVM en général. la modélisation repose sur une structuration

verticale en deux couches bien distinctes. La première correspond aux fonctions à base radiale qui permettent de localiser des zones de connaissances précises dans l'espace de représentation. Cette forme de modélisation produit ainsi une structuration horizontale importante au niveau de cette couche. Elle permet en outre de déterminer si une forme présentée en entrée est en adéquation avec cette partie du modèle. Si ce n'est pas le cas, c'est-à-dire si elle ne « tombe » dans aucune des zones d'influences associées aux supports, elle peut être rejetée (rejet de distance). Cette particularité a été exploitée notamment en déterminant des facteurs ou degrés de confiance [154, 173, 65]. Enfin, en prenant des précautions particulières sur la forme et la position des fonctions à base radiale, il est possible d'aboutir à une modélisation relativement interprétable du problème en décrivant la structure sous-jacente aux données des différentes classes [15].

La deuxième couche réalise quand à elle une modélisation discriminante pour séparer les classes en effectuant une combinaison linéaire de la couche précédente.

Même s'il subsiste une forte interconnexion entre les différents éléments du réseau, ce type d'architecture est beaucoup plus modulaire et compréhensible que les approches précédentes. Les RBFN peuvent même, sous certaines conditions, être assimilés à des systèmes à base de règles floues [21, 87], formalisme très intéressant pour une représentation plus facilement manipulable et interprétable comme nous allons le voir dans la partie suivante. Les inconvénients principaux des RBFN sont donc directement liés aux choix des fonctions à base radiale et à leur pouvoir de description, aux limites des techniques utilisées pour déterminer leurs paramètres et aux interactions importantes de celles-ci au travers de la couche de poids en sortie.

1.3.6 Les systèmes d'inférences floue

Les systèmes à base de règles sont très souvent utilisés en intelligence artificielle notamment pour la mise en œuvre de systèmes experts. Mais ils sont aussi couramment employés en reconnaissance et en classification, notamment quand le mécanisme de décision doit être interprétable pour l'utilisateur.

Une règle de décision en reconnaissance de formes se présente sous la forme d'une formule en logique des propositions :
« SI la forme e satisfait la condition $Cond_i$ ALORS la classe est ω_j ».
La prémisse de la règle correspond à une condition que doit satisfaire la forme pour que la conclusion soit valide. Il s'agit d'une proposition logique qui peut être simple ou complexe et qui porte en général sur la valeur de certains attributs de la forme. Cette prémisse est mise en relation avec la conclusion (une autre proposition logique) par une implication. En définissant plusieurs de ces règles, il est possible de décrire les concepts associés aux différentes classes. Des mécanismes de raisonnement comme la déduction peuvent ensuite être mis en œuvre pour prendre des décisions.

Les *systèmes d'inférence floue* (SIF) étendent les principes des systèmes à base de règles classiques en modélisant les imperfections liées aux connaissances manipulées grâce aux outils de la théorie des sous-ensembles flous. Les mécanismes de raisonnement résultants sont ainsi plus robustes et plus proches de la réalité (cf. paragraphe 2.1 pour

plus de détails sur les principes de la logique floue et des SIF).

L'avantage des SIF est qu'ils représentent un formalisme pour la décision (et donc la reconnaissance) qui est facilement interprétable et manipulable. Ils permettent donc de traduire des relations simples ou complexes entre les connaissances et les classes. Même s'il est ainsi possible de produire une modélisation interprétable du problème, cet aspect dépend avant tout de la façon dont sont extraites et organisées les connaissances. Si l'on souhaite obtenir une certaine lisibilité tout en utilisant un apprentissage automatique, les algorithmes utilisés doivent refléter les propriétés des formes dans l'espace de représentation.

1.3.7 Les arbres de décision

Les arbres de décision [17, 97, 129, 150, 188] sont parmi les méthodes de classification les plus anciennes et les plus intuitives. Leur modélisation repose sur une structuration précise des connaissances sous la forme d'un graphe orienté (un arbre). Cette organisation les rend particulièrement simples à utiliser et à interpréter. C'est pourquoi cette approche a connu un vif succès dans différents domaines comme la fouille de données [17, 38, 37] ou encore l'aide au diagnostic et les systèmes d'expertise (médicale, financière, etc.) [129].

En reconnaissance de formes, la structure d'arbre peut être élaborée manuellement pour opérer une modélisation explicite voir structurelle des formes. Lorsqu'un apprentissage automatique est effectué, les informations les plus pertinentes permettant de partitionner l'espace de représentation en régions de décision représentatives d'une classe sont extraites de la base d'apprentissage. Le partitionnement final est obtenu en utilisant une stratégie de type « diviser pour régner » afin de ne pas considérer le problème de manière « globale » mais au contraire pour le subdiviser en sous-problèmes plus aisés à traiter. Pour cela, des partitionnements récursifs de l'espace sont faits en déterminant des conditions (tests) à satisfaire sur les différents attributs. Le choix des conditions est fait en fonction de leur pouvoir de discrimination, ce qui permet de les organiser hiérarchiquement, rendant ainsi le processus de décision plus fiable et plus robuste.

Une particularité de ce principe de modélisation est qu'il est possible de mettre « à plat » la structure arborescente pour pouvoir travailler directement sur une base de règles de décision. Ce formalisme est plus lisible et facile à manipuler que la structure d'arbre en elle-même.

Le principal inconvénient des arbres de décision réside dans leur difficulté à traiter les données numériques. Dans ce cas, le mécanisme de partitionnement repose sur des tests à partir de seuils sur les attributs. Ces seuils rendent la modélisation particulièrement sensible au bruit et à la variabilité des formes en entrée. C'est pourquoi ils sont plus souvent utilisés dans des domaines où l'espace de représentation est représenté par des attributs symboliques. Cependant, l'introduction d'une représentation par sous-ensembles flous a permis d'élaborer des arbres de décision flous, particulièrement adaptés et plus efficaces pour traiter les données numériques et symboliques sujettes aux imprécisions. Nous étudierons ceux-ci en détails dans la partie 3.4.

1.4 Vers une approche générique pour une modélisation explicite centrée sur les connaissances

Les approches de reconnaissance présentées dans le paragraphe précédent ont montrées de bonnes performances pour un grand nombre d'applications. Cependant, leurs aptitudes face à des problèmes complexes comme celui de la reconnaissance de formes manuscrites restent souvent limitées sur au moins un aspect. Il peut s'agir des performances et/ou de la fiabilité qui restent insuffisantes, d'un paramétrage du système difficile à obtenir à cause du manque d'interprétabilité de la modélisation, des ressources nécessaires (mémoire et processeur) trop importantes, etc. La conséquence générale est que leur configuration (ou adaptation) à un contexte applicatif particulier s'avère souvent difficile.

De nouvelles approches ont donc été développées afin de spécialiser les systèmes existants en modifiant leur structure, en faisant collaborer plusieurs classifieurs ou encore en élaborant des systèmes complètement dédiés à un problème de reconnaissance précis et ce afin d'obtenir les propriétés souhaitées. Le système *ResifCar* [14], que nous décrivons dans le paragraphe suivant, est un exemple particulièrement intéressant qui a été une source d'inspiration importante pour notre approche. Ce système a été spécifiquement conçu pour la reconnaissance de lettres manuscrites de façon à pouvoir faire face à la grande variabilité des caractères. La modélisation sur laquelle il repose permet en outre de faire évoluer le classifieur pour l'adapter à différents contextes d'utilisation. Ce sont là deux atouts indéniables pour une approche de reconnaissance de formes. Cependant, le système étant dédié, il est difficile de le réexploiter pour la résolution d'autres problèmes.

La question qui nous intéresse alors est la suivante :

> Est-il possible de concevoir une approche de reconnaissance suffisamment ouverte pour pouvoir bénéficier à la fois de la généricité des méthodes reposant sur une modélisation « abstraite » du problème (PMC, RBFN, SVM, etc.) tout en offrant des possibilités d'adaptation suffisantes aux différents contextes d'utilisation pour parvenir à des propriétés proches de celles que l'on obtiendrait avec une approche dédiée ?

Dans cette thèse, nous essayons d'apporter un début de réponse en proposant une méthodologie de conception qui s'articule autour de l'extraction de connaissances, de leur organisation sous la forme d'une modélisation fortement structurée ainsi que de leur formalisation de la façon la plus compréhensible possible pour le concepteur du système.

Dans cette partie, nous décrivons dans un premier temps les principes et mécanismes qui font l'originalité et l'intérêt de *ResifCar* et qui nous ont guidés dans nos travaux. Nous donnons ensuite de façon explicite les objectifs de notre approche avant de présenter les grandes lignes de celle-ci.

1.4.1 Le système *ResifCar*

ResifCar est un classifieur dédié pour la reconnaissance en-ligne de caractères latins manuscrits. Il a été élaboré au cours de travaux précédents par Anquetil dans le cadre de recherches sur la reconnaissance en-ligne de mots manuscrits [13, 16]. Le système mis au point repose sur une étude approfondie de la façon dont les caractères sont tracés, aboutissant à une modélisation structurelle et explicite de ceux-ci. Les objectifs étaient de concevoir un système de reconnaissance performant et capable d'appréhender la grande variabilité des caractères de façon à ce que l'utilisateur ne soit pas contraint dans son style d'écriture.

D'un point de vue théorique, l'approche se fonde sur une modélisation dans laquelle les connaissances sont structurées à la fois horizontalement et verticalement. Les classes sont modélisées de façon explicite par des modèles eux-même structurés hiérarchiquement sur plusieurs niveaux. Les niveaux supérieurs décrivent les propriétés fondamentales des formes qui sont les plus stables et les plus robustes. Les niveaux inférieurs opèrent quant à eux une description plus fine d'éléments plus caractéristiques mais aussi moins stables. Pour chacun des niveaux, les éléments de description sont représentés par des prototypes extraits automatiquement à partir d'une base d'apprentissage et formalisés par des sous-ensembles flous (cf. partie 2.1.1). Cela les rend robustes face à la variabilité des formes et au bruit tout en leur conférant une bonne interprétabilité. Le système peut ainsi être plus facilement optimisé et adapté au contexte applicatif par un expert et en fonction de son comportement en phase d'exploitation. Il est en effet possible de rajouter/supprimer des modèles de classe et de modifier la forme et la position des prototypes flous.

La fonction de décision associée à cette modélisation s'appuie sur la structuration horizontale et verticale des connaissances. Les modèles de classes sont mis en compétition les uns avec les autres, niveaux par niveaux en utilisant le formalisme interprétable des systèmes d'inférence floue. Ce processus de décision hiérarchique permet de n'utiliser que les connaissances nécessaires, accélérant les traitements et limitant les effets de bords.

En dehors de cet aspect théorique de l'approche, le caractère dédié de *ResifCar* vient de la structure des modèles qui est définie *a priori* en s'appuyant sur les propriétés morphologiques des différents allographes des caractères à reconnaître. Les modèles se présentent sous la forme d'une arborescence à trois niveaux correspondant à la hiérarchisation des connaissances des plus stables et pertinentes aux plus fines et sensibles évoquée ci-dessus. C'est hiérarchisation issue de l'expertise de l'écriture manuscrite qui permet de faire une discrimination robuste des caractères. D'une façon plus précise, chaque niveau du modèle caractérise un type de primitive correspondant à différents éléments du tracé manuscrit (cf. figure 1.19) :

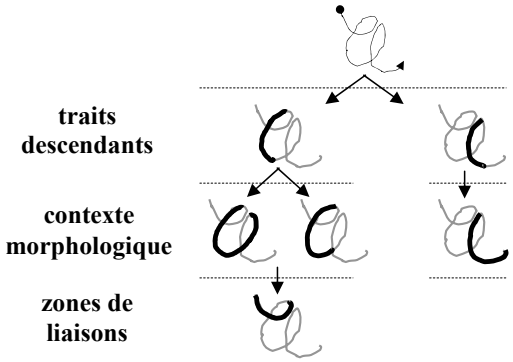

FIG. 1.19 – *Niveaux de descriptions associés au modèle d'un allographe du caractère 'a' dans le système ResifCar.*

- **traits descendants :**
  Le premier niveau modélise les traits descendants[15] les plus significatifs des caractères. Ceux-ci sont reconnus pour être des éléments stables du tracés, de par leur nombre, leur forme et leur position. Ils représentent donc la structure fondamentale des caractères.
- **contexte morphologique :**
  Afin d'obtenir une modélisation plus précise, les contextes morphologiques associés à chacun des traits descendants du niveau précédent sont ensuite caractérisés. Ces primitives correspondent au trait descendant lui même prolongé par une portion du tracé avant et après. Cette portion est déterminée essentiellement à partir de points d'ancrage visuel, tels que les points de rebroussement, les points multiples (boucles par exemple), etc. Comme le montre la figure 1.19, pour un même trait descendant, plusieurs contextes morphologiques peuvent être modélisés.
- **zones de liaisons :**
  Le troisième niveau de modélisation décrit les zones du tracé correspondant aux signes diacritiques (points sur les 'i' et 'j', barre du 't', etc.) et aux zones de liaisons entre les contextes morphologiques des traits descendants. Ces dernières peuvent être des liaisons inter-caractères ou intra-caractères. L'ensemble de ces primitives peuvent s'avérer très discriminantes pour distinguer certaines lettres mais elles sont aussi moins robustes et moins fiables que celles des niveaux précédents. C'est pourquoi elles ne sont utilisées que pour affiner les décisions lorsque cela est nécessaire.

L'intérêt de la modélisation de *ResifCar* a pu être validé récemment au cours d'un transfert industriel [14]. Le système a en effet été embarqué sur des téléphones portables offrant la possibilité à l'utilisateur de saisir ses informations (numéros de téléphones,

15. Rappelons que ce système fonctionne pour la reconnaissance en-ligne.

adresses, textos, *etc.*) en les écrivant naturellement à l'aide d'un stylet sur l'écran tactile. Ces téléphones bon marché n'ont que très peu de ressources disponibles. Ils utilisent notamment un processeur de type ARM7 TDMI cadencés à 13Mhz qui rendrait impossible l'utilisation de systèmes de reconnaissance reposant sur des approches classiques. Dans le cas de *ResifCar*, grâce à la modélisation explicite et intelligible des connaissances, il a été possible d'une part, d'effectuer une optimisation algorithmique ciblée pour réduire la place mémoire occupée ainsi que les temps d'exécution et d'autre part, d'opérer une optimisation progressive des performances en terme de taux de reconnaissance par une série de tests d'exploitation mettant en relief les connaissances à l'origine des confusions inter-classes.

Le système *ResifCar* constitue donc un bel exemple de ce qu'il est possible de faire pour des problèmes complexes comme en reconnaissance de formes manuscrites. C'est pourquoi nous nous sommes inspiré de ses principes pour essayer d'en déduire une méthodologie de conception plus générique pour la reconnaissance de formes.

1.4.2 Les objectifs de notre approche

L'approche de reconnaissance de formes *Mélidis* présentée ici s'attache à résoudre plusieurs objectifs qui sont rarement satisfaits en même temps par un même classifieur. L'idée n'est pas de concevoir un système « universel[16] » mais plutôt de s'appuyer sur une méthodologie conceptuelle suffisamment riche pour pouvoir aborder un grand nombre de problèmes. Cela passe nécessairement par des compromis mais si le système est suffisamment ouvert, il doit permettre au concepteur de mettre l'accent sur ses objectifs principaux, en fonction du contexte d'utilisation.

Les objectifs généraux de l'approche sont présentés ci-dessous en les distinguant bien les uns des autres pour plus de clarté. Cependant, le lecteur ne doit pas perdre de vue qu'ils sont fortement interdépendants.

1.4.2.1 Généricité

Étant donné le nombre important de problèmes de reconnaissance qu'il peut être nécessaire de traiter, bénéficier d'une approche de reconnaissance générique est un atout majeur en terme de coût de conception. Elle permet de limiter l'intervention humaine, dégageant un gain de temps important. Pour satisfaire cet objectif, la structuration de la modélisation (l'architecture) du classifieur doit être indépendante des problèmes de reconnaissance traité, au contraire d'approches comme *ResifCar* par exemple. De plus pour pouvoir s'adapter aux différents niveaux de complexité, elle doit être suffisamment modulaire. En outre, les connaissances nécessaires à l'élaboration du modèle doivent être extraites de façon automatique à partir d'une base d'apprentissage. Les seules contraintes imposées ici sur le contenu et la forme de cette base sont que les échantillons soient étiquetés (supervision), représentés par un vecteur de caractéristiques à valeurs numériques et que ce vecteur soit de taille fixe.

16. Est-il réellement possible de concevoir un tel système?

1.4.2.2 Performances

Les performances du système de reconnaissance restent un des objectifs privilégiés afin de permettre son intégration dans des applications réelles. Ces performances se traduisent directement par les capacités de reconnaissance du classifieur (taux de reconnaissance) qui pourraient constituer l'objectif de performances en soi. Mais celui-ci dépend directement de sous-objectifs plus précis dont la réalisation unitaire permet d'améliorer les performances d'ensemble. Il s'agit de :

- *La robustesse face à la variabilité des formes :*
  Dans la plupart des problèmes de reconnaissance, deux entrées du système représentant une même forme ne seront pas strictement identiques. Cette variabilité peut avoir plusieurs origines. Elle peut provenir des formes elles mêmes, comme c'est le cas en reconnaissance d'écriture manuscrite par exemple (variabilité intra-scripteur). Elle peut aussi résulter du processus d'acquisition de la forme, pendant lequel du bruit peut venir perturber le signal. Afin de pouvoir faire face à ces imprécisions relatives aux données, la modélisation sur laquelle repose le système doit être suffisamment robuste pour absorber la variabilité des formes. Comme les données traitées sont numériques, la modélisation doit donc s'affranchir de frontières de décision strictes trop souvent contraignantes et sources d'instabilités.

- *La fiabilité des décisions :*
  Le processus de décision du système doit fournir des réponses fiables notamment en vue d'intégration dans des systèmes de traitement automatiques. Cet objectif est intimement lié à celui de robustesse : moins le classifieur sera sensible aux variations des données et plus il sera fiable. Cependant, la fiabilité peut aussi être améliorée par la mise en œuvre d'une gestion du refus de classement et du rejet distance qui permettent de ne pas considérer les cas trop difficiles ou incohérents avec la modélisation.

- *L'adaptabilité et l'optimisation :*
  Après l'apprentissage du modèle, le comportement du classifieur ne sera pas toujours entièrement satisfaisant pour l'intégration dans son contexte applicatif. De plus, ce dernier peut évoluer au cours du temps. Il est donc important de pouvoir adapter la modélisation afin d'améliorer les performances générales du système pour son exploitation. Par exemple, pour le traitement de l'écriture manuscrite, il semble intéressant de pouvoir adapter un reconnaisseur au style d'écriture d'un scripteur. Cette adaptation peut s'effectuer de différentes manières. Il peut s'agir de faire un ré-apprentissage complet en utilisant des informations *a priori* supplémentaires (base d'apprentissage spécifique, modification de paramètres, etc.) ou encore de faire des adaptations du système complet ou de sous-parties de celui-ci en modifiant « manuellement » ou automatiquement la modélisation ou la fonction de décision. Les possibilités d'adaptation dépendent directement de la modularité de l'architecture ainsi que du formalisme utilisé pour représenter les connaissances.

1.4.2.3 Interprétabilité

Afin de rendre l'optimisation finale du classifieur et son adaptation à une situation particulière plus facile, le concepteur du système doit pouvoir comprendre et maîtriser le fonctionnement de celui-ci. Aussi, pour que le processus de conception soit le plus efficace possible, il faut que la modélisation et le processus de décision associé soient suffisamment compréhensibles et interprétables pour permettre au concepteur de déterminer précisément les maillons à améliorer. Il pourra alors injecter de nouvelles connaissances dans le système afin de corriger son fonctionnement. L'objectif est donc d'aboutir à une modélisation suffisamment structurée et explicite.

1.4.2.4 Compacité

La compacité (et donc la concision) de la modélisation est un autre facteur qui va dans le sens de l'interprétabilité puisqu'un système avec peu de paramètres sera plus facile à analyser qu'un système complexe. Cet objectif est aussi conforme à la problématique générale de la reconnaissance de formes puisqu'il est reconnu que de deux classifieurs qui ont les mêmes performances sur la base d'apprentissage, le plus simple (celui qui utilise le moins de paramètres) devrait avoir de meilleures performances en généralisation. Enfin, dans certains cadres applicatifs où peu de ressources sont disponibles, la compacité devient un facteur critique. Nous avons déjà vu un exemple avec le système *ResifCar*. Même si ce n'est pas systématique, la tendance actuelle est fortement orientée vers l'intégration de systèmes informatiques de plus en plus évolués sur des machines de poches. Il nous paraît donc important de ne pas élaborer notre approche sans considérer la taille de la modélisation. Il est donc essentiel que, dès le départ, le nombre de paramètres nécessaires soit le plus réduit possible. Si les objectifs d'adaptabilité et d'interprétabilité sont eux aussi satisfaits, ils devraient ensuite permettre de réduire au mieux les coûts d'exploitation du reconnaisseur.

1.4.3 La philosophie de notre méthode

Comme nous l'avons déjà souligné, la problématique de la reconnaissance de formes manuscrites est à l'origine de notre démarche. Elle est en effet suffisamment riche et intéressante pour pouvoir être généralisée à de nombreux problèmes de reconnaissance de formes en général. Ceci nous a permis de déterminer les objectifs de notre approche.

Les recherches faites pour le système *ResifCar* fournissent en outre un exemple de conception capable de répondre à une grande partie de nos besoins pour un problème spécifique. Elles constituent donc une source d'inspiration importante et nous avons essayé d'en extraire les principes susceptibles de satisfaire nos attentes, en les reconsidérant hors contexte pour pouvoir les reformuler et les intégrer dans une méthodologie pour la conception d'un classifieur générique.

Le système final repose essentiellement sur l'utilisation de techniques de modélisation et d'apprentissage issues du domaine du *soft computing* [184], en adoptant notamment un formalisme à base de systèmes d'inférence floue pour la décision.

1.4.3.1 Architecture générale

L'architecture générale du système peut être synthétisée comme à la figure 1.20. Les formes sont traitées par deux modules hiérarchisés. Le premier, qualifié de *module de description intrinsèque*[17] *par prototypes*, caractérise les propriétés remarquables et stables des classes par des prototypes. Le second, qualifié de *module de discrimination focalisée*, se concentre ensuite sur les propriétés plus spécifiques des formes afin de résoudre les confusions mise en évidence par la description intrinsèque. La classification résulte de la combinaison des deux modules par un *module de fusion*.

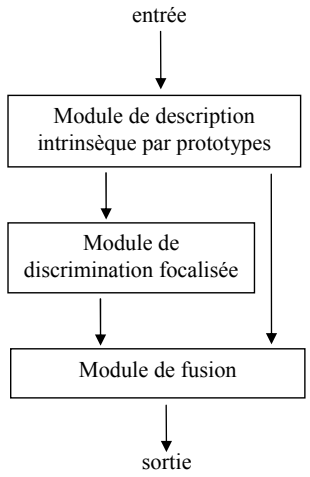

FIG. 1.20 – *Schéma-bloc du système Mélidis.*

Pour parvenir à l'élaboration de cette architecture, notre démarche a été guidée par la nature, le rôle et les propriétés des connaissances dans un système de reconnaissance de formes. Notamment, chaque module ainsi que le lien les unissant s'appuie fortement sur la structure des données (leurs propriétés) dans l'espace de représentation. Il s'agit donc d'un mode de combinaison de classifieurs un peu particulier que nous qualifions ici de *combinaison structurée guidée par les données*.

1.4.3.2 Méthodologie de conception

En se basant sur l'expérience de *ResifCar*, deux notions nous paraissent particulièrement importantes pour la réalisation de nos objectifs : la nature et le formalisme des connaissances constituant la modélisation ainsi que leur organisation (qui correspond à l'architecture du système). C'est en effet la manière d'extraire celles-ci, de les représenter et de les mettre en relation qui confère au système ses propriétés.

La méthodologie présentée ici s'attache donc à déterminer quelles sont les connaissances utiles, comment les extraire, les formaliser et les structurer au mieux. Dans ce

17. intrinsèque : caractère qui appartient à un objet en lui-même, et non dans ses relations avec un autre.

contexte, les connaissances de nature *intrinsèque* aux classes et les connaissances de nature *discriminante* entre classes jouent des rôles fondamentaux de par leur complémentarité.

Les connaissances intrinsèques permettent de faire une modélisation descriptive robuste et explicite des classes. Elles sont aussi tout à fait adaptées pour mettre en évidence la présence de sous-classes et opérer une structuration/décomposition horizontale du problème. De plus, comme elles caractérisent les propriétés générales et stables des classes, elles peuvent être utilisées pour faire du rejet de formes inconnues. Enfin, elles permettent de mettre en évidence les sources de confusion importantes entre les classes qu'il faudra résoudre.

Les connaissances discriminantes servent à distinguer différentes entités (classes, sous-classes voir sous-ensembles de formes) et n'ont donc de sens que dans le cadre de cette comparaison. Elles s'appuient sur des propriétés très spécifiques des formes qui ne transparaissent que dans des sous-espaces particuliers. Elles sont donc beaucoup plus sensibles que les connaissances intrinsèques, notamment face à la variabilité des formes et au bruit. De plus, elles n'ont pas toutes le même pouvoir de discrimination. Toutes ces raisons font qu'elles doivent être utilisées avec parcimonie et de préférence dans un contexte bien défini comme les différents types de confusion mis en évidence par les connaissances intrinsèques.

Pour que la modélisation de ces deux natures de connaissances soit à la fois robuste, compacte et interprétable nous utilisons, comme dans le système *ResifCar*, un formalisme à base de sous-ensembles flous. Ceux-ci sont extraits automatiquement en s'appuyant sur les regroupements naturels des formes dans l'espace de représentation [18], c'est-à-dire sur leurs propriétés dans cet espace. Cela permet d'extraire des connaissances plus stables et plus explicites, ce qui va dans le sens de l'interprétabilité au sens où nous l'avons définie dans le paragraphe 1.2.4.5.

En s'appuyant sur ces éléments, la modélisation a été organisée autour d'une structuration verticale sur deux niveaux (cf. figure 1.21). Le premier niveau, qualifié de *niveau de modélisation intrinsèque*, est structuré horizontalement. Il définit des modèles flous intrinsèques à chacune des classes ω_i sous forme de prototypes, révélant ainsi leur caractère multimodal. Ce niveau s'inspire donc directement des principes de modélisation utilisés dans le système *ResifCar*. Il est aussi utilisé pour décomposer le problème en sous-problèmes représentés par les formes des différentes classes qui se ressemblent d'après les modèles intrinsèques, et qui sont donc sources potentielles de confusions. Il s'agit de la (*décomposition guidée par les données*). Le deuxième niveau, qualifié de *niveau de modélisation discriminante*, effectue une modélisation discriminante focalisée sur chacun de ces sous-problèmes à l'aide d'arbres de décision flous. Il est donc fortement structuré, à la fois horizontalement et verticalement, structuration qui là encore est inspirée des modèles de allographes utilisés dans *ResifCar*. Cependant, alors que dans *ResifCar* la discrimination est faite de manière implicite par le choix *a priori* des connaissances modélisées à chacun des niveaux et la compétition des différents modèles,

18. Rappelons ici que nous considérons que l'espace de représentation est numérique.

dans *Mélidis*, les connaissances discriminantes sont extraites et formalisées explicitement dans le niveau de modélisation discriminante.

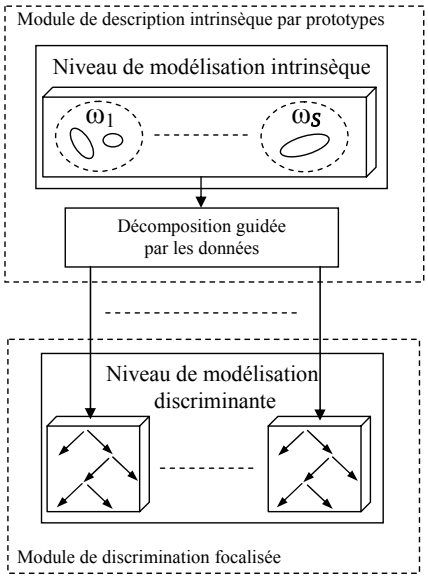

FIG. 1.21 – *Structuration des connaissances dans le système Mélidis.*

1.4.3.3 Mécanisme de décision

Pour la reconnaissance, la fonction de décision utilise les modèles flous préalablement extraits en essayant de bénéficier au mieux de leur complémentarité. Comme nous l'avons déjà dit au paragraphe 1.2.4.2, une modélisation par description explicite des classes peut aussi être utilisée pour discriminer les classes. Ainsi, comme les connaissances extraites par la modélisation intrinsèque sont différentes de celles utilisées par la modélisation discriminante, elles peuvent apporter un complément d'information intéressant pour la classification. Nous exploitons ici cette particularité en définissant trois niveaux de décision (cf. figure 1.22). Le premier exploite ($\Rightarrow$) les connaissances du niveau intrinsèque pour effectuer une *pré-classification* de la forme. Le second utilise ($\Rightarrow$) ensuite les connaissances discriminantes en adéquation avec la forme pour faire la *classification principale*. Cette sélection des connaissances se fait grâce au même mécanisme de décomposition que celui utilisé pour la modélisation (cf. figure 1.21). Enfin, le troisième niveau combine les deux précédents pour fournir la *classification finale*. Pour que cette combinaison soit facilitée mais aussi pour améliorer la compréhension des mécanismes de décision, les deux premiers niveaux de décision sont formalisés de façon homogène par des systèmes d'inférence floue (SIF), principe repris là aussi de *ResifCar*.

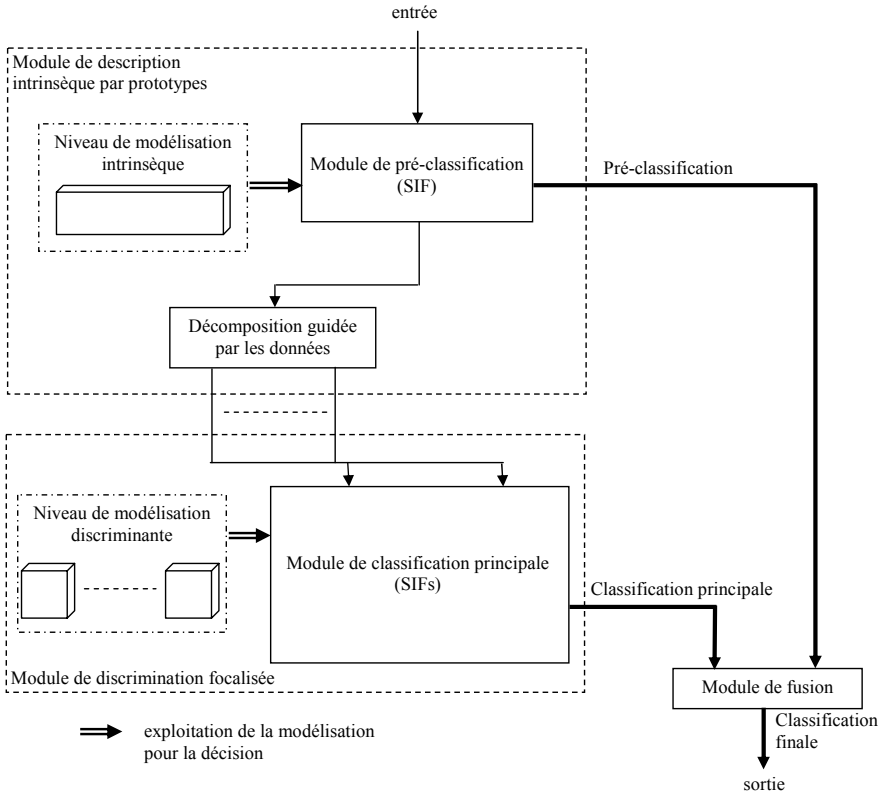

FIG. 1.22 – *Mécanisme de décision du système Mélidis.*

1.5 Bilan

Dans ce chapitre, nous avons présenté la problématique de la reconnaissance de formes manuscrites pour en extraire un ensemble de difficultés générales à la reconnaissance de formes. Nous avons ensuite présenté les mécanismes sur lesquels reposent les systèmes de reconnaissance en les abordant du point de vue des connaissances qu'ils renferment. Cette vision permet de se focaliser sur l'architecture, sa structuration, et les propriétés qui en découlent et de les confronter aux objectifs que le concepteur souhaite atteindre (description et/ou discrimination des classes, robustesse, fiabilité, interprétabilité, etc.).

Nous avons finalement terminé en présentant les grandes lignes de notre méthode, pour la conception d'une architecture générique pour la reconnaissance de formes. Celle-ci a été élaborée de façon à offrir une bonne souplesse et un certain nombre de fonctionnalités nécessaires pour pouvoir aborder différents types de problèmes dans différents contextes d'utilisation. Elle repose notamment sur les principes de structuration, de complémentarité entre description explicite et discrimination des classes ainsi que sur la notion d'interprétabilité du point de vue du concepteur.

Avant d'aborder plus précisément les principes liés à la conception de notre approche ainsi que les mécanismes d'apprentissage et de décision associés, le chapitre suivant présente deux points fondamentaux. Dans un premier temps, les notions élémentaires de la logique floue sont rappelées. Celles-ci permettent de représenter et de manipuler des connaissances qui sont par nature imprécises et sujettes à la variabilité. Dans un deuxième temps, les mécanismes de combinaison de classifieurs sont décrits et nous montrons comment ceux-ci peuvent s'inscrire dans un processus de structuration de la modélisation d'un problème de reconnaissance.

Chapitre 2

Représentation de connaissances imprécises et architecture structurée par multiples classifieurs

Comme tout système de traitement, les classifieurs utilisés en reconnaissance de formes sont amenés à effectuer des tâches complexes (notamment lors de l'apprentissage) à partir de données réelles.

Pour que le système soit opérationnel en phase d'exploitation, il est primordial que la nature des données et notamment leurs imperfections soit correctement prises en compte. De nombreux systèmes informatiques ont été et sont encore établis en partant du principe que les données et l'information qu'elles véhiculent sont fiables et précises. Les éventuelles imperfections sont alors supposées « insignifiantes » au regard de la tâche à accomplir. Cette approximation permet de simplifier l'élaboration des systèmes ainsi que la modélisation sur laquelle ils reposent. Malheureusement, cette hypothèse très forte s'avère parfois fausse, et peut être à l'origine de dysfonctionnements importants. Ce décalage entre la théorie et la pratique n'est pas le seul fait des systèmes de traitement de l'information. Citons par exemple le cas de l'élaboration des semi-conducteurs en physique. Les propriétés de ces composants ont été établies à partir de modèles théoriques reposant sur des structures moléculaires pures de composés comme le germanium et le silicium. Les premières tentatives d'exploitation de ces modèles furent des échecs car ils ne correspondaient pas à la réalité. Le comportement du composant était en effet perturbé par les impuretés présentes dans les matériaux. La difficulté à prendre en compte ces imprécisions dans les modèles ont amené les physiciens à chercher des processus pour éliminer les impuretés du germanium, pari réussit puisque sa pureté atteint aujourd'hui des taux proches de 100 %. Mais dans les systèmes de traitement, il n'est pas toujours possible d'éliminer les imperfections, notamment lorsque celles-ci sont intrinsèquement liées au phénomène considéré. Il faut alors adopter une stratégie différente telle que leur prise en compte de façon explicite [47].

Si on considère à présent les traitements eux-mêmes, lorsque ceux-ci s'avèrent trop difficiles à réaliser par un seul module, une solution consiste à en faire collaborer plu-

sieurs. Ils peuvent alors être dédiés à la réalisation de sous-tâches spécifiques afin de réduire la complexité ou bien effectuer les mêmes traitements mais d'une façon différente afin d'essayer d'augmenter les performances globales de l'ensemble.

Dans la première partie de ce chapitre, nous présentons les principes généraux de la *Théorie des sous-ensembles flous* qui permettent de représenter des connaissances imprécises d'une façon compacte et intuitive facilitant leur compréhension. Ce formalisme peut ensuite être utilisé pour faire des raisonnements robustes à partir de données réelles, ce qui nous intéresse particulièrement ici dans le cadre de notre approche de reconnaissance de formes.

Dans un second temps, nous abordons le problème de la réalisation de tâches complexes en reconnaissance au travers des principes de la combinaison de classifieurs. Dans le cadre de notre problématique, nous proposons un nouveau schéma de combinaison, la *combinaison structurée guidée par les données*. Celle-ci s'appuie sur les propriétés des données dans l'espace de représentation de façon à organiser les différents modules du système pour conférer à l'ensemble une plus grande robustesse, une meilleure lisibilité et une plus grande modularité pour la mise au point et la maintenance.

2.1 Représentation de connaissances imprécises

Lorsque l'on souhaite modéliser un problème qui est par nature soumis aux imprécisions et à l'incertitude, il est important de prendre en compte ces imperfections pour que l'exploitation du modèle soit la plus robuste possible. Dans ce cadre, les probabilités constituent un premier outil de représentation permettant de gérer certaines formes d'incertitudes. Elles peuvent être utilisées pour traduire le caractère aléatoire des phénomènes observés (approche fréquenciste) ou encore pour représenter l'incertitude liée à des connaissances *a priori* (approche subjectiviste) [24]. Mais les approches probabilistes sont généralement contraintes par les axiomes et les hypothèses sur lesquels repose la théorie et limitées par la difficulté d'obtenir de « bonnes » estimations. De plus, les probabilités ne permettent pas de traiter les imperfections liées aux imprécisions, phénomène particulièrement présent dès lors que des données du monde réel sont manipulées.

Pour faire face à ces limitations, Zadeh a introduit en 1965 la *Théorie des sous-ensembles flous* [180] et en 1978 la *Théorie des possibilités* [183]. Alors que la première vise à gérer les imprécisions liées aux observations, la seconde permet en plus de prendre en compte les incertitudes et notamment celles qui ne sont pas de nature probabiliste. Il existe aussi d'autres théories permettant de raisonner avec les deux types d'imperfections, telles que la *Théorie de l'évidence* de Shafer [155].

Dans cette étude, nous nous sommes restreint à la seule prise en compte des imperfections liées aux imprécisions dans les données. Nous ne présentons donc ici que les principes de la théorie des sous-ensembles flous ainsi que la façon dont ils peuvent être utilisés dans un système de reconnaissance. L'extension à la gestion des incertitudes est cependant une perspective intéressante qui reste à étudier.

2.1.1 Théorie des sous-ensembles flous

La théorie des sous-ensembles flous est une extension de la théorie ensembliste qu'elle généralise pour pouvoir représenter les imprécisions des ensembles manipulés. Nous ne donnons ici que les bases nécessaires à notre étude. Le lecteur intéressé pourra se reporter aux nombreux ouvrages sur le domaine [26, 27] dont ce résumé s'inspire.

2.1.1.1 Ensembles, sous-ensembles et sous-ensembles flous

Dans la théorie ensembliste classique, un sous-ensemble d'un ensemble de référence est constitué des éléments qui possèdent une ou plusieurs propriétés communes caractéristiques de ce sous-ensemble. Chaque élément e de l'univers de référence E est ainsi caractérisé par son appartenance ou sa non appartenance aux sous-ensembles E_i définis. Cela s'exprime par la *fonction caractéristique* $\varphi_{E_i} : E \to \{0,1\}$:

$$\varphi_{E_i}(e) = \begin{cases} 1 & si\ e \in E_i \\ 0 & sinon \end{cases} \tag{2.1}$$

Par exemple, si E est l'ensemble des tailles des personnes vivant en France, le sous-ensemble E_{grand} correspondant aux grandes tailles est défini par :

$$\varphi_{E_{grand}}(e) = \begin{cases} 1 & si\ Taille(e) \geq\ 180cm \\ 0 & sinon \end{cases} \tag{2.2}$$

Ce concept ce traduit par le graphe de la figure 2.1

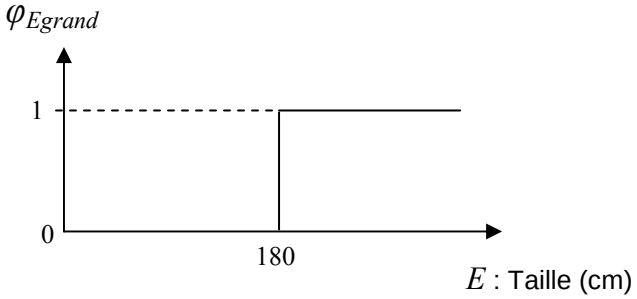

FIG. 2.1 – *Fonction caractéristique de* E_{grand}

Les limites de cette représentation apparaissent ici très clairement. La grande taille d'une personne est un concept que l'être humain est capable de formuler et d'utiliser pour raisonner. Sa définition se fonde essentiellement sur des observations et l'expérience acquise dans son environnement. Sa représentation stricte formulée par la fonction caractéristique ci-dessus n'est donc qu'une restriction du concept initial.

La formulation par sous-ensemble flou essaie de pallier les contraintes imposées par le précédent mode de représentation. Ainsi, un sous-ensemble flou E_i^f étend la notion de sous-ensemble en généralisant la fonction caractéristique par une *fonction d'appartenance* $\mu_{E_i^f} : E \rightarrow [0,1]$ qui permet de rendre compte de l'appartenance partielle (ou encore graduelle) d'un élément au sous-ensemble flou. Les fonctions d'appartenance peuvent être définies de différentes manières (cf. figure 2.2) mais leur principal objectif est de décrire le concept associé au sous-ensemble flou. Il existe donc une relation sémantique forte entre les deux qui rend leur extraction automatique souvent difficile (cf. paragraphe 3.2).

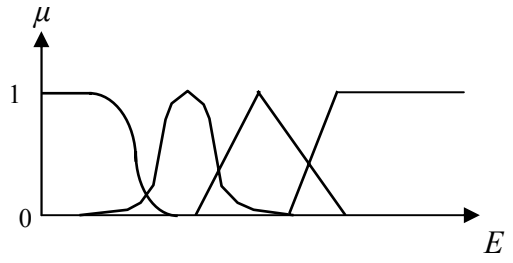

FIG. 2.2 – *Exemples de fonctions d'appartenance.*

En reprenant l'exemple sur les personnes de grande taille, la figure 2.3 illustre le graphe d'une fonction d'appartenance possible pour le sous-ensemble flou E^f_{grand}.

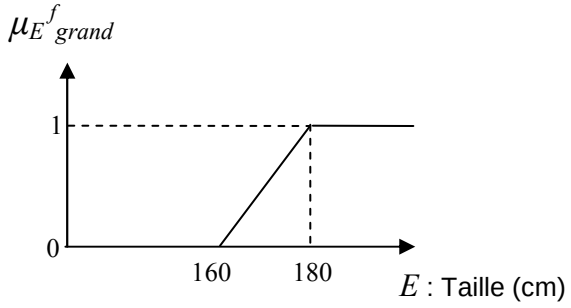

FIG. 2.3 – *Fonction d'appartenance à* E^f_{grand}

La forme du sous-ensemble flou ainsi défini traduit que les personnes de plus de 180cm sont considérées comme étant grandes et que celles dont la taille se trouve entre 160cm et 180cm le sont plus ou moins.

On remarquera à ce stade de la présentation qu'en dehors de la prise en compte des imprécisions, les sous-ensembles flous permettent de représenter des concepts de façon intuitive, ce qui facilite leur compréhension. De plus, de par leur granularité, les ensembles flous sont parfaitement adaptés pour représenter des données de façon synthétique et compacte, notamment lorsque l'univers de référence est numérique. Par exemple, si un échantillon de 10000 personnes est utilisé pour étudier la taille d'une population, celui-ci peut être synthétisé par trois (voir plus) sous-ensembles flous (petit, moyen et grand), évitant ainsi d'avoir à manipuler et/ou stocker l'ensemble des données. Ces deux points vont donc tout à fait dans le sens de nos objectifs.

2.1.1.2 Éléments caractéristiques des sous-ensembles flous

Bien que la fonction d'appartenance μ_A d'un sous-ensemble flou A permette de le définir entièrement, plusieurs éléments caractéristiques sont couramment employés pour les manipuler. Nous les énumérons ici de manière rapide en considérant E comme étant l'univers de référence.

- **Le noyau :**
  Le noyau $Noy(A)$ d'un sous-ensemble flou A est l'ensemble des éléments de E dont le degré d'appartenance à A vaut 1 :

$$Noy(A) = \{e \in E \,|\, \mu_A(e) = 1\} \tag{2.3}$$

– **Le support :**
Le support $Supp(A)$ d'un sous-ensemble flou A est l'ensemble des éléments de E dont le degré d'appartenance à A est non nul :

$$Supp(A) = \{e \in E \,|\, \mu_A(e) > 0\} \tag{2.4}$$

– **La hauteur :**
La hauteur $h(A)$ d'un sous-ensemble flou A est la valeur maximale de la fonction d'appartenance μ_A :

$$h(A) = \sup_{e \in E} \mu_A(e) \tag{2.5}$$

Si $h(A) = 1$, on dit que le sous-ensemble flou A est *normalisé*.

– **La cardinalité :**
La cardinalité $|A|$ d'un sous-ensemble flou A est la quantité (floue) d'éléments de E qui appartiennent à A :

$$|A| = \sum_{e \in E} \mu_A(e) \tag{2.6}$$

– **L'α-coupe :**
L'α-coupe A_α d'un sous-ensemble flou A est l'ensemble des éléments de E qui appartiennent à A avec un degré au moins égale à α :

$$A_\alpha = \{e \in E \,|\, \mu_A(e) \geq \alpha\} \tag{2.7}$$

L'α-coupe est un procédé qui permet à partir de sous-ensembles flous de revenir à la notion de sous-ensembles classiques. Ainsi, A_α est un sous-ensemble décrit par sa fonction caractéristique comme définie par l'équation 2.1.

La figure 2.4 illustre les principales notions définies ci-dessus.

2.1.1.3 Opérations sur les sous-ensembles flous

La théorie des sous-ensembles flous étant une généralisation de la théorie ensembliste, la plupart des opérations ensemblistes ordinaires sont aussi généralisées. Nous présentons ici les principales opérations utilisées pour la manipulation de deux sous-ensembles flous A et B :

– **L'égalité :**
Deux sous-ensembles flous A et B sont égaux si leurs fonctions d'appartenance sont égales en tout point de E :

$$A = B \iff \forall e \in E,\ \mu_A(e) = \mu_B(e) \tag{2.8}$$

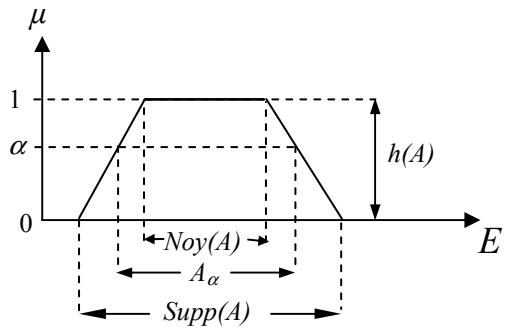

FIG. 2.4 – *Principaux éléments caractéristiques d'un sous-ensemble flou A.*

– **L'inclusion :**
Le sous-ensemble flou A est inclus dans le sous-ensemble flou B si tout élément de E appartient à A avec un degré d'appartenance inférieur ou égal à celui à B :

$$A \subseteq B \iff \forall e \in E,\ \mu_A(e) \leq \mu_B(e) \tag{2.9}$$

– **L'union :**
L'union de deux sous-ensembles flous A et B est un sous-ensemble flou C constitué des éléments de E associés à un degré d'appartenance à C qui correspond à la plus grande valeur[1] entre l'appartenance à A et l'appartenance à B :

$$\forall e \in E,\ \mu_C(e) = \mu_{A \cup B}(e) = \max(\mu_A(e), \mu_B(e)) \tag{2.10}$$

Un exemple de résultat de l'union entre deux sous-ensembles flous est présenté sur la figure 2.5.

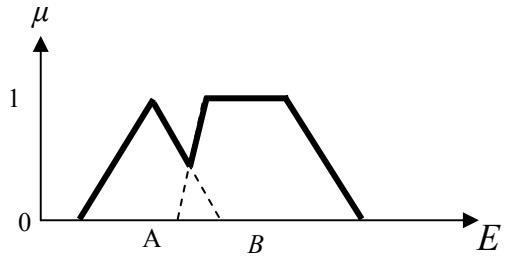

FIG. 2.5 – *Union de deux sous-ensemble flou A et B.*

– **L'intersection :**
L'intersection de deux sous-ensembles flous A et B est un sous-ensemble flou C

1. L'union peut être définie de manière plus générale en utilisant un type d'opérateur appelé conorme triangulaire (ou encore *t-conorme*) dont max fait partie (cf. paragraphe 2.1.1.4).

constitué des éléments de E associés à un degré d'appartenance à C qui correspond à la plus petite valeur[2] entre l'appartenance à A et l'appartenance à B :

$$\forall e \in E,\ \mu_C(e) = \mu_{A \cap B}(e) = \min(\mu_A(e), \mu_B(e)). \tag{2.11}$$

Un exemple de résultat de l'intersection entre deux sous-ensembles flous est présenté sur la figure 2.6.

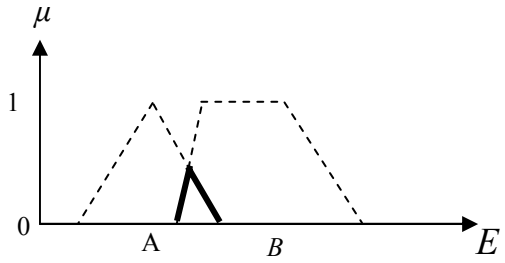

FIG. 2.6 – *Intersection de deux sous-ensemble flou A et B.*

– **Le complément :**
Le complément $\overline{A}$ du sous-ensemble flou A est un sous-ensemble flou constitué des éléments de E associés à un degré d'appartenance à $\overline{A}$ qui est d'autant plus grand que le degré d'appartenance à A est faible :

$$\forall e \in E,\ \mu_{\overline{A}}(e) = 1 - \mu_A(e). \tag{2.12}$$

– **Le produit cartésien :**
Les problèmes considérés sont souvent décrits dans plusieurs univers de référence $E_1, \ldots, E_n$ et il peut être intéressant de pouvoir raisonner dans un univers de référence E global composé de chacun des univers initiaux. E correspond donc au produit cartésien de $E_1, \ldots, E_n$: $E = E_1 \times \ldots \times E_n$ et ses éléments e sont des n-uplets : $e = (e_1, \ldots, e_n)$. Le produit cartésien de n sous-ensembles flous $A_1, \ldots, A_n$ définis respectivement sur les univers de référence $E_1, \ldots, E_n$ est un sous-ensemble flou A défini sur E par sa fonction d'appartenance :

$$\forall e = (e_1, \ldots, e_n) \in E,\ \mu_A(e) = min(\mu_{A_1}(e_1), \ldots, \mu_{A_n}(e_n)) \tag{2.13}$$

– **La projection :**
De façon analogue, si E est un univers de référence complexe résultant du produit cartésien d'autres univers de référence $E_1, \ldots, E_n$, il peut être intéressant d'obtenir

2. L'intersection peut être définie de manière plus générale en utilisant un type d'opérateur appelé norme triangulaire (ou encore *t-norme*) dont min fait partie (cf. paragraphe 2.1.1.4).

des informations sur un univers plus simple $F = F_a \times \ldots \times F_k$ avec $\{a, \ldots, k\} \subset \{1, \ldots, n\}$. La projection d'un sous-ensemble flou A, défini sur E, sur l'univers F est un sous-ensemble flou B défini sur F par :

$$\forall f = (f_a, \ldots, f_k) \in F,\ \mu_B(f) = \sup_{\{i | 1 \leq i \leq n,\ i \neq a, \ldots, i \neq k\}} \mu_A((\ldots, e_i, \ldots)) \qquad (2.14)$$

– **Relations floues et compositions :**
Une relation floue R permet de représenter un lien entre deux univers E_1 et E_2 par un sous-ensemble flou défini dans $E_1 \times E_2$ par sa fonction d'appartenance $\mu_R : E_1 \times E_2 \rightarrow [0, 1]$. La *composition* $B = A \circ R$ est alors l'opération qui permet, à partir de la relation R et d'un sous-ensemble flous A de E_1, de déduire le sous-ensemble flou B de E_2 défini par :

$$\forall e_2 \in E_2,\ \mu_B(e_2) = \sup_{e_1} \min(\mu_A(e_1), \mu_R(e_1, e_2)) \qquad (2.15)$$

L'extension à des relations floues n-aires et la composition de plusieurs relations floues sont bien entendu possibles.

2.1.1.4 Normes et conormes triangulaires

Dans la partie précédente, les opérations d'union et d'intersection ont été réalisées grâce aux fonctions min et max. Il ne s'agit en fait que de deux fonctions particulières qui appartiennent à une famille plus vaste : les *normes triangulaires* (*t-normes*) et les *conormes triangulaires* (*t-conormes*) respectivement. D'une façon générale, les t-normes et t-conormes permettent de généraliser les opérations ensemblistes d'union et d'intersection (et par extension la disjonction et la conjonction de propositions) sur les sous-ensembles flous. Pour que cette généralisation soit correcte, un certain nombre de propriétés doivent être vérifiées :

– **t-norme :**
La norme triangulaire est une fonction $\top : [0, 1] \times [0, 1] \rightarrow [0, 1]$ qui pour tout élément $x, y, z, t \in [0, 1]$ possède les propriétés suivantes :
  – commutativité : $\top(x, y) = \top(y, x)$,
  – associativité : $\top(x, \top(y, z)) = \top(\top(x, y), z)$
  – monotonie : $\top(x, y) \leq \top(z, t)\ si\ x \leq z\ et\ y \leq t$
  – 1 est élément neutre : $\top(x, 1) = x$

  De plus, elle assure que $\forall x, y \in [0, 1]$, $\top(x, y) \leq x$ et $\top(x, y) \leq y$. Le tableau 2.1 regroupe les t-normes les plus courantes.
– **t-conorme :**
La conorme triangulaire est une fonction $\perp : [0, 1] \times [0, 1] \rightarrow [0, 1]$ qui pour tout élément $x, y, z, t \in [0, 1]$ possède les propriétés suivantes :
  – commutativité : $\perp(x, y) = \perp(y, x)$,

Nom	Fonction
Zadeh	$\min(x, y)$
probabiliste	$x.y$
Lukasiewicz	$\max(x + y - 1, 0)$

TAB. 2.1 – *Exemples de normes triangulaires.*

- associativité : $\perp(x, \perp(y, z)) = \perp(\perp(x, y), z)$
- monotonie : $\perp(x, y) \leq \perp(z, t)$ *si* $x \leq z$ *et* $y \leq t$
- 0 est élément neutre : $\perp(x, 0) = x$

Elle assure aussi que $\forall x, y \in [0, 1]$, $\perp(x, y) \geq x$ et que $\perp(x, y) \geq y$. Le tableau 2.2 regroupe les t-conormes les plus courantes.

Nom	Fonction
Zadeh	$\max(x, y)$
probabiliste	$x + y - x.y$
Lukasiewicz	$\min(x + y, 1)$

TAB. 2.2 – *Exemples de conormes triangulaires.*

L'existence de fonctions différentes pour réaliser les normes et conormes triangulaires n'est pas anodin. Elles ont été définies pour obtenir des comportements particuliers en fonction du type d'opération à réaliser et du contexte d'utilisation ([25] donne un exemple dans le cadre de la fusion de données). Le choix de ces fonctions dans un système revêt donc une importance toute particulière.

2.1.2 Raisonnement à partir des sous-ensembles flous

Lors de leur définition, les sous-ensembles flous peuvent être associés à un concept à décrire. Il existe alors un lien sémantique fort entre le sous-ensemble flou décrit par sa fonction d'appartenance et l'univers de référence sur lequel il est défini. Dans ce contexte, des mécanismes de raisonnement peuvent être mis en œuvre pour étendre les principes de la logique classique à ceux de la logique floue.

2.1.2.1 Variable linguistique

En logique, les concepts manipulés sont décrits par des attributs qui prennent leurs valeurs sur un univers de référence. En logique floue, ces concepts sont représentés par des *variables linguistiques*. Une variable linguistique est définie par un triplet (A, E_A, F_A) où A est un attribut (nom de variable), E_A son univers de référence et $F_A = \{A_1, A_2, \ldots\}$ est un ensemble de sous-ensembles flous décrivant A. Les sous-ensembles flous $A_1, A_2, \ldots$ sont souvent désignés par un terme linguistique (ou valeur) précisant leur rôle. La figure 2.7 montre un exemple de variable linguistique associée au concept de taille d'une personne : $(taille, [0, 300], \{petit, moyen, grand\})$.

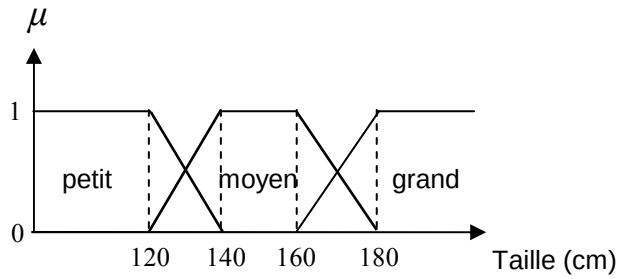

FIG. 2.7 – *Variable linguistique associée à la taille d'une personne*

Une variable linguistique permet donc d'une part de synthétiser l'information manipulée grâce aux sous-ensembles flous (modélisation qualitative) et d'autre part de représenter des concepts imprécis tels que l'homme en manipule quotidiennement. La détermination de la forme et de la position de ces sous-ensembles flous est un élément essentiel pour pouvoir effectuer des raisonnements valides, robustes et compréhensibles. C'est pour cette raison que dans beaucoup de systèmes comme ceux destinés à l'aide à la décision, les sous-ensembles flous sont définis *a priori* par des experts du domaine afin qu'ils représentent exactement leurs connaissances. Cependant, il n'est pas toujours possible d'obtenir une telle expertise, que ce soit à cause de la complexité du problème ou bien parce que les experts sont trop rares voir inexistants. Dans ces conditions, des algorithmes peuvent être mis en œuvre pour extraire automatiquement les sous-ensembles flous (cf. chapitre 3). Une expertise du résultat peut éventuellement être faite par la suite afin de déterminer la signification (le terme linguistique) des sous-ensembles flous obtenus. En général, cette expertise ne sera réellement possible que si les algorithmes utilisés limitent la quantité de sous-ensembles flous produits ainsi que leur chevauchements [66].

2.1.2.2 Propositions floues

La logique classique manipule des propositions qui sont soit vraies, soit fausses. En logique floue, les propositions sont généralisées sous la forme de propositions floues. Si (A, E_A, F_A) est une variable linguistique, « A est A_1 » est un exemple de proposition floue. Celle-ci n'est plus caractérisée par une valeur de vérité dans $\{vrai, faux\}$ mais par un degré de vérité dans $[0, 1]$ qui est déduit de la fonction d'appartenance au sous-ensemble flou mis en jeu. En reprenant l'exemple précédent, le degré de vérité de « A est A_1 » sera déterminée par μ_{A_1} pour chaque élément de E_A.

Ces propositions floues élémentaires peuvent ensuite être utilisées pour former des propositions floues plus complexes, en les combinant entre elles par différents opérateurs. Ainsi, la négation d'une proposition floue « A est A_1 » sera désignée par « A n'est pas A_1 ». Son degré de vérité peut être obtenu à partir de la fonction d'appartenance au complément du sous-ensemble flou A_1 soit $\mu_{\overline{A_1}}$ (mais d'autres types de négation peuvent aussi être employés).

Deux propositions p_A et p_B peuvent aussi être combinées afin de construire une nouvelle proposition floue dont la valeur de vérité dépendra de celles de p_A et p_B. Il existe trois formes courantes de combinaison : la conjonction ($\wedge$), la disjonction ($\vee$) et l'implication ($\Longrightarrow$). En logique classique, la conjonction $p_{A \wedge B}$ est vraie si et seulement si p_A et p_B sont toutes les deux vraies. En logique floue, l'imprécision relative aux propositions se traduira par $p_{A \wedge B}$ est d'autant plus vraie que p_A et p_B le sont ensemble. Le degré de vérité associé à $p_{A \wedge B}$ est généralement obtenu en utilisant une t-norme : $\mu_{A \wedge B} = \min(\mu_A, \mu_B)$ par exemple.

De façon similaire, la disjonction ordinaire veut que $p_{A \vee B}$ soit vraie si p_A est vraie ou bien si p_B est vraie. En logique floue, $p_{A \vee B}$ sera d'autant plus vraie que p_A ou p_B le sera. Le degré de vérité s'obtient généralement par une t-conorme : $\mu_{A \vee B} = \max(\mu_A, \mu_B)$ par exemple.

L'implication joue un rôle particulier puisqu'elle est à la base de la déduction logique. Il existe là aussi plusieurs fonctions permettant de mettre en relation deux propositions floues, suivant que l'on souhaite conserver les propriétés de la logique classique ou non. Ce cas particulier est décrit en détails dans le paragraphe suivant.

2.1.3 Règles floues et systèmes d'inférence floue

Grâce aux éléments de base définis ci-dessus, des raisonnements à base de règles floues peuvent être effectués. Une règle de décision ordinaire se présente sous la forme suivante : « Si A est A_i Alors B est B_j ». Une règle floue étend ce principe afin de pouvoir gérer les imprécisions. Des bases de règles peuvent ainsi être élaborées pour modéliser différents types de problèmes, qu'il s'agisse de classification, d'aide au diagnostic, de commande, etc. Ces bases de règles floues sont appelées systèmes d'inférence floue. Ils mettent en œuvre un mécanisme de déduction[3] pour obtenir de nouvelles connaissances à partir de celles représentées dans les règles.

2.1.3.1 Les règles floues

Une règle floue R : « Si A est A_i Alors B est B_j » est une relation entre deux propositions floues ayant chacune un rôle particulier. La première (A est A_i) est appelée *prémisse* de la règle alors que la seconde (B est B_j) est la *conclusion*. Dans le cas de propositions floues élémentaires, la prémisse et la conclusion sont définies à partir de deux variables linguistiques (A, E_A, F_A) et (B, E_B, F_B) qui décrivent les connaissances relatives aux univers de référence E_A et E_B de manière à prendre en compte l'imprécision relative aux modalités de A et B. Grâce à ce mode de représentation, des relations imprécises comme « S'il fait beau temps alors la visibilité en mer est bonne » peuvent être exprimées alors que ce n'est pas le cas en logique classique.

Une proposition floue élémentaire est souvent insuffisante pour représenter l'ensemble des informations à manipuler. Plusieurs propositions floues peuvent alors être combinées pour enrichir et détailler la représentation. Ainsi, la prémisse correspondant à « le temps est beau » peut correspondre à la conjonction de deux autres propositions :

3. Il existe aussi d'autres formes de raisonnement [27].

« la mer est calme **et** le taux d'humidité est faible ». Les opérateurs de conjonction, disjonction et de négation énoncés dans la partie précédente sont souvent utilisés à cet effet.

La relation R entre la prémisse et la conclusion de la règle est déterminée par une implication floue dont le degré de vérité est défini par une fonction d'appartenance μ_R qui dépend du degré de vérité de chacune des deux propositions. Si μ_{A_i} et μ_{B_j} désignent les fonctions d'appartenance aux sous-ensembles flous A_i et B_j caractérisant la prémisse et la conclusion de la règle R, la fonction d'appartenance décrivant la proposition floue R est de la forme :

$$\mu_R(e, y) = I(\mu_{A_i}(e), \mu_{B_j}(y)),$$

où (e, y) appartient à $E_A \times E_B$ et $I : [0, 1] \to [0, 1]$ est une fonction correspondant à l'implication floue. Il existe un certain nombre de fonctions permettant d'implémenter l'implication floue et de prendre en compte l'aspect graduel des propositions floues qu'elle relie. Le tableau 2.3 en précise quelques unes.

Nom	Degré de vérité
Reichenbach	$1 - \mu_A(e) + \mu_A(e)\mu_B(y)$
Rescher-Gaines	$\begin{cases} 1 & si\ \mu_A(e) \leq \mu_B(y) \\ 0 & sinon \end{cases}$
Kleene-Dienes	$\max(1 - \mu_A(e), \mu_B(y))$
Lukasiewicz	$\min(1 - \mu_A(e) + \mu_B(y), 1)$

TAB. 2.3 – *Exemples d'implications floues* $I(\mu_A(e), \mu_B(y))$.

Ces implications généralisent l'interprétation des règles implicatives de la logique classique et ont des comportements plus ou moins analogues dans les applications. Elles possèdent cependant des propriétés différentes qui peuvent être plus ou moins adaptées selon le contexte d'utilisation (cf. [27] pour plus de détails).

Il existe aussi une autre façon de représenter la relation R entre la prémisse et la conclusion d'une règle. Cette méthode considère que le lien entre les deux propositions est de nature conjonctive. On parle alors de *relation floue conjonctive* et par extension, l'opérateur I est souvent appelé implication « conjonctive » [85] bien que celle-ci ne corresponde pas au comportement de l'implication en logique classique. Elle repose sur l'utilisation d'un opérateur type t-norme (cf. tableau 2.4).

Nom	Degré de vérité
Mamdani	$\min(\mu_A(e), \mu_B(y))$
Larsen	$\mu_A(e) \cdot \mu_B(y)$

TAB. 2.4 – *Exemples d'implications conjonctives* $I(\mu_A(e), \mu_B(y))$.

2.1.3.2 Raisonnement par déduction

Le *modus ponens* est le mode de raisonnement principal utilisé en logique classique. Dans le cas des règles floues, ce mode de raisonnement est étendu par le *modus ponens généralisé*. Étant donné une règle floue R : « Si A est A_i Alors B est B_j » et une proposition floue « A est A'_i » qui ne correspond que partiellement à la prémisse de R, le *modus ponens* généralisé permet d'en déduire un sous-ensemble flou B'_j décrivant B. B'_j est défini par sa fonction d'appartenance :

$$\forall y \in E_B,\ \mu_{B'_j}(y) = \sup_{e \in E_A} Ponens(\mu_{A'_i}(e), \mu_R(e, y)), \tag{2.16}$$

où $Ponens : [0,1] \times [0,1] \rightarrow [0,1]$ est l'opérateur du *modus ponens généralisé*, le plus souvent choisi parmi les t-normes.[4]

Le comportement du *modus ponens* généralisé doit permettre d'obtenir en résultat un sous-ensemble flou B'_j d'autant plus proche de B_j que A'_i est proche A_i. Pour que la généralisation soit effective, il doit permettre de retrouver B_j si $A'_i = A_i$. Le choix de la fonction $Ponens$ est donc important et dépend généralement de l'implication floue I choisie [27].

Il existe un cas particulier de l'utilisation des règles qu'il est intéressant de souligner. Celui-ci intervient lorsque les observations présentées en entrée du système sont numériques. Dans cette situation, lors de la mise en œuvre du raisonnement déductif, l'observation mise en relation avec les règles n'est pas un sous-ensemble flou A'_i mais une valeur précise e_0 de l'univers de référence E_A. Le *modus ponens* généralisé (cf. équation (2.16)) se simplifie alors de la manière suivante :

$$\forall y \in E_B,\ \mu_{B'_j}(y) = \mu_R(e_0, y) = I(\mu_{A_i}(e_0), \mu_{B_j}(y)) \tag{2.17}$$

2.1.3.3 Systèmes d'inférence floue

En général, une simple règle de décision ne permet pas à elle seule de condenser toute l'information nécessaire à la représentation d'un problème complexe. Des bases de règles doivent alors être élaborées pour aboutir à une description suffisamment complète qui puisse être utilisable. Ces bases de règles floues, associées aux mécanismes de décision, permettent de déduire de nouvelles connaissances. Elles constituent des *systèmes d'inférence floue* (SIF). Les paragraphes précédents ont décrit les processus permettant de mettre en relation une règle floue et une observation (qu'elle soit précise ou floue) pour déterminer une conclusion. Nous présentons ici le fonctionnement lorsque plusieurs règles sont utilisées de façon conjointe.

Lorsque plusieurs règles d'un SIF sont activées en même temps, soit les conclusions portent sur des variables linguistiques différentes, soit un certain nombre d'entre elles partagent les même variables linguistiques. Dans le premier cas, la mise en œuvre du

4. Dans le cadre de relations floues, le *modus ponens généralisé* correspond à la *règle compositionnelle d'inférence* [182] qui se déduit du principe de composition (cf. équation (2.15)).

SIF aboutit à des conclusions multiples, une pour chaque variable linguistique. Dans le deuxième cas, il faut agréger les résultats des différentes règles, de manière à n'obtenir qu'une seule conclusion par variable linguistique. Il existe alors deux manières de procéder. Considérons alors une observation « A est A' » et une base composée de r règles $R_i,\ i = 1, \ldots, r$ de la forme suivante : « Si A est A_i Alors B est B_i ».

La première solution, appelée *inférence locale*, consiste à établir le résultat de chaque règle indépendamment les unes des autres. En reprenant l'équation 2.16, cela nous donne les sous-ensembles flous définis par : $\mu_{B'_1}(y), \ldots, \mu_{B'_r}(y)$. Ensuite, une intersection de ces résultats intermédiaires est faite à l'aide d'une t-norme. La conclusion finale est donc le sous-ensemble flou B' de fonction d'appartenance :

$$\mu_{B'}(y) = \top_{i=1,\ldots,r}(\mu_{B'_i}(y)) \tag{2.18}$$

Si les règles utilisent une implication conjonctive (implication de Larsen ou de Mamdani par exemple), il faut combiner les conclusions de chaque règle non pas par une intersection mais par une union avec une t-conorme [27, 85] :

$$\mu_{B'}(y) = \perp_{i=1,\ldots,r}(\mu_{B'_i}(y)) \tag{2.19}$$

La deuxième solution est la méthode dite d'*inférence globale* ou encore d'*agrégation*. Elle consiste à élaborer une nouvelle règle R qui se substitue aux précédentes. Celle-ci est définie par la fonction d'appartenance décrivant la relation floue : $\mu_R(e, y) = \top_{i=1,\ldots,r}(\mu_{R_i}(e, y))$. Là encore, si les règles utilisent une implication conjonctive, il faut utiliser une t-conorme au lieu de la t-norme. La conclusion finale est obtenue en utilisant le *modus ponens généralisé* sur cette nouvelle règle :

$$\mu_{B'}(y) = \sup_{e \in E_A} Ponens(\mu_{A'}(e), \mu_R(e, y)).$$

Il existe plusieurs différences entre ces deux types d'inférence. La première est d'ordre calculatoire : l'inférence globale est plus coûteuse que l'inférence locale qui lui est donc souvent préférée. Une seconde différence réside dans leur comportement en fonction de l'implication utilisée. Ainsi, les règles basées sur une implication conjonctive ont l'intéressante particularité de produire le même résultat pour les deux types d'inférence. En revanche, ce n'est pas toujours le cas lorsque les règles utilisent une implication floue qui généralise l'implication de la logique ordinaire. Dans ce cas, les résultats d'une inférence locale sont en général moins restrictifs (moins informants) que ceux d'une inférence globale. Elles peuvent alors aboutir à des conclusions de type « indéfini », ce qui n'est pas toujours souhaitable en pratique [85].

2.1.4 Cas particuliers en reconnaissance de formes et en soft computing

Suivant les contextes applicatifs et la façon dont le problème est envisagé, il existe différentes façons de concevoir un SIF et de l'exploiter. Les problèmes de reconnaissance de formes qui nous intéressent ici possèdent certaines particularités qui doivent

être prises en compte lors de l'élaboration de systèmes d'inférence. On considère notamment que les entrées sont des formes décrites de façon précise (non floue) dans leur espace de représentation et que l'univers de sortie S des classes est une variable catégorielle non ordonnée (chiffres arabes ou lettres de l'alphabet latin par exemple). Il existe alors différentes possibilités pour représenter les règles de classification [27, 28]. Nous n'en présentons que quelques unes ici.

Une première façon de procéder consiste à établir une relation floue entre l'espace des entrées et l'espace des sorties. Pour cela, l'espace des entrées est découpé en sous-domaines qui sont associés aux classes. En reprenant le formalisme de règle, cela se traduit de la façon suivante :

R_i : « Si e_1 est A_{1i} et ... et e_n est A_{ni} Alors e appartient à ω_1 avec un degré b_{1i} et... et e appartient à $\omega_{\mathcal{S}}$ avec un degré $b_{\mathcal{S}i}$. »

Dans cette règle, la prémisse correspond à la description d'un sous-domaine de l'espace des entrées qui est associé à chacune des classes ω_s. Les valeurs b_{si} sont des cœfficients mesurant la force de la relation entre le sous-domaine et les classes.[5] Pour une donnée e, son appartenance à chacune des classes est obtenue en utilisant une composition de type sup-min :

$$\mu_{\omega_j}(e) = \sup_i \min(\beta_i(e), b_{ji})$$

avec $\beta_i(e)$ représentant l'adéquation de e au sous-domaine décrit dans la prémisse de la règle i, soit par exemple : $\beta_i(e) = \min(\mu_{A_{1i}}(e), \ldots, \mu_{A_{ni}}(e))$.

Une autre possibilité, consiste à représenter les règles en les formalisant de façon générale par :

R_i : « Si e_1 est A_{1i} et ... et e_n est A_{ni} Alors ω_1 est b_{1i} et ... et $\omega_{\mathcal{S}}$ est $b_{\mathcal{S}i}$ »

La signification des conclusions peut être envisagée ici de différentes façons. Il peut par exemple s'agir de sous-ensembles flous réduits à des singletons, de cœfficients mesurant le lien entre la prémisse et la conclusion (ce qui est similaire à la représentation par relations floues vue ci-dessus) ou encore de poids. Ces règles peuvent notamment être exploitées dans des systèmes de type Mamdani ou Takagi-Sugeno d'ordre 0 que nous présentons ci-dessous.

Les systèmes de type Mamdani reposent sur des règles semblables à celles qui ont été présentées tout au long du chapitre. Elles utilisent en prémisse et en conclusion des variables linguistiques, dont les sous-ensembles flous peuvent éventuellement être réduits à des singletons. Soit $E = E_1 \times \ldots \times E_n$ l'univers de référence et en considérant

5. Ce type de règles à conclusion multiple peut aussi être divisé en $\mathcal{S}$ règles pour simplifier l'interprétation.

que $e_0 = (e_1, \ldots, e_n)$ est un vecteur numérique en entrée du système, une règle R_i est formalisée de la manière suivante :

$$R_i : \text{« Si } e_1 \text{ est } A_{1i} \text{ et } \ldots \text{et } e_n \text{ est } A_{ni} \text{ Alors } B \text{ est } B_i \text{ »}$$

Les opérateurs de conjonction et d'implication sont classiquement représentés par le minimum, et l'agrégation des règles par le maximum. De part les simplifications du *modus ponens généralisé* (équation 2.17) résultant de l'utilisation d'entrées numériques, la méthode d'inférence est souvent appelée par le nom de ces deux opérateurs. Les principales variantes sont résumées ci-dessous :

- l'inférence *max-min* :

$$\mu_{B'}(y) = \max_{i=1,\ldots,r} \min(\beta_i, \mu_{B_i}(y)), \tag{2.20}$$

  où $\beta_i = \min_{j=1,\ldots,n}(\mu_{A_{ji}}(e_j))$ représente l'activation de la prémisse de la règle R_i ;
- l'inférence *max-produit* :

$$\mu_{B'}(y) = \max_{i=1,\ldots,r} \beta_i \cdot \mu_{B_i}(y), \tag{2.21}$$

  où $\beta_i = \min_{j=1,\ldots,n}(\mu_{A_{ji}}(e_j))$ ou encore, $\beta_i = \Pi_{j=1}^{n} \mu_{A_{ji}}(e_j)$;

Lorsque la sortie du système doit être numérique, pour qu'elle puisse être traitée par d'autres modules, il faut faire une « défuzzification » du sous-ensemble flou résultant de l'inférence. Il existe plusieurs méthodes [85] dont la plus connue est la méthode du centre de gravité. Dans le cas discret elle s'exprime par :

$$b = \frac{\sum_{k=1}^{K} y_k \cdot \mu_{B'}(y_k)}{\sum_{k=1}^{K} \mu_{B'}(y_k)},$$

où K représente le nombre de partitions utilisées pour discrétiser l'espace de sortie.

Les systèmes type Takagi-Sugeno utilisent quant à eux des règles dont les conclusions sont des fonctions des entrées. Une règle R_i est formalisée par :

$$R_i : \text{« Si } e_1 \text{ est } A_{1i} \text{ et } \ldots \text{et } e_n \text{ est } A_{ni} \text{ Alors } b_i = f_i(e_0) \text{ »},$$

où b_i est une valeur numérique et f_i une fonction (constante, polynômiale, non-linéaire, etc.). De part cette forme particulière des conclusions, le calcul de la sortie du système s'obtient par une opération mêlant « inférence » et « défuzzification », souvent appelée inférence de type *somme-produit* :

$$b = \frac{\sum_{i=1}^{r} \beta_i \cdot b_i}{\sum_{i=1}^{r} \beta_i}, \tag{2.22}$$

où $\beta_i = \Pi_{j=1}^{n} \mu_{A_{ji}}(e_j)$ représente l'activation de la prémisse de la règle R_i et r est le nombre de règles. On notera que lorsque les conclusions des règles sont des constantes (fonctions d'ordre 0), elles deviennent équivalentes aux règles utilisées dans les systèmes type Mamdani avec des sous-ensembles flous B_i réduits à des singletons b_i.

Les systèmes de type Mamdani et Takagi-Sugeno sont très utilisés dans de nombreux domaines et notamment en commande floue [85, 28] parce qu'ils possèdent des propriétés très particulières. Ils reposent sur un mécanisme d'inférence simple qui permet d'obtenir un résultat analytique sans surcharge calculatoire puisque, comme souligné dans le paragraphe 2.1.3.3, dans le cas d'entrées numériques, l'inférence d'une règle se simplifie de façon considérable (cf. équation (2.17)). De plus, ils peuvent être aisément améliorés par des méthodes d'optimisation comme les moindre carrés ou une descente de gradient, même s'il existe des risques de perdre au niveau de l'interprétabilité des règles si aucune contrainte n'est imposée sur l'optimisation. Il faut donc en général faire un compromis entre les performances et l'interprétabilité. Pour de plus amples détails, le lecteur peut se référer à [66].

Ces systèmes offrent donc une certaine souplesse sur la forme de la sortie et permettent assez simplement d'obtenir de bonnes performances. Cela rend leur utilisation en reconnaissance de forme et en soft computing particulièrement intéressante.

2.2 Systèmes à multiples classifieurs : vers une approche de combinaison structurée guidée par les données

Dans la partie précédente, nous avons présenté les principes permettant de modéliser des connaissances imprécises de façon robuste, synthétique et interprétable ainsi que la façon dont elles pouvaient être intégrées pour élaborer des règles de décision pour la reconnaissance de formes. Il nous reste à présent à voir comment traiter les problèmes complexes tout en conservant une modélisation la plus interprétable et compacte possible.

La plupart des systèmes de reconnaissance courants reposent sur une modélisation unique et globale du problème considéré. Même si de telles approches peuvent suffire lorsque la complexité de la tâche de reconnaissance est modérée, elles montrent souvent leurs limites sur des problèmes complexes. Ces limitations sont généralement de deux ordres : soit les performances sont insuffisantes, soit la complexité de la modélisation devient trop importante, rendant le système difficile à maintenir et à exploiter notamment sur des machines aux ressources limitées.

Pour pouvoir aborder ces problèmes complexes, des systèmes à classifieurs multiples ont commencé à être développés dans les années 1990. L'idée principale derrière ces approches est d'essayer de bénéficier au maximum de la complémentarité de différentes modélisations. Pour cela, plusieurs classifieurs utilisant des architectures et/ou des données différentes sont combinés de manière à obtenir des décisions plus robustes et à compenser les faiblesses propres à chacun d'eux.

Dans cette partie, les principes généraux de la combinaison de classifieurs sont abordés en s'appuyant sur le point de vue du concepteur. En effet, pour la réalisation de notre système, le choix des différents constituants ainsi que leur organisation sont primordiaux. Ils ont été guidés avant tout par les fonctionnalités et propriétés désirées.

C'est pourquoi nous introduisons une catégorisation des approches selon la nature du lien unissant les « composants » du système combiné et leur rôle par rapport aux objectifs que le concepteur peut avoir. Nous montrons alors que dans le cadre de notre problématique, il est important que chaque élément, ainsi que le lien les unissant, repose de façon explicite sur les propriétés des données dans leur espace de représentation afin de maintenir la plus grande lisibilité possible et accroître la robustesse. C'est pourquoi ce type d'approche est qualifié de *combinaison structurée guidée par les données.*

2.2.1 Accroître les performances en combinant plusieurs classifieurs

L'idée d'utiliser plusieurs classifieurs pour obtenir un système de reconnaissance plus performant provient en grande partie de l'observation du fonctionnement humain, notamment au niveau de ses méthodes d'analyse et de son comportement décisionnel au sein d'un groupe.

Lorsque l'homme analyse une situation, il utilise plus ou moins consciemment plusieurs de ses sens afin d'améliorer sa compréhension du problème. Ainsi, lorsque nous parlons avec quelqu'un, nous analysons bien entendu les mots que nous entendons mais notre cerveau « augmente » aussi cette information en exploitant des informations visuelles telles que les gestes de l'interlocuteur ou encore les traits que prend tour à tour son visage. Même si cette information supplémentaire ne permet pas à elle seule de trouver le sens des mots, elle ajoute des connaissances supplémentaires sur la sémantique du message émis par l'interlocuteur. Il est d'ailleurs reconnu depuis longtemps que sous certaines conditions, la multimodalité, c'est-à-dire la prise en compte de plusieurs modes de communication, facilite la compréhension. Il suffit pour s'en convaincre de prendre l'exemple simple des exposés oraux qui sont désormais presque systématiquement appuyés d'une présentation visuelle. Ce gain lié à la multimodalité peut aussi être généralisé à l'exploitation de sources multiples d'informations. Prenons le cas de l'apprentissage scolaire. Quand une notion est difficile à acquérir, un étudiant utilisera bien souvent un livre pour reprendre les points mal compris pendant le cours. Si ce livre ne lui permet toujours pas de bien assimiler la notion, il pourra en rechercher un second, voire un troisième, présentant les choses différemment. Il pourra aussi se faire réexpliquer le problème par un ou plusieurs de ses camarades. Ces différentes sources d'informations ne font pas qu'introduire une redondance de la notion à acquérir. Elles permettent au contraire de fournir des points de vue différents que l'homme est capable de fusionner pour faciliter sa compréhension du sujet. Les décisions ultérieures qu'il sera amené à prendre et qui dépendront de ces informations acquises n'en seront que plus efficaces.

En considérant à présent l'homme dans une structure de groupe élaborée dans le but de résoudre une tâche particulière, les mécanismes d'analyse et de décision associés deviennent plus riches et plus variés. Ce schéma de groupe est essentiellement caractérisé par ses membres, la fonction qu'ils y remplissent et le mécanisme de décision associé. Les fonctionnalités des membres sont très importantes car ce sont elles qui déterminent les propriétés de l'ensemble. Les capacités de traitement du groupe ainsi que

son organisation en dépendent donc directement.

Dans le cas le plus simple, chaque membre agit comme un acteur indépendant dans le mécanisme de décision. Chacun possède sa propre vision globale du problème en fonction de ses connaissances personnelles, de son intuition et des sources d'informations plus ou moins complètes qu'il possède. Chacun donne son opinion directement sur la décision à prendre, sans être influencé par les autres membres. Des avis différents vont émerger, chacun ayant sa propre justification à sa vision du problème. La décision finale se fait alors en prenant en compte les différents avis par un vote par exemple et en accordant éventuellement plus de poids à tel ou tel membre en fonction de sa position au sein du groupe, que celle-ci soit liée à son expérience, son efficacité, sa renommée ou totalement arbitraire.

Un autre cas de figure se présente lorsque les membres du groupe sont non plus indépendants mais au contraire fonctionnellement dépendants les uns des autres par rapport à la tâche principale à effectuer. Chaque individu est ici spécialisé, remplit une fonction précise dont le résultat peut dépendre de celui d'un ou plusieurs autres membres. Le groupe est donc organisé autour des fonctionnalités de ses membres.

Ces schémas de décision peuvent être appliqués aux problèmes de classification par le biais d'approches à multiples classifieurs.[6] Comme les structures de groupes à l'échelle humaine, ces ensembles de classifieurs sont caractérisés par leurs membres (les systèmes de reconnaissance qui les composent), leurs fonctions au sein du système et le mécanisme de décision associé. Dans la littérature, plusieurs catégorisations de ces approches de classification ont été faites, en se basant sur : la topologie, c'est-à-dire l'architecture du système [106] ; l'espace de représentation des entrées, qu'il soit multiple ou pas [95] ; la stratégie de résolution, par optimisation de la combinaison ou par optimisation de la couverture des classifieurs (*decision optimization*/*coverage optimization* [75]) qui se rapprochent des méthodes *génératives* et *non-génératives* [165]). Ces catégorisations ont un inconvénient : elles reflètent plus les moyens utilisés pour faire la combinaison que la nature même de celle-ci, c'est-à-dire son rôle, qui se traduit par la nature du lien entre les classifieurs. Or c'est bien celle-ci, ainsi que les fonctionnalités souhaitées, qui guident l'élaboration du système et de son architecture.

Aussi nous proposons une nouvelle catégorisation des approches de combinaison de classifieurs qui est orthogonale aux précédentes. Elle ne vise pas à les remplacer mais plutôt à introduire le point de vue du concepteur, en s'appuyant sur l'aspect fonctionnel du système. Pour cela, nous distinguons d'une part les *approches de combinaison fonctionnellement indépendantes* et d'autre part les *approches de combinaison fonctionnellement dépendantes* par rapport à la tâche de classification. Ces dernières sont elles-mêmes subdivisées en qualifiant la nature de la relation entre les classifieurs par rapport aux objectifs. Ce type de catégorisation n'est ni strict, ni absolu mais encore une fois, son objectif est de refléter l'intention du concepteur.

6. Même si la multimodalité peut être traitée d'une façon différente, notamment par le biais de la fusion de données.

2.2.2 Approches de combinaison fonctionnellement indépendantes

Ces approches de combinaison ont comme objectif de résoudre le problème de classification en utilisant un mécanisme de décision reposant sur l'utilisation de points de vue multiples. Chaque classifieur du système possède sa propre modélisation/vision du problème et prend ses décisions indépendamment des autres. Soit ces classifieurs existent déjà, auquel cas il ne reste plus qu'à trouver une combinaison optimale, soit il est nécessaire de générer ces classifieurs de façon à ce qu'ils reposent sur différents points de vue. Les approches fonctionnellement indépendantes peuvent donc être divisées en deux, en reprenant la distinction faite dans [165] entre *approches génératives* et *approches non-génératives.*[7]

Qu'elles soient génératives ou non, ces approches possèdent de nombreux points communs. Elles sont donc présentées en même temps, en précisant uniquement les notions qui les différencient (cf. paragraphe 2.2.2.2).

2.2.2.1 Indépendance des classifieurs

L'intérêt des approches fonctionnellement indépendantes repose sur la possibilité d'utiliser des points de vue différents du problème en espérant que ceux-ci puissent se compléter pour la tâche de classification. Ce mode de fonctionnement a été directement déduit de l'analogie avec les mécanismes d'analyse et de décision de l'homme, seul ou en groupe, comme ils ont été présentés en introduction.

Ces raisons intuitives qui poussent à combiner des classifieurs effectuant la même tâche mais différemment sont aussi appuyées par des raisons plus conceptuelles liées aux mécanismes de classification en eux-mêmes. Ainsi, comme le montre Dietterich [44], il existe au moins trois raisons pour lesquelles l'utilisation d'un ensemble de classifieurs peut améliorer les performances en reconnaissance.[8] La première raison est d'ordre statistique : utiliser plusieurs reconnaisseurs différents permet de réduire les risques d'erreurs liés à l'utilisation d'un seul classifieur. La deuxième raison est liée aux méthodes d'apprentissage des algorithmes de classification qui atteignent souvent des solutions optimales mais localement seulement. Utiliser plusieurs reconnaisseurs permet alors d'obtenir une meilleure approximation de la solution en limitant les effets de ces optima locaux. Enfin, la troisième raison est liée au pouvoir d'expression des classifieurs. Même si certains sont reconnus comme étant des approximateurs universels (PMC [78], RBFN [134], SIF [87]), la quantité et la qualité des données nécessaires pour obtenir une telle approximation ainsi que la complexité du classifieur résultant limitent les performances en pratique. Utiliser plusieurs classifieurs permet donc d'étendre le pouvoir de représentation d'un seul reconnaisseur.

Quelle que soit l'origine de cette diversité dans la représentation du problème, celle-ci se traduit directement sur les sorties des classifieurs qui doivent être différentes. C'est pourquoi, il est souvent question d'*indépendance* des classifieurs (du point de

7. Cette distinction est similaire à celle faite dans [75].
8. Celles-ci sont reprises et étendues dans [165].

vue de la décision) comme étant un des critères nécessaires pour pouvoir faire une bonne combinaison [164, 51, 5]. Mais la terminologie est en fait assez vague puisque d'autres termes comme la *non-corrélation* [19, 18, 109] ou encore la *différence* [44] sont couramment rencontrés. Il est donc important ici de s'attarder sur ce qui est réellement sous-entendu par ces termes.

La notion d'indépendance vient du cadre de décision Bayésien où les classifieurs sont assimilés à des variables aléatoires. Ainsi, si s^1 et s^2 sont les sorties de deux classifieurs, ils sont considérés comme indépendants si la condition suivante est assurée :

$$P(s^1 = \omega_i, s^2 = \omega_j) = P(s^1 = \omega_i)P(s^2 = \omega_j),\ \forall \omega_i, \omega_j \in S \tag{2.23}$$

où S est l'ensemble des classes et P représente une probabilité. Comme le montre Moobed dans sa thèse [127], l'indépendance n'est en général pas équivalente à la non-corrélation de deux variables aléatoires. De plus, cette indépendance ne peut être obtenue que si les classifieurs sont aléatoires ce qui est rarement le cas en pratique. On considère alors l'*indépendance conditionnelle* qui impose simplement que les classifieurs soient indépendants pour chaque classe. Elle se traduit par :

$$P(s^1 = \omega_i, s^2 = \omega_j|\omega_l) = P(s^1 = \omega_i|\omega_l)P(s^2 = \omega_j|\omega_l),\ \forall \omega_i, \omega_j, \omega_l \in S \tag{2.24}$$

C'est cette indépendance qui est en générale requise et plus particulièrement pour certaines méthodes de combinaison comme la combinaison par le produit [164], la combinaison Bayésienne ou encore par la règle de Dempster-Shafer [19, 18, 127]. Cependant, même si cette condition n'est pas complètement vérifiée, certaines combinaisons peuvent malgré tout s'appliquer et être bénéfiques [19, 18]. Dans [102], les auteurs vont encore plus loin en montrant qu'une dépendance ou corrélation négative est même préférable à une indépendance. En fait, l'idée intuitive sous-jacente consiste à considérer que si des classifieurs font des erreurs différentes, alors une combinaison adéquate doit permettre de donner des résultats meilleurs que chacun des classifieurs pris indépendamment.

Deux points sont donc particulièrement importants ici. Le premier est de déterminer comment obtenir des classifieurs suffisamment indépendants ou négativement corrélés. C'est l'objet du paragraphe 2.2.2.2. Le deuxième point relève quant à lui du mode de combinaison utilisé et de sa sensibilité (cf. paragraphe 2.2.2.4).

2.2.2.2 Comment obtenir des points de vue différents

Ce paragraphe concerne essentiellement le cas des approches génératives puisque pour les approches non-génératives, ces points de vue existent déjà comme nous l'avons déjà mentionné en introduction du paragraphe 2.2.2. L'idée ici consiste à élaborer différents classifieurs en essayant de les rendre les plus indépendants possibles ou négativement corrélés.

D'une manière générale, cette indépendance peut avoir deux origines. Soit elle provient de la nature des classifieurs en eux-mêmes, soit elle résulte de la nature des données utilisées pour leur apprentissage. En s'inspirant de l'énumération faite dans [51], voici un ensemble de solutions pour obtenir des classifieurs ayant des comportements différents.

Celle-ci est rangée approximativement par ordre croissant de l'impact sur la différence entre les classifieurs obtenus :

- utiliser des paramètres d'initialisation différents lorsque cela est possible et que la fonction de décision résultante en dépend ;
- utiliser des paramètres différents : critères d'arrêt ou d'élagage dans un arbre de décision, paramètre de régularisation dans un réseau de neurone, fonction des noyaux dans les machines à vecteurs supports, etc. ;
- utiliser des paramètres d'architecture différents : nombre de couches cachées et de neurones par couche dans un réseau de neurones, arbres de décision binaires ou n-aires, etc. ;
- utiliser différents types de classifieurs [94, 171, 177] ;
- utiliser des bases d'apprentissage différentes, si besoin en les manipulant : bagging [29], etc.
- utiliser des espaces de représentation différents [95, 6].

Cette liste n'est bien sûr ni exhaustive ni restrictive, plusieurs techniques pouvant être utilisées en même temps.

Même si l'utilisation de ces méthodes est sensée produire des classifieurs différents, il est souvent difficile d'évaluer *a priori* à quel point leurs décisions seront réellement indépendantes ou complémentaires. De nombreuses mesures ont donc été élaborées afin d'estimer l'indépendance ou la non corrélation de classifieurs [171, 19, 105, 102, 109]. Des techniques de production de classifieurs puis de sélection des plus intéressants par rapport à la mesure peuvent alors être employés [148].

À présent que les moyens permettant de générer des classifieurs différents (lorsque ceux-ci n'existent pas) ont été présentés, il reste un point important à étudier avant de pouvoir faire la combinaison elle-même. Il s'agit du problème de la nature des sorties des systèmes de reconnaissance utilisés.

2.2.2.3 Combinaison de classifieurs avec des sorties hétérogènes

Comme énoncé dans le paragraphe 1.2.3, parmi l'ensemble des classifieurs existants, tous ne fournissent pas les mêmes informations en sortie. Rappelons que celles-ci peuvent être de type classe, ensemble, rang ou mesure. Pour pouvoir combiner ou fusionner les réponses de plusieurs classifieurs, il faut : soit utiliser des classifieurs fournissant des sorties homogènes ; soit se ramener à une sortie homogène [74] ; ou encore utiliser un processus de combinaison capable de gérer ces classifieurs hétérogènes [127, 18, 177, 161].

Chacun des types de sortie peut en général être converti dans un autre type avec ou sans perte d'information. Par exemple, les sorties de type mesure peuvent être converties dans tous les autres formats[9] mais au prix d'une perte d'information. Les sorties de type classe peuvent être converties sans problème en sorties de type ensemble, rang ou mesure

9. Pour la conversion en sortie de type ensemble, il faut ajouter de l'information *a priori* comme le nombre de classes désiré pour l'ensemble.

car elles n'en représentent que des cas particuliers. Cette conversion s'effectue cette fois sans perte d'information.

Parce qu'ils apportent le plus d'information, les classifieurs à sorties de type mesure sont assez souvent choisis dans les systèmes à multiples classifieurs. Mais avant de pouvoir effectuer celle-ci, il faut bien souvent procéder à une homogénéisation des sorties. En effet, tous les classifieurs utilisés ne produiront pas nécessairement le même genre de mesures. Il pourra s'agir de distances, de probabilités, de scores, etc. Elles ne s'exprimeront donc pas toutes dans le même intervalle de valeur ni sur la même échelle. Bien souvent, il faut opérer une uniformisation des sorties. Il n'existe pas de fonction universelle pour cela. Cependant, plusieurs contraintes sont généralement imposées, notamment afin de se rapprocher des modes de raisonnement usuels. Soit $\{s_1, \ldots, s_i, \ldots, s_{\mathcal{S}}\}$ le vecteur de sortie de type mesure (une par classe i de S) d'un classifieur et $(s'_1, \ldots, s'_i, \ldots, s'_{\mathcal{S}})$ le vecteur obtenu après uniformisation. Les contraintes généralement imposées sont les suivantes :

- $\forall i, s'_i \in [0, 1]$;
- les mesures s'_i sont croissantes avec l'importance accordée à la classe i par le classifieur. Ainsi, pour une entrée donnée, les classes i pour lesquelles $s'_i = 1$ seront les plus privilégiées et celles pour lesquelles $s'_i = 0$ seront celles qui seront écartées par le classifieur ;
- la transformation appliquée pour uniformiser les sorties ne doit pas changer les préférences initiales du classifieur ;
- la fonction d'uniformisation doit être consistente et donc pouvoir s'appliquer à toutes les valeurs possibles des mesures initiales.

D'autres contraintes peuvent aussi être rajoutées pour se rapprocher d'un cadre probabiliste par exemple (ce qui ne garantit absolument pas que les nouvelles sorties obtenues seront des probabilités). Dans ce cas, la condition suivante est rajoutée : $\sum_{i=1}^{\mathcal{S}} s'_i = 1$. Celle-ci est vérifiée en utilisant une méthode de normalisation dont voici les plus courantes :

$$s'_i = \frac{s_i}{\sum_{i=1}^{\mathcal{S}} s_i} \tag{2.25}$$

$$s'_i = \frac{s_i^2}{\sum_{i=1}^{\mathcal{S}} s_i^2} \tag{2.26}$$

Bien que ce genre de normalisation soit souvent nécessaire, il faut y faire très attention. En effet, comme montré dans [10], elles réduisent la dimension de l'espace de sortie, provoquant une perte d'information. Ainsi, des classes initialement séparables dans l'espace de sortie associé à l'ensemble des classifieurs, peuvent devenir non séparable après normalisation. Une grande attention doit donc être portée pour le choix de la fonction de transformation. Dans ce cadre, il semblerait que (2.26) soit meilleure que (2.25) [10].

2.2.2.4 Différentes méthodes de combinaison

Du point de vue de la topologie du système, les approches fonctionnellement indépendantes reposent essentiellement sur une *combinaison parallèle* de N classifieurs utilisant chacun son propre espace de représentation E_i, $i = 1, \ldots, N$, qui peut être éventuellement commun pour certains. Il s'agit donc d'une structuration horizontale du point de vue de l'ensemble de classifieurs. Un module de combinaison est utilisé afin de produire une nouvelle sortie y à partir de l'ensemble des sorties s^i (cf. figure 2.8). C'est pourquoi on parle très souvent de *fusion* des sorties des classifieurs.

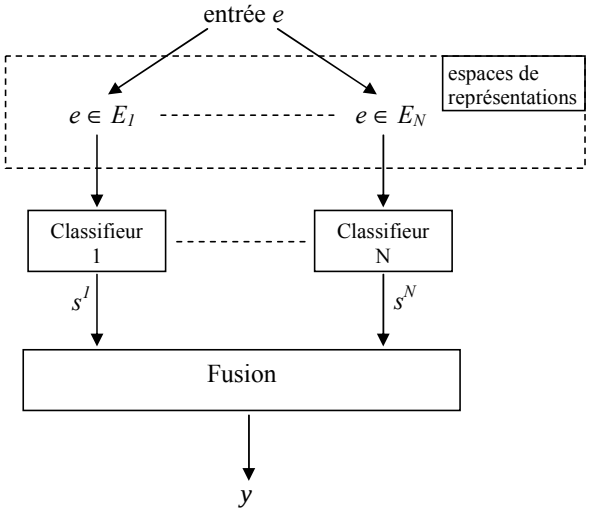

FIG. 2.8 – *Schéma de combinaison parallèle*

La méthode de fusion dépend le plus souvent du type de sortie des classifieurs [127]. Si les sorties sont des classes, des méthodes par votes, pondérés ou non, sont fréquemment utilisées [107]. La classe correspondant à la sortie y est celle qui est représentée majoritairement dans l'ensemble des sorties des classifieurs. Plusieurs variantes existent, notamment en fonction de la fiabilité désirée de la réponse. Ainsi, pour que la classe majoritaire soit acceptée, il est possible de rajouter des conditions telles que : l'unanimité ; l'absence de conflit (cette méthode diffère de la majorité unanime dans le sens où les classifieurs peuvent être « sans opinion » c'est-à-dire rejeter la forme) ; un nombre de voix suffisantes ; la majorité notoire (un nombre de voix suffisant séparant les deux classes les plus majoritaires) ; etc.

Si les sorties s^i sont de type mesure ($s^i = \{s^i_1, \ldots, s^i_s, \ldots, s^i_S\}$), les modes de combinaison les plus courants sont la règle du *maximum* :

$$y_s = \max_{i=1,\ldots,N} s^i_s \ ;$$

du *minimum* :

$$y_s = \min_{i=1,\ldots,N} s_s^i \ ;$$

la moyenne arithmétique :

$$y_s = \frac{1}{N} \sum_{i=1}^{N} s_s^i \ ;$$

et géométrique :

$$y_s = \left(\Pi_{i=1}^{N} s_s^i \right)^{1/N} \ ;$$

ces deux dernières pouvant être pondérées. Dans le cas de classifieurs indépendants, les méthodes multiplicatives donnent en général de meilleurs résultats que les méthodes reposant sur l'utilisation de la somme. Cependant, si les sorties des classifieurs sont sujettes au bruit ou peu fiables, l'utilisation de la somme donne des résultats plus stables alors que le produit peut dégrader très nettement les performances du système, au point que celles-ci deviennent inférieures à celles du meilleur classifieur de l'ensemble [164, 4]. Il faut donc prendre grand soin lorsqu'une telle combinaison est utilisée. Une solution consiste à réduire les « effets de bords » en ne combinant que les décisions les plus fiables [8].

L'ensemble des techniques précédentes utilise une règle de combinaison fixe. Il existe aussi des méthodes de fusion reposant sur un apprentissage de la règle de combinaison. Celui-ci s'effectue à partir d'une base de validation permettant notamment de prendre en compte l'efficacité de chacun des classifieurs de l'ensemble. Ces méthodes donnent en général de bons résultats mais nécessitent que la base de validation soit différente des bases d'apprentissage (pour éviter le sur-apprentissage) et de taille suffisamment importante pour avoir de bonnes propriétés de généralisation [51]. Citons par exemple l'utilisation de la règle de Bayes ou encore la technique du BKS (Behavior Knowledge Space), etc. [52, 93, 80]

Pour plus d'informations sur l'ensemble de ces techniques, le lecteur peut se référer à [7, 104, 127, 19, 18, 177, 161].

Les méthodes de combinaison par fusion de classifieurs énumérées ci-dessus utilisent systématiquement les résultats de tous les reconnaisseurs pour établir la décision finale y. Pourtant, pour une forme donnée en entrée, tous les classifieurs n'auront pas nécessairement la même pertinence du fait de leur indépendance. Certains peuvent être plus efficaces que d'autres pour la reconnaissance de certaines formes. Cette constatation est à l'origine des mécanismes de sélection de classifieurs. À l'inverse des méthodes de fusion, celles-ci considèrent que pour une entrée donnée, il existe un classifieur de l'ensemble qui est mieux adapté pour prendre la décision. La sélection s'effectue par exemple en évaluant l'efficacité de chaque classifieur dans le voisinage de la forme à reconnaître [63]. Des méthodes ont aussi été élaborées de façon à choisir, pour chaque entrée, entre une combinaison par sélection ou par fusion de classifieur [103].

Le mécanisme de sélection peut aussi être utilisé afin de réduire la nombre de classifieurs utilisés pour la combinaison et donc pour accélérer les traitements. Cela a notamment été mis en œuvre dans [137, 136] en hiérarchisant les classifieurs sur deux

niveaux. Le deuxième niveau est un classifieur reposant sur des modèles de classes.[10]. Seuls ceux correspondant aux classes les plus activées par le premier niveau sont utilisés pour la combinaison. L'architecture n'est donc plus parallèle mais de type cascade (cf. paragraphe 2.2.3.1).

2.2.3 Approches de combinaison fonctionnellement dépendantes

Les approches de combinaison fonctionnellement indépendantes présentées dans le paragraphe précédent visent à utiliser un ensemble de classifieurs considérant indépendamment les uns des autres la tâche de classification. Chacun repose sur un point de vue qui lui est propre et c'est cette diversité qui permet d'obtenir de bonnes performances. Une autre façon de concevoir des approches à multiples classifieurs consiste à assigner à chacun des éléments du système un rôle particulier correspondant à une sous-tâche de la fonction principale de classification. Dans ces systèmes, les fonctionnalités des classifieurs sont dépendantes les unes des autres pour la résolution du problème considéré. Seule l'utilisation de l'ensemble des classifieurs et donc la réalisation de chacune des sous-tâches auxquelles ils correspondent permet à l'ensemble de fonctionner correctement. C'est pourquoi nous qualifieront ces approches de *fonctionnellement dépendantes*. Leur intérêt est double : elles permettent d'une part de résoudre la tâche de classification principale par la résolution de sous-tâches plus simples grâce à des classifieurs spécialisés et d'autre part, elles offrent la possibilité de réaliser des objectifs secondaires beaucoup plus facilement que si une seule approche était utilisée. Elle peuvent ainsi favoriser l'interprétabilité et la maintenance du système en conférant une certaine modularité à l'ensemble, ou encore utiliser des architectures plus adaptées pour la gestion du refus de classement et du rejet de distance.

Plusieurs distinctions peuvent être faites, selon la nature du lien qui unit les classifieurs de l'ensemble. Nous distinguons ici essentiellement deux possibilités (ce qui n'est pas nécessairement exhaustif) que nous allons détailler dans les paragraphes suivants : les *approches de combinaison correctives* et les *approches de combinaison par experts*. Les premières reposent sur l'utilisation d'un classifieur principal dont les performances sont jugées insuffisantes (décisions pas assez fiables et/ou taux de reconnaissance trop faibles). L'objectif principal consiste alors à utiliser d'autres classifieurs afin de corriger les décisions du premier.

Les approches de combinaison par experts sont quant à elles issues de la volonté de décomposer le problème initial en sous-problèmes. Un classifieur spécialisé est ensuite élaboré pour traiter chacun d'eux séparément et leurs résultats sont combinés. Rappelons encore une fois que la distinction entre ces différentes approches n'est pas stricte. Il est tout à fait envisageable par exemple qu'un système repose sur un mécanisme de correction lui-même établit à partir d'experts. En pratique, c'est même assez souvent le cas.

10. Ce reconnaisseur est donc lui même un système à multiples classifieurs qui sont fonctionnellement dépendants les uns des autres. Plus précisément, il s'agit d'une approche par experts (un par classe) comme décrit dans le paragraphe 2.2.3.2.

2.2.3.1 Approches de combinaison correctives

Lorsque l'on traite un problème complexe de reconnaissance et que l'approche de classification dont on dispose a des performances insuffisantes, il est possible d'utiliser d'autres classifieurs pour essayer de corriger ses erreurs. Les approches correctives utilisent la plupart du temps un classifieur principal qui traite le problème de reconnaissance dans son intégralité. Un ou plusieurs autres modules de reconnaissance sont utilisés ensuite pour traiter les cas difficiles que le système principal ne peut résoudre lui-même.

Ces systèmes reposent le plus souvent sur des schémas de combinaison de type série ou série-parallèle (cascade) dans lesquels les classifieurs sont hiérarchisés (structuration verticale). La sortie d'un classifieur est donc utilisée comme entrée d'un ou plusieurs autres qui peuvent éventuellement réutiliser en plus le vecteur d'entrée. Dans le cas de la combinaison série la sortie y du système est celle du dernier classifieur de la chaîne (cf. figure 2.9). Dans le cas de la combinaison cascade, elle peut soit être récupérée au niveau d'un des classifieurs de la chaîne, soit être une combinaison d'une partie des sorties (cf. figure 2.10).

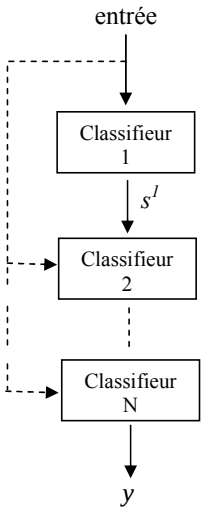

FIG. 2.9 – *Schéma de combinaison série.*

Les techniques issues du *weak learning* et notamment celles de *boosting* [60] font partie de ces approches. Ces dernières utilisent une chaîne de classifieurs entraînés sur la base d'apprentissage dans laquelle chaque individu possède un poids. Si au début tous sont aussi représentatifs, leur poid augmente lorsqu'ils sont mal classés par un classifieur. Le reconnaisseur suivant dans la chaîne se focalise donc davantage sur la classification de ces individus afin d'améliorer les performances de l'ensemble. La sortie est une somme pondérée des sorties de chaque classifieur de la chaîne en fonction de leurs performances sur la base ayant servie à leur apprentissage.

D'autres types de systèmes utilisent un principe de combinaison similaire pour ré-

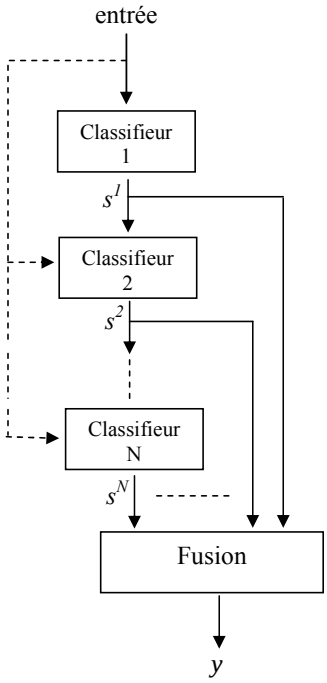

FIG. 2.10 – *Schéma de combinaison série-parallèle (cascade).*

soudre explicitement les cas difficiles à traiter pour le classifieur principal. Ce mécanisme est possible si le classifieur principal met en œuvre le refus de classement [81] comme nous l'avons présenté dans le paragraphe 1.2.5.1. La topologie de ce mode de combinaison, qualifiée de *conditionnelle* dans [106], permet d'optimiser le processus de classification en limitant l'utilisation des ressources. En effet, le classifieur principal peut être de complexité modérée voire faible afin de minimiser les coûts de traitement dans la majorité des cas. Les classifieurs spécialisés peuvent en revanche utiliser des représentations et des techniques de classification plus coûteuses afin d'améliorer les performances uniquement pour les cas difficiles [9, 64, 6]. Une variante des approches précédentes consiste non plus à utiliser un seul classifieur pour résoudre les cas difficiles mais plusieurs. Ainsi, dans [20] par exemple, les auteurs construisent une machine à vecteur support pour faire la discrimination de chaque paire de classes qui sont sources de confusions après l'utilisation d'un classifieur principal de type PMC. Ce principe est aussi utilisé dans [139] en combinant des réseaux de neurones pour faire la discrimination des classes restant en confusion après l'utilisation d'un classifieur reposant sur une modélisation explicite des classes par prototypes.

Les méthodes de *stacked generalization* [175] sont aussi un exemple d'approches de combinaison correctives. Dans celles-ci, un classifieur est utilisé afin de corriger le comportement d'un ou plusieurs autres. Il effectue son apprentissage dans un espace de représentation intermédiaire correspondant aux sorties des classifieurs principaux. Elles repose aussi souvent sur le principe des approches de combinaison par experts (cf.

paragraphe suivant) puisque les classifieurs qui sont corrigés sont très souvent appris sur des régions précises de l'espace des données.

2.2.3.2 Approches de combinaison par experts

Ces approches traitent les problèmes de classification complexes en utilisant la méthode « diviser pour régner ». Elles procèdent donc par une décomposition du problème initial en sous-problèmes. Cette décomposition peut se faire soit sur l'espace de représentation, soit sur l'espace de sortie et le schéma de combinaison le plus courant est la combinaison parallèle (un expert par sous-problème).

Les méthodes les plus courantes pour diviser l'espace de représentation consistent à faire un découpage selon le principe de la validation croisée [5]. La base d'apprentissage est divisée en k sous-bases disjointes et k classifieurs sont appris sur chacune de ces sous-bases. La décision est faite le plus souvent par une règle de combinaison classique comme vue dans le paragraphe 2.2.2.4. Les « mélanges d'experts » (*mixture of experts*) [84, 160] sont une variante des méthodes précédentes. Les experts sont appris de la même façon à partir d'une partition de la base d'apprentissage. En revanche, la combinaison est faite par un classifieur spécialisé qui pondère chacune des sorties avant que celles-ci soient sommées.

Les approches utilisant une décomposition de l'espace de sortie sont aussi assez fréquentes. Elles sont notamment utilisées lorsque l'on souhaite résoudre un problème multi-classes avec un classifieur binaire ne permettant de faire que la discrimination de deux classes. La décomposition consiste le plus souvent à apprendre un classifieur pour faire la discrimination d'une classe contre toutes les autres, et ce pour chaque classe (approches *one against rest*). Ces techniques sont fréquentes pour l'utilisation de machines à vecteurs supports [79, 124] et d'arbres de décision [121] par exemple. Une variante consiste à élaborer un classifieur pour faire la discrimination de chaque paire de classes. Cette solution est cependant moins économique et son efficacité dépend de la méthode de combinaison utilisée [163].

Un autre exemple d'utilisation de la décomposition de l'espace de sortie se retrouve dans les systèmes reposant sur la génération d'un modèle par classes par une modélisation descriptive. L'extension à une décomposition en sous-classes a aussi été envisagée comme par exemple dans le système *ResifCar* ou encore dans [62].

Le mécanisme de décision le plus simple associé à ces approches consiste à mettre les experts en compétition. La classe choisie est celle correspondant à l'expert ayant le meilleur score (dans le cas de sorties de type mesure bien entendu). On remarquera que dans le système *ResifCar*, cette compétition est un peu particulière puisqu'elle intervient sur chacun des niveaux des modèles (cf. paragraphe 1.4.1).

2.2.4 Une nouvelle approche : la combinaison structurée guidée par les données

Les approches à multiples classifieurs ont montré leur intérêt dans de nombreuses applications, aboutissant la plupart du temps à des systèmes très performants. Pour notre approche, l'utilisation de plusieurs classifieurs peut aussi être bénéfique et ce à plusieurs points de vue. Ils peuvent non seulement permettre d'obtenir de bonnes performances, mais aussi de traiter plus facilement des problèmes complexes en les décomposant. Enfin, ils peuvent être utiles pour favoriser la réalisation de nos autres objectifs tels que l'interprétabilité, en structurant l'ensemble pour le rendre plus modulaire et explicite, ou encore d'améliorer la fiabilité, en introduisant un module pour le rejet de distance ou le refus de classement par exemple.

L'inconvénient dans la conception de ces approches de combinaison est que le choix des composants par rapport au problème à traiter reste souvent assez obscur et arbitraire. Même si la définition d'objectifs secondaires et donc la détermination du lien fonctionnel entre les éléments du système et le problème de reconnaissance est une aide précieuse, il manque encore actuellement des informations précises permettant de guider les choix de conceptions. Nous détaillons ces problèmes dans le paragraphe suivant. Suite à ce constat, nous proposons, pour notre approche de reconnaissance, de s'appuyer sur la structure des données dans l'espace de représentation afin de guider l'élaboration du mode de combinaison.

2.2.4.1 Problèmes de conception avec les approches de combinaison usuelles

Dans [75, 76], une critique est faite sur l'utilisation de multiples classifieurs, soulevant un certain nombre de questions sur la manière dont l'élaboration de ces approches est actuellement envisagée et les problèmes qui peuvent en résulter. Une des principales constatations est l'absence d'étude théorique complète sur le comportement des classifieurs par rapport aux problèmes considérés et par rapport à leurs possibles interactions. C'est pourquoi, les systèmes qui reposent explicitement sur la nécessité d'indépendance des classifieurs peuvent avoir des comportements inattendus. Cette contrainte est en effet très forte et souvent éloignée de la réalité puisque les décisions des classifieurs sont intrinsèquement corrélées car liées par la forme à reconnaître. De plus, cette corrélation peut varier d'une entrée à l'autre. Même si cette contrainte d'indépendance est relaxée en imposant plus simplement d'avoir des classifieurs faisant des erreurs différentes, la meilleure manière d'y parvenir n'est pas évidente. De combien de classifieurs a-t-on besoin dans tel ou tel cas ? Quel classifieur utiliser pour tel espace de représentation ou telle base d'apprentissage ? Quel sont les classifieurs qui se complètent le mieux pour un problème donné ? Bien souvent, ces questions sont laissées de côté, l'attention étant plus particulièrement portée sur la méthode de combinaison qui doit se « débrouiller » pour exploiter au mieux l'interdépendance ou l'indépendance des classifieurs utilisés. Mais là encore, il est difficile de trouver le meilleur mode de combinaison en fonction des classifieurs utilisés et du contexte d'utilisation.

Les approches fonctionnellement dépendantes possèdent alors un avantage certain :

le choix des éléments du système ainsi que leur mise en relation sont guidés par le rôle que chacun d'eux doit remplir au sein du système. Malgré tout, les différentes possibilités restent nombreuses et il est difficile de trouver une méthode optimale. Comme le montre T.K. Ho [75, 76], il semble donc nécessaire d'élaborer un « langage » afin de décrire les problèmes de classification et le comportement des classifieurs en fonction de celles-ci. Cette description doit se baser sur les données du problème, notamment en établissant la pertinence du jeu de caractéristiques mais aussi la densité, la couverture et donc la structure des données dans l'espace de représentation. Car même si un problème contient une composante stochastique, il devrait y avoir une structure sous-jacente régissant le processus [77].[11] Avec un tel outil, il deviendrait théoriquement possible de choisir spécifiquement des classifieurs adaptés pour résoudre chaque problème de reconnaissance, que ce soit par des approches mono-classifieur ou multi-classifieurs. Mais les recherches dans ce sens ne sont que très récentes [76, 77, 158].

2.2.4.2 Le mode de combinaison proposé

Puisqu'il est difficile d'établir *a priori* le meilleur système et par extension le meilleur schéma de combinaison pour un problème donné, nous choisissons ici de structurer notre approche en nous basant sur les propriétés des formes, en utilisant une combinaison structurée guidée par les données. Le schéma-bloc correspondant à notre système de reconnaissance *Mélidis* (ModÉLisation Intrinsèque-DIScriminate), est représenté sur la figure 2.11.

Afin de pouvoir traiter les problèmes difficiles tout en conservant une bonne lisibilité, nous utilisons un premier module, le *module de description intrinsèque par prototypes*, dont l'objectif est triple. Il doit d'une part pouvoir offrir un premier niveau de lisibilité du problème en le décrivant dans son ensemble. Pour cela, la modélisation repose sur une description des formes en s'appuyant sur leurs propriétés intrinsèques dans l'espace de représentation pour faciliter l'interprétabilité (*niveau de modélisation intrinsèque*). Cette description doit en outre permettre de mettre en lumière les principales sources de confusion entre les classes. Celles-ci sont alors utilisées pour décomposer le problème initial en sous-problèmes (*décomposition guidée par les données*). Enfin, ce module doit permettre de contrôler le *rejet de distance*. Cette fonctionnalité est intégrée ici afin de pouvoir le gérer par une modélisation explicite par prototypes et non pas par une modélisation discriminante et ce pour les raisons évoquées dans le paragraphe 1.2.5.2.

Un deuxième module, le *module de discrimination focalisée*, effectue une modélisation discriminante fine et ciblée au sein des sous-problèmes (sources de confusions) déterminés par le module de description intrinsèque (*niveau de modélisation discriminante*). Cette modélisation doit aussi utiliser des connaissances interprétables et donc s'appuyer sur les propriétés des données dans l'espace de représentation, mais cette fois-ci sur les propriétés discriminantes.

11. La plupart des processus physiques considérés comme chaotiques reposent en fait sur une structure qui leur est propre [2].

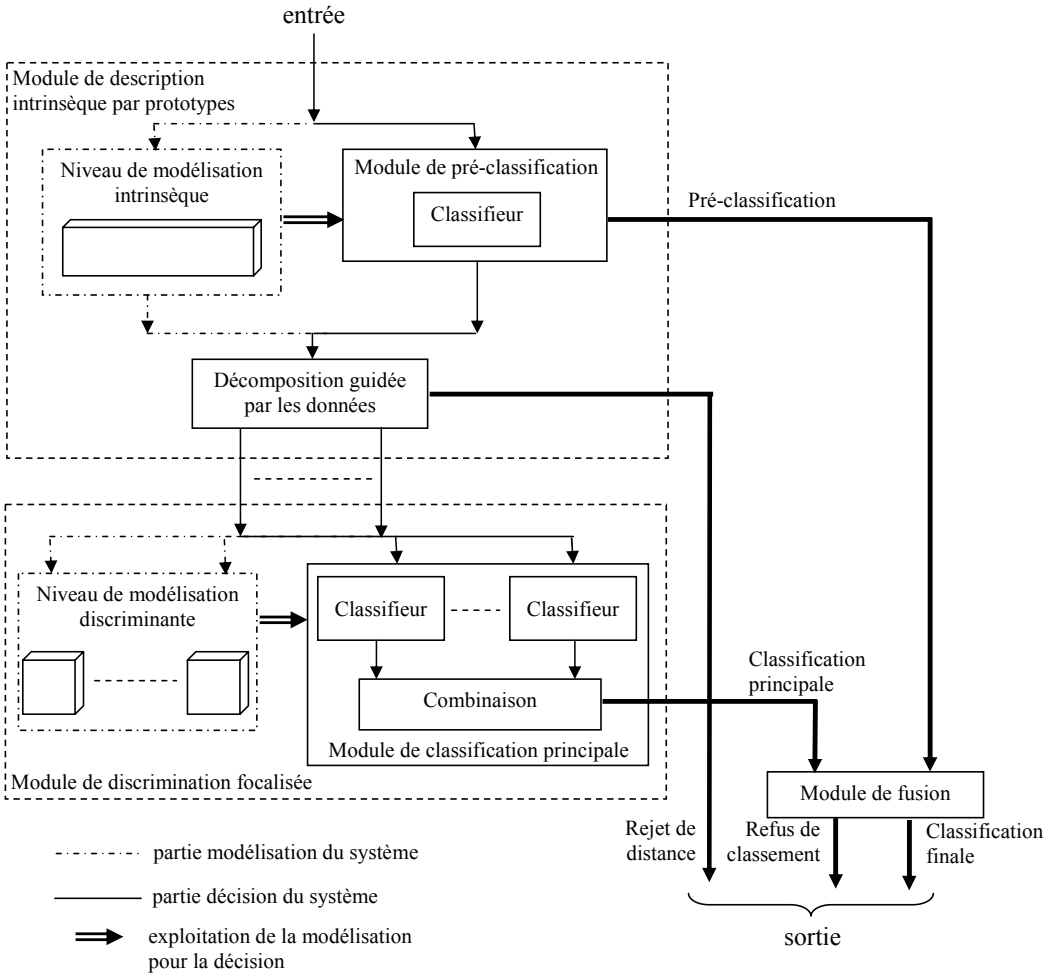

FIG. 2.11 – *Schéma-bloc du système Mélidis, s'appuyant sur une combinaison structurée guidée par les données.*

Pour la classification, puisque l'utilisation de points de vue différents est un facteur permettant d'accroître les performances, chacun des deux modules fournit sa propre décision sous la forme de scores relatifs à chaque classe (sorties de type mesures). Ainsi, le module de description intrinsèque par prototypes utilise un module de pré-classification. Celui-ci contient un classifieur qui exploite ($\Rightarrow$) la modélisation intrinsèque. De même le module de discrimination focalisée utilise un module de classification principale qui contient un classifieur expert par sous-problèmes issus de la décomposition et qui exploite ($\Rightarrow$) la modélisation discriminante. On remarquera que la classification principale repose sur la combinaison des sorties de ces classifieurs experts. Ces deux décisions intermédiaires sont fusionnées pour obtenir la classification finale qui peut intégrer la gestion du refus de classement.

Ce schéma de combinaison est donc structuré verticalement et horizontalement sous la forme d'une « cascade arborescente ». On remarquera que dans celle-ci, la structuration verticale correspond à la fois à une dépendance fonctionnelle entre les deux modules (description explicite et discrimination) et à une dépendance par rapport aux propriétés intrinsèques des formes, ce qui distingue assez nettement notre approche des modes de combinaison usuels. Cette dernière est particulièrement intéressante pour trois raisons.

La première est qu'elle permet au module de discrimination focalisée de se concentrer uniquement sur les formes possédant des propriétés similaires (celles déterminées par le module de description intrinsèque) et qui sont donc potentiellement difficiles à distinguer. Ce mécanisme de focalisation permet de réduire au mieux la taille des sous-problèmes (à la différence d'une approche de décomposition ordinaire comme présentée en 2.2.3.2), ce qui devrait permettre une plus grande efficacité.

Cette focalisation permet en outre d'accroître la robustesse du lien entre les deux modules. Dans les approches utilisant le résultat de la classification d'un premier niveau pour construire/sélectionner les classifieurs à utiliser au deuxième niveau, cette sélection est tributaire de la robustesse du mécanisme de décision. Il peut alors arriver que pour une faible variation de la forme en entrée, celle-ci soit traitée par des classifieurs complètement différents. La figure 2.12 (a) illustre ce phénomène de façon simple pour un problème à trois classes ($\omega_1, \omega_2, \omega_3$). Les frontières de décision d'un classifieur utilisé comme premier étage d'un système à deux niveaux y sont représentées. Si le deuxième étage dépend de la décision du premier, les formes noires à classer seront traitées différemment alors qu'elles possèdent des propriétés très similaires dans l'espace de représentation (il peut d'ailleurs s'agir de la même forme mais légèrement bruitée). Ce problème est en revanche correctement absorbé si le lien entre les deux niveaux se base sur les propriétés des formes dans l'espace de représentation (b).

Enfin, la troisième raison est que cette structuration permet de rendre l'ensemble modulaire. En effet, on conserve d'un côté la modélisation du problème favorisant son interprétabilité et de l'autre le mécanisme de décision que l'on peut optimiser de façon séparée, sans remettre en cause la modélisation. De plus, le processus de décomposition du problème peut facilement opérer à différents niveaux de précision sans remettre en cause l'architecture de l'ensemble. Par exemple, pour des problèmes de complexité modérée, les experts du niveau de discrimination focalisée pourront fonctionner au niveau d'une décomposition en classes. Pour des problèmes plus difficiles, leur précision pourra être du niveau de la sous-classe si celles-ci ont été identifiées par le module de description intrinsèque . Dans les approches plus classiques à deux étages, le deuxième niveau opère généralement sur les sorties du premier, ce qui limite sa précision au niveau de la classe (que ce soit pour discriminer une classe contre toutes les autres ou bien les paires de classes à l'origine de confusions) [20, 139].

Outre ces propriétés liées à la dépendence entre les modules, une deuxième originalité importante de notre système réside dans l'utilisation systématique de classifieurs reposant sur des connaissances et un formalisme interprétable, s'appuyant sur la structure des données dans l'espace de représentation. Cela permet de rendre l'ensemble plus homogène facilitant ainsi la fusion des deux niveaux de classification (cf. chapitres 3 et 4).

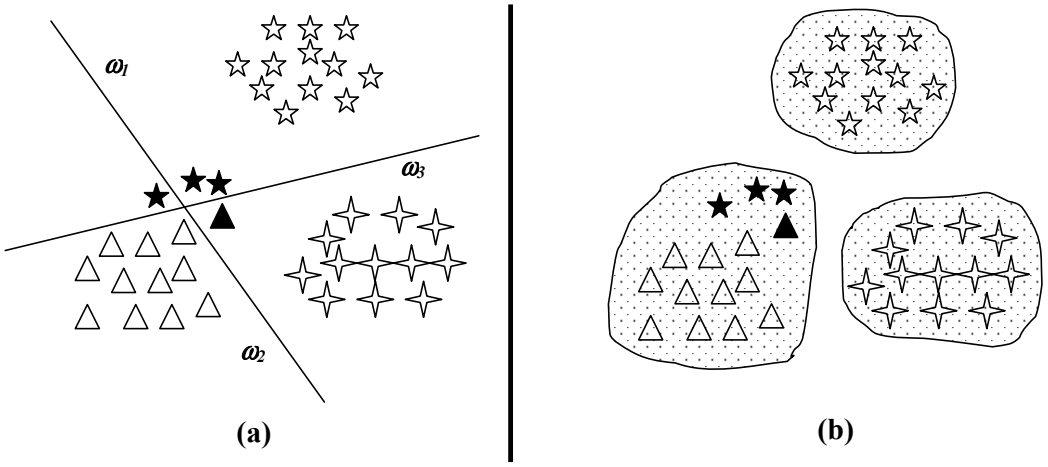

FIG. 2.12 – *Exemples de problèmes relatif à l'utilisation directe des sorties d'un classifieur pour une combinaison à deux étages. Si le deuxième étage utilise les sorties du premier, les formes noires, qui sont semblables, seront traitées différemment (a). Ce problème est absorbé dans le cas de la combinaison structurée guidée par les données (b).*

2.3 Bilan

Dans ce chapitre, nous avons présenté les mécanismes généraux essentiels pour la résolution de notre problématique. Ceux-ci sont relatifs d'une part à la représentation des connaissances dans le cadre de données imprécises et sujettes à la variabilité et d'autre part à l'architecture générale de notre approche.

Le premier point a été traité en introduisant les principes de la théorie des sous-ensembles flous qui, en plus de la robustesse face aux imprécisions, permettent de synthétiser les données sous une forme plus interprétable. Leur utilisation pour la mise en œuvre de raisonnements et plus particulièrement pour la prise de décision en reconnaissance de formes a ensuite été présentée.

Le problème de la conception de l'architecture de notre méthode de reconnaissance a été envisagée dans le cadre de la combinaison de classifieurs. Afin de répondre à nos objectifs de robustesse et de performances, de modularité et d'interprétabilité, nous avons présenté un nouveau type de combinaison, structurée et guidée par les données. Celle-ci repose sur la hiérarchisation de deux modules liés l'un à l'autre par les propriétés intrinsèques des formes dans l'espace de représentation. Chaque module possède sa propre modélisation et un mécanisme de décision séparé. Ces derniers sont fusionnés pour la classification finale.

Dans le chapitre suivant, nous décrivons en détails la modélisation de chacun des deux modules ainsi que les méthodes d'apprentissage associées. Le formalisme ainsi que les mécanismes utilisés pour la décision seront ensuite présentés dans le chapitre 4.

Chapitre 3

Extraction et structuration des connaissances dans notre modélisation : méthodologie et apprentissage

Dans l'architecture de notre approche, les modules de description intrinsèque par prototypes et de discrimination focalisée reposent sur une modélisation qui leur est propre pour pouvoir remplir leur rôle dans le système. Le but de la phase d'apprentissage est d'extraire des données disponibles l'ensemble des connaissances permettant d'aboutir à cette modélisation pour un problème donné. Mais avant d'arriver à cette étape, il est important de définir précisément quelles connaissances utiliser pour remplir quelles fonctions ainsi que les propriétés qu'elles doivent posséder pour y parvenir. On pourra ensuite en déduire les mécanismes à utiliser pour les extraire et le mode de représentation à adopter.

Dans la définition 3 du paragraphe 1.2.4.1, nous avons introduit la notion de connaissances dans un système de reconnaissance de formes en présentant celles-ci de façon très générale. Il existe bien entendu différentes natures de connaissances que l'on doit pouvoir répertorier selon leurs propriétés et leurs usages au sein d'un système. Notre objectif n'est pas ici de produire une telle taxonomie.[1] Cependant, il nous semble important d'insister sur les conditions qu'elles doivent remplir ici pour pouvoir atteindre les objectifs que nous nous sommes fixés.

Dans ce chapitre, nous présentons les différentes natures de connaissances que nous avons choisi d'utiliser pour établir la modélisation de notre système. Nous présentons ensuite les mécanismes d'extraction et de modélisation associés à chacun des deux mo-

1. Celle-ci pourrait en revanche s'avérer intéressante pour élaborer un langage de description des systèmes de reconnaissance et éventuellement permettre d'établir un lien avec les différentes natures de problèmes à traiter. Le choix ou la conception d'un système pour un problème donné s'en trouverait alors facilité (cf. paragraphe 2.2.4.1).

dules du système, définissant ainsi le processus d'apprentissage.

L'exploitation de cette modélisation et des connaissances associées au travers des modules de pré-classification, de classification principale et de fusion sera ensuite présentée dans le chapitre 4.

3.1 Les connaissances utilisées pour la modélisation du système *Mélidis*

À partir de l'ensemble des fonctionnalités du système présentées dans le paragraphe 2.2.4.2 ainsi que des objectifs définis en 1.4.2, nous avons déterminés essentiellement deux natures de connaissances différentes : les connaissances de nature *intrinsèque* aux classes et les connaissances de nature *discriminante* par rapport aux classes. Nous détaillons ci-dessous chacune d'elle en précisant leurs rôles et leurs propriétés. Nous décrivons ensuite comment ces connaissances sont structurées entre elles en fonction de leur nature et des objectifs fixés.

3.1.1 Les connaissances de nature intrinsèque aux classes

Les connaissances intrinsèques sont utilisées pour décrire le ou les caractères intrinsèques d'un objet. Rappelons que ces derniers, sont définis comme étant ce qui appartient à un objet, en lui-même, et non dans ses relations avec un autre.

Dans le cadre de la description de classes, ces connaissances définissent donc de façon « absolue » un modèle ou patron[2] qui peut être plus ou moins complexe. Il peut s'agir par exemple d'un ensemble de prototypes représentant les caractères les plus typiques[3] des classes (par le biais d'exemples de formes, fictives ou réelles [138], etc.). Les modèles d'allographes utilisés dans *ResifCar* [16] (cf. paragraphe 1.4.1) sont aussi des exemples de modèles, plus complexes, où chaque primitive d'un allographe est elle-même caractérisée de façon intrinsèque par un ou plusieurs prototypes flous.

3.1.1.1 Leurs rôles dans notre approche

Dans le contexte de notre approche, nous utilisons ces connaissances dans le module de description intrinsèque par prototypes afin de caractériser chacune des classes du problème indépendamment les unes des autres. Cela permet au module de remplir l'ensemble de ses fonctions.

Tout d'abord, en caractérisant les classes par leurs propriétés intrinsèques, on opère une description générale du problème. Bien que ce type de modélisation nécessite comme connaissance *a priori* de connaître les classes, cela ne va pas à l'encontre de l'objectif de généricité puisque nous travaillons dans le cadre d'un apprentissage supervisé (cf. paragraphe 1.4.2.1). Dans un espace de représentation numérique, cette description se traduit ici par la délimitation de régions de l'espace correspondant aux valeurs les plus typiques de chaque classe, formant des prototypes. Ceux-ci sont définis par leur forme et leur position. S'il y en a plusieurs qui correspondent à une même classe, c'est que celle-ci possède un caractère multimodal. Chaque prototype décrit alors une sous-classe (cf. figure 3.1).

2. On retrouve ici la notion de motif utilisée dans [56].

3. La notion de typicalité diffère d'un auteur à l'autre [147, 46]. Par exemple, dans [147], la définition de typicalité inclue explicitement un critère de dissemblance avec les autres classes, que nous ne considérons pas pour nos connaissances intrinsèques aux classes.

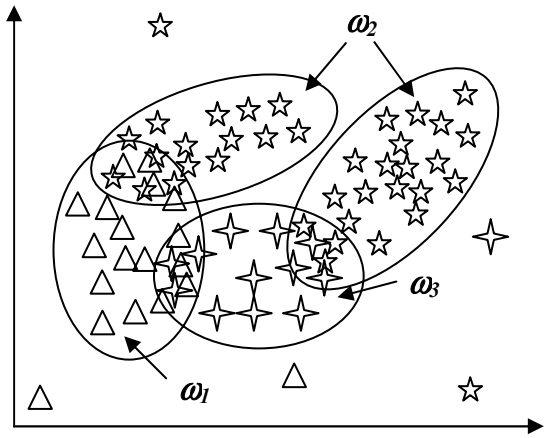

FIG. 3.1 – *Illustration de la description de trois classes par des prototypes intrinsèques dans un espace à deux dimensions. Ces prototypes ne tiennent pas compte des valeurs non typiques.*

Grâce à ce type de description, il est aussi possible de décomposer le problème initial en sous-problèmes plus simples. Il suffit pour cela de constituer un sous-problème par classe. Un sous-problème est constitué de l'ensemble des formes ayant des propriétés intrinsèques similaires à celles de la classe considérée (cf. figure 3.2 (a)). Cette décomposition a l'avantage d'être étroitement reliée à la complexité du problème. En effet, même si plusieurs problèmes qui possèdent le même nombre de classes n'ont pas tous la même complexité, celle-ci est en générale d'autant plus importante qu'il y a de classes. On remarquera aussi que pour les problèmes très complexes, la décomposition peut être facilement adaptée en effectuant un découpage sur la base des sous-classes directement (cf. figure 3.2 (b)). Cependant, pour des raisons de simplicité, nous ne décrivons ici que les processus associés à une décomposition sur la base des classes (mais la description intrinsèque des classes modélise bien les sous-classes lorsqu'elles existent).

Grâce à leur nature même, ces connaissances permettent aussi la mise en œuvre du rejet de distance. En effet, la description des classes étant explicite et par prototypes, il est possible de comparer une forme donnée à chacun des prototypes pour évaluer sa ressemblance. Si la forme ne correspond à aucun d'eux, elle ne peut être correctement identifiée par le système et sera donc rejetée. Nous reparlerons plus précisément de cela dans le paragraphe 4.3.1.

3.1.1.2 Leurs propriétés

Dans notre contexte, les connaissances intrinsèques doivent posséder un certain nombre de propriétés pour ne pas aller à l'encontre des objectifs fixés.

La définition même d'un caractère intrinsèque impose à ces connaissances qu'elles soient déterminées de façon *absolue*, c'est-à-dire en dehors de toute comparaison avec

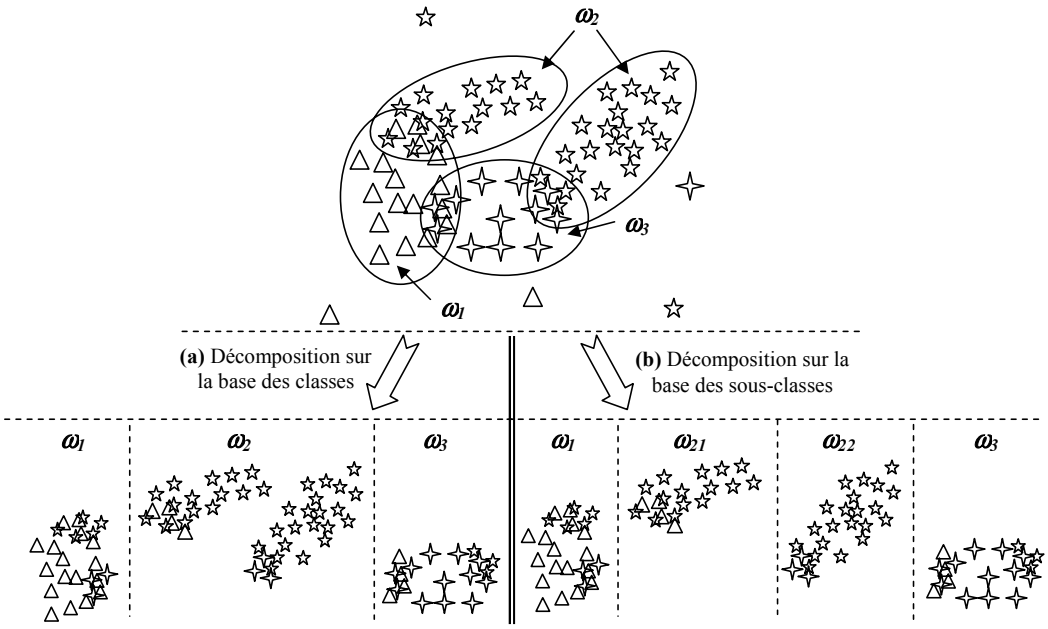

FIG. 3.2 – *Illustration de la décomposition du problème à partir de la description intrinsèque des classes par des prototypes : sur la base des classes (a) ; sur la base des sous-classes (b).*

les autres classes. Elles doivent en outre être *représentatives* des propriétés de la classe.[4]

Une autre propriété essentielle est qu'elles soient *interprétables* au moins du point de vue du concepteur. En plus de la propriété de représentativité énoncée ci-dessus, cette interprétabilité n'est possible que si le formalisme utilisé pour les représenter est compréhensible.

Comme les connaissances intrinsèques sont utilisées pour décomposer le problème, il est important qu'elles représentent des propriétés *stables* des classes. Cela implique notamment une bonne *immunité au bruit* lors du processus d'extraction afin que les données erronées ou peu représentatives n'influent pas sur la détermination des prototypes (cf. figure 3.1).

D'une façon générale, il est aussi important que les connaissances extraites soient représentées d'une façon *robuste*, notamment pour pouvoir absorber les problèmes liés à la variabilité et à l'imprécision des formes. Cela passe par une modélisation qualitative souple et non une représentation quantitative stricte (numérique). Cette granularité permet aussi de synthétiser les connaissances, les rendant plus *compactes*, ce qui va dans le sens de nos objectifs de compacité et d'interprétabilité.

4. Un parallèle plus approfondit devrait être fait ici avec le vocabulaire utilisé pour la détermination de prototypes flous, notamment en terme de mesure de représentativité, de compatibilité, de spécificité, etc. Cependant, les processus d'extraction des prototypes utilisés ici ne reposent pas explicitement sur de telles mesures. Pour un ensemble de référence, le lecteur peut consulter [147].

3.1.2 Les connaissances de nature discriminante

Les connaissances de nature discriminante sont utilisées lorsque l'on souhaite faire la distinction de plusieurs entités en mettant en avant ce qui les différencie. Elles représentent des informations de dissemblance et ont donc une fonction complètement différente des connaissances de nature intrinsèque. Elles peuvent être représentées de plusieurs façons, que ce soit par des fonctions discriminantes (simples ou complexes), ou encore par la présence (resp. absence) d'un caractère d'une entité qui est absent ou différent (resp. présent) sur l'autre entité.

3.1.2.1 Leur rôle dans notre approche

Ces connaissances sont utilisées pour faire la modélisation dans le module de discrimination focalisée. Plus précisément, elles sont employées par chacun des experts sur chacun des sous-problèmes déterminés par le module de description intrinsèque (rappelons que dans notre cas il y a un sous-problème par classe (voire sous-classe)). Dans un sous-problème, la discrimination s'effectue entre les formes de la classe (voire sous-classe) à l'origine de la création du sous-problème et une classe fictive représentée par les formes des autres classes qui possèdent des propriétés intrinsèques similaires (et uniquement celles là). La figure 3.3 reprend l'exemple de la figure 3.2 pour illustrer ce mécanisme. Dans ce sous-problème, la classe réelle ω_1 doit être discriminée de la classe fictive $(\omega_2 + \omega_3)$ constituée uniquement par les 4 étoiles de ω_2 et les 3 étoiles de ω_3.

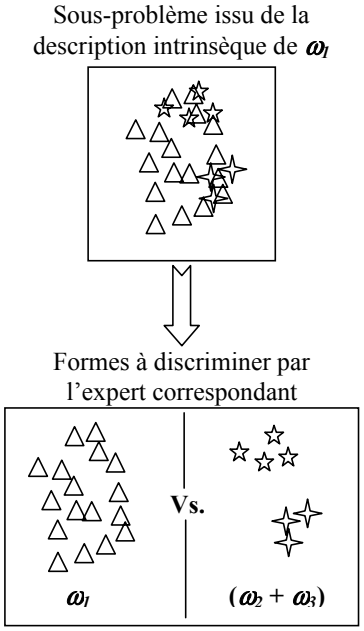

FIG. 3.3 – *Classes à discriminer (ω_1 Vs. $(\omega_2 + \omega_3)$) dans un sous-problème issu de la description intrinsèque de la classe ω_1.*

Comme leur rôle est de permettre la distinction entre les classes (réelles ou fictives), elles doivent matérialiser dans l'espace de représentation des frontières séparatrices.

3.1.2.2 Leurs propriétés

La particularité essentielle de ces connaissances est qu'elles sont *contextuelles* (ou relatives), c'est-à-dire qu'elles n'ont d'intérêt que dans le cadre de la discrimination des deux entités pour lesquelles elles ont été extraites. Par exemple, si le contexte consiste à faire la discrimination d'un 'a' et un 'u', la différence entre la fermeture des deux caractères est une connaissance discriminante intéressante (cf. figure 3.4). Elle n'a en revanche que peu d'intérêt pour faire la discrimination d'un 'u' et d'un 'y'.

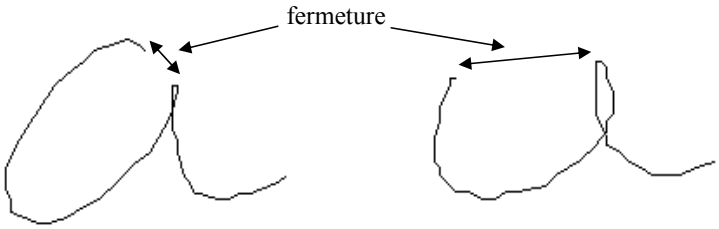

FIG. 3.4 – *Fermeture du 'a' comme critère de discrimination par rapport au 'u'.*

Dans le cadre des sous-problèmes issus de la description intrinsèque, le contexte se matérialise par l'ensemble des formes relatives aux deux classes (fictives ou réelles) à discriminer. Il peut s'agir de faire la distinction entre une classe et un ensemble de classes, entre deux classes, ou entre deux sous-classes appartenant à deux classes différentes. D'une façon générale, le contexte est représenté par deux ensembles de formes issus de classes différentes. Pour reprendre l'exemple de la figure 3.3, le contexte associé au sous-problème consistant à faire la discrimination entre ω_1 et $(\omega_2 + \omega_3)$ est représenté par la classe ω_1 dans son ensemble (tous les triangles du problème initial illustré par la figure 3.1) et les 7 étoiles représentant la classe fictive $(\omega_2 + \omega_3)$. De cette propriété de contextualité découle un certain nombre d'autres propriétés.

Ainsi, pour un contexte fixé, il peut exister plusieurs connaissances permettant de faire la discrimination des entités considérées. Par exemple, pour distinguer un 'u' d'un 'o', la fermeture du caractère ou encore sa courbure totale sont deux connaissances discriminantes utiles. Elles sont donc *multiples*. Cependant, toutes n'auront pas le même *pouvoir de discrimination* dans le même contexte.

On constate par les exemples précédents que l'information utilisée par ces connaissances est essentiellement portée par quelques attributs de l'espace de représentation. Plus particulièrement, dans un espace de représentation numérique, elles sont *associées à un sous-espace* de l'espace de représentation complet dans lequel sont décrites les formes. Dans ces sous-espaces, les connaissances discriminantes doivent définir des *frontières séparatrices*. Pour que celles-ci soient *robustes* (avec un fort pouvoir de généralisation), leur placement doit être optimisé en s'appuyant sur les propriétés des formes dans cet espace. De plus, cela favorise leur *interprétabilité*.

Enfin, comme pour les connaissances intrinsèques, on souhaite que leur représentation soit *compacte*.

3.1.3 Structuration des connaissances

L'ensemble des connaissances qui vont être extraites doivent être structurées les unes par rapport aux autres pour établir leurs liens de dépendance. On en distingue deux types ici.

Le premier traduit la dépendance fonctionnelle entre les connaissances intrinsèques et les connaissances discriminantes, c'est-à-dire entre le module de description intrinsèque par prototypes et le module de discrimination focalisée. Plus précisément, il permet de faire la décomposition du problème en assignant les formes du niveau de modélisation intrinsèque à chacun des sous-problèmes du niveau de modélisation discriminante auxquels elles correspondent. Cette affectation s'effectue à partir de la description de la forme, en la comparant à chacun des prototypes des classes définis par les connaissances intrinsèques.

Le deuxième type de lien est utilisé pour organiser les connaissances de nature discriminante. En effet, nous avons montré en présentant ces dernières, qu'elles étaient multiples et qu'elles n'avaient pas toutes le même pouvoir de discrimination. La conséquence directe est que si plusieurs éléments de connaissances sont nécessaires pour parvenir à discriminer les classes d'un sous-problème (ce qui sera presque systématique hormis dans les cas très simples), alors il faut structurer celles-ci de manière à prendre en compte leur pertinence. Cette structuration se présente donc naturellement sous la forme d'une hiérarchisation des connaissances.

3.1.4 Synthèse

Le tableau 3.1 récapitule la nature, la fonction et les propriétés des connaissances, ainsi que les liens qui les unissent.

L'intégration de ces connaissances dans la modélisation de notre système est alors illustrée sur la figure 3.5.

Au module de description intrinsèque par prototypes est associé le *niveau de modélisation intrinsèque*. Il représente chaque classe ω_i par un modèle intrinsèque $MI(\omega_i)$ décrivant sa structure en sous-classes.

Le module de discrimination focalisée est quant à lui associé au *niveau de modélisation discriminante*. Il élabore un modèle discriminant $MD(\omega_{i+}, \omega_{i-})$ par sous-problème, c'est-à-dire par classe ω_i dans notre cas. Celui-ci a pour objectif de différencier les formes du sous-problème appartenant à ω_i (notées ω_{i+}) de l'ensemble ω_{i-} des autres formes ayant des propriétés intrinsèques proches.

Pour que les connaissances puissent remplir leur fonction et satisfaire nos objectifs, il faut à présent déterminer quels mécanismes d'extraction et quel formalisme de représentation utiliser. C'est l'objet de la suite du chapitre. Pour pouvoir faciliter la

Nature	**Rôle dans la modélisation**	**Propriétés**
Intrinsèques	- description des classes - sert de base pour la décomposition - sert de base pour le rejet de distance	- stables - représentatives de la classe
		- interprétables - robustes (formalisme) - compactes
Discriminantes	- discrimination des classes d'un sous-problème	- associées à un sous-espace - pouvoir de discrimination (contexte) - multiples - frontières séparatrices
Structuration	- décomposition - hiérarchisation des connaissances discriminantes	

TAB. 3.1 – *Les connaissances utilisées dans notre approche et leur structuration.*

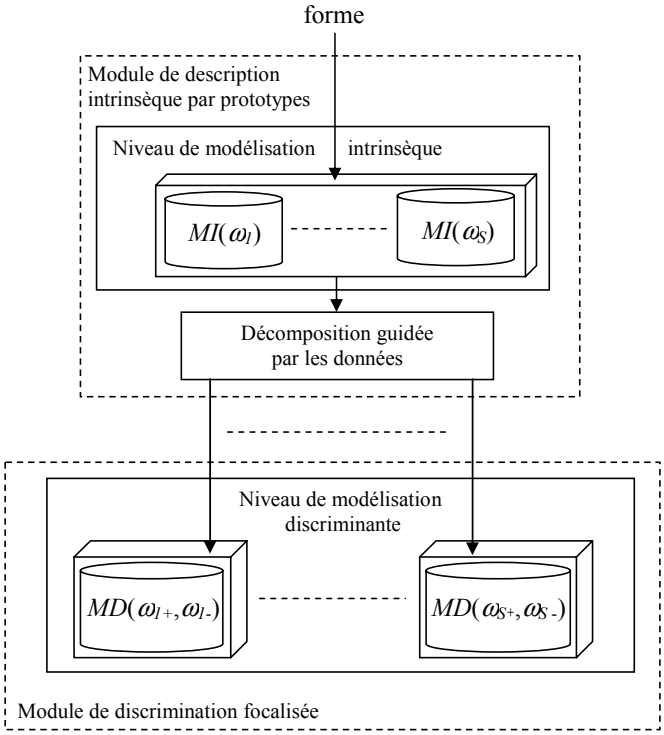

FIG. 3.5 – *Structuration des connaissances dans la modélisation du système Mélidis.*

lecture et la compréhension de celle-ci nous reportons sur la figure 3.6 un résumé des moyens utilisés pour l'apprentissage de la modélisation. Le lecteur peut s'y référer à tout moment pour avoir une vision synthétique du processus.

Niveau de modélisation intrinsèque :

- *Étape 1 : Extraction des modèles intrinsèques aux classes*
  - But : extraire les sous-classes des classes du problème et les modéliser par des prototypes flous
  - Initialisation : création d'une base d'apprentissage B^i_{app} par classe ω_i contenant uniquement les exemples appartenant à cette classe
  - Processus : utilisation des *C-moyennes possibilistes* sur ces bases ; le nombre de sous-classes est défini par l'*indice de Xie et Beni* et les fonctions d'appartenance sont des fonctions à base radiale
  - Résultat : les modèles $MI(\omega_i)$ constitués d'un ensemble de sous-ensembles flous décrivant de façon intrinsèque les sous-classes des classes ω_i
- *Étape 2 : Décomposition guidée par les données du problème en sous-problèmes*
  - But : création d'une sous-base d'apprentissage par classe ω_i regroupant les exemples ayant des propriétés intrinsèques similaires à celles des sous-classes
  - Processus : utilisation d'un seuillage relatif sur l'appartenance des exemples aux modèles $MI(\omega_i)$. Cette appartenance est déterminée par la disjonction (t-conorme max) sur l'appartenance à chacune des sous-classes du modèle. Remarque : les exemples peuvent être dupliqués dans plusieurs sous-bases
  - Résultat : une sous-base d'apprentissage B^i_{app2} par classe ω_i dans laquelle les exemples appartenant à ω_i sont étiquetés par le label ω_{i+}, les autres par ω_{i-}

Niveau de modélisation discriminante :

- But : extraire les connaissances permettant de discriminer ω_{i+} et ω_{i-} dans chaque sous-base B^i_{app2}
- Processus : création d'un arbre de décision flou à partir de chaque base B^i_{app2}
- Particularités des arbres :
  - Modalités floues : extraction dynamique localement à chaque nœud par les *C-moyennes floues*
  - Mode de partitionnement : partitionnements binaires dans des sous-espaces F_k à deux dimensions. Les modalités floues sont nommées F_{k1} et F_{k2}. Propagation des individus le long des branches par une α-coupe
  - Choix de l'espace : utilisation d'un algorithme génétique couplé à la mesure d'entropie étoile
  - Construction des feuilles : critère d'arrêt reposant sur le nombre minimum d'individus nécessaire à un nœud pour continuer le partitionnement et le gain d'information ; calcul des degrés de représentativité d_{fi+} et d_{fi-} des classes ω_{i+} et ω_{i-} par la *probabilité conditionnelle floue* d'obtenir ces classes au niveau de la feuille

FIG. 3.6 – *Résumé du processus d'apprentissage de la modélisation du système Mélidis.*

3.2 Modélisation des connaissances par extraction automatique de sous-ensembles flous

Nous avons choisi d'utiliser les sous-ensembles flous comme formalisme de représentation des connaissances de nature intrinsèque et discriminante. Les spécificités de ces derniers permettent en effet de leur conférer une bonne partie des propriétés que nous avons énoncées, notamment celles qui leurs sont communes. Ainsi, comme montré dans le chapitre 2, les sous-ensembles flous définis sur des attributs numériques permettent de décrire un ensemble de valeurs de façon synthétique. Grâce à cette granularité ainsi qu'à la notion d'appartenance partielle à l'ensemble, les connaissances ainsi représentées sont plus robustes et permettent d'absorber efficacement la variabilité ainsi que les imprécisions relatives aux concepts qu'elles représentent. Enfin, ce formalisme est assez intuitif et offre une meilleure lisibilité que les modes de représentation courants.

Les autres propriétés, comme le lien avec la structure des données dans l'espace de représentation ainsi que celles qui sont spécifiques à chacune des natures de connaissances ne peuvent être obtenues que par le processus permettant de définir la forme et la position des sous-ensembles flous. Il existe de nombreuses façons pour obtenir ces derniers. Bien souvent, lorsqu'une expertise existe ou que l'on souhaite avoir la plus grande interprétabilité (du point de vue de l'utilisateur), les sous-ensembles flous sont définis *a priori*. Cependant, dans notre contexte d'étude, cela va à l'encontre de l'objectif de généricité. Il faut donc utiliser un procédé d'extraction automatique. Là encore différentes techniques sont possibles, qu'il s'agisse d'un partitionnement régulier de l'espace par un système de grille ou d'une détermination automatique par algorithmes génétiques, par des techniques reposant sur la morphologie mathématique, etc. Pour notre approche, nous avons repris les mêmes principes que ceux utilisés dans *ResifCar* [16]. Ils reposent sur l'utilisation d'algorithmes de classification floue non-supervisée qui permettent de conserver un lien explicite avec les données et donc d'aller dans le sens de l'interprétabilité. De plus, il existe un certain nombre d'algorithmes, chacun ayant des propriétés spécifiques, permettant ainsi d'adapter l'extraction des sous-ensembles flous en fonction du but recherché. Ce dernier point est important pour que les connaissances intrinsèques et discriminantes possèdent le comportement souhaité.

Dans ce paragraphe, nous faisons quelques brefs rappels sur les principes de classification non supervisée avant d'introduire leurs extensions « floues » et notamment l'algorithme des *C-moyennes floues* [22]. C'est en effet ce dernier ainsi que ses versions dérivées qui nous permettront d'extraire et de représenter automatiquement les connaissances qui nous intéressent.

3.2.1 Intérêts de la classification non supervisée

L'objectif de la classification non supervisée est d'extraire une structure dans une base de données non étiquetée. Pour cela, des regroupements (ou classes) sont recherchés. Chacun d'eux correspond à un sous-ensemble d'échantillons ayant des propriétés communes.

On distingue essentiellement deux grands types d'approches pour la classification non supervisée : les approches hiérarchiques et les approches adaptatives. Les premières utilisent les données initiales soit pour les regrouper au fur et à mesure (approches agglomératives), soit au contraire pour effectuer des divisions successives (approches divisives). Les secondes visent à optimiser une partition initiale de manière itérative en se basant sur une fonction objectif. Les principes généraux de ces approches sont décrits dans l'annexe B.

Les approches ordinaires de classification non-supervisée ont un inconvénient majeur : les données ne peuvent appartenir qu'à un seul regroupement. Cette vision est en fait assez éloignée de la réalité. En effet, dans la plupart des cas, les regroupements recherchés ne possèdent pas de frontières bien déterminées. Il arrive aussi qu'ils se chevauchent. Il est alors plus correct de dire qu'un individu situé près d'une frontière peut appartenir aux différents groupes concernés plutôt que de l'affecter « arbitrairement » à un seul regroupement. Ce comportement est gênant non seulement pour la classification de nouveaux individus mais aussi pour la détermination de la partition si celle-ci s'effectue de manière itérative. En effet si les données sont variables, ou bruitées (c'est-à-dire imprécises), une erreur de classement lors d'une itération peut avoir une répercution importante sur les itérations suivantes et produire une classification peu pertinente.

L'introduction de la logique floue dans les algorithmes de classification non supervisée constitue alors une alternative séduisante comme nous allons le voir ci-dessous.

3.2.2 Les C-moyennes floues

L'algorithme des *C-moyennes floues* (CMF), introduit par Dunn [53] et défini par Bezdek [22] est une généralisation de l'algorithme des *C-moyennes* décrit dans l'annexe B.2. Il se base sur une fonction objectif pour optimiser une partition initiale des données. Cette partition est caractérisée par un ensemble de prototypes correspondant à des centroïdes.

Le principe de fonctionnement de l'algorithme ainsi que ses avantages et inconvénients sont décrits ci-dessous.

3.2.2.1 Principes

Soit $\mathcal{E} = \{e_j | j = 1, \ldots, N\}$ l'ensemble des données définies dans $\mathcal{R}^n$ que l'on souhaite partitionner en C regroupements $C_1, \ldots, C_c, \ldots, C_C$. Soit $\mathcal{P} = \{P_1, \ldots, P_C\}$ l'ensemble des centroïdes caractérisant ces regroupements et $d(e_j, P_c)$ une distance de la forme générale des distances de Mahalanobis (cf. annexe A) entre une donnée e_j et un centroïde P_c (l'algorithme standard des *C-moyennes floues* est défini avec la distance Euclidienne).

Chaque individu e_j de $\mathcal{E}$ est caractérisé par un degré d'appartenance (comme défini au paragraphe 2.1.1.1) μ_{jc} à chacun des regroupements C_c.

Le but de l'algorithme consiste alors à minimiser la fonction objectif suivante qui

représente la distance moyenne intra-groupe « floue » au sens des moindres carrés :

$$J_{P,U,\mathcal{E}} = \sum_{c=1}^{C}\sum_{j=1}^{N}(\mu_{jc})^m d^2(e_j, P_c) \tag{3.1}$$

Dans cette fonction, U représente la matrice $C \times N$ de partitionnement flou (par opposition à un partitionnement net) au sens de Ruspini [149]. Elle est constituée des éléments μ_{jc} tels que :

$$\forall j \in \{1,\ldots,N\},\ c \in \{1,\ldots,C\},\ \mu_{jc} \in [0,1]\ ; \quad \forall j,\ \sum_{c=1}^{C}\mu_{jc} = 1\ ; \quad \forall c,\ 0 < \sum_{j=1}^{N}\mu_{jc} < N \tag{3.2}$$

La contrainte imposée sur les degrés d'appartenance (leur somme vaut 1) empêche d'obtenir $\mu_{jc} = 0,\ \forall j, c$ comme solution triviale pour la minimisation de $J_{P,U,\mathcal{E}}$. Enfin, le paramètre m défini dans $[1, \infty[$ introduit le degré de flou de la partition. Il permet de jouer sur le degré de chevauchement autorisé entre les regroupements. Quand $m \to 1$, la partition tend vers une partition nette (c'est-à-dire non floue) ; au contraire, quand $m \to \infty$, la partition devient de plus en plus floue. Lorsque $m = 2$ (choix qui est très souvent fait en pratique), la fonction objectif est équivalente à celle introduite initialement dans [53].

Pour que la fonction objectif puisse être minimisée, le calcul des centroïdes et des fonctions d'appartenance s'effectue de la manière suivante :

$$P_c = \frac{\sum_{j=1}^{N}(\mu_{jc})^m e_j}{\sum_{j=1}^{N}(\mu_{jc})^m}, \tag{3.3}$$

$$\left.\begin{array}{ll} \mu_{jc} = \dfrac{1}{\sum_{l=1}^{C}\left(\frac{d^2(e_j,P_c)}{d^2(e_j,P_l)}\right)^{\frac{1}{m-1}}} & \text{si } J_j = \emptyset \\ \left.\begin{array}{ll} \mu_{jc} = 0 & \text{si } c \notin J_j \\ \sum_{c\in J_j}\mu_{jc} = 1 & \text{si } c \in J_j \end{array}\right\} & \text{si } J_j \neq \emptyset \end{array}\right\} \tag{3.4}$$

avec $J_j = \{c | 1 \leq c \leq C,\ d^2(e_j, P_c) = 0\}$.

L'algorithme des *C-moyennes floues* se déroule finalement selon les étapes suivantes :

- *initialisation :* déterminer le nombre de regroupements C, le degré de flou m, la distance d. Initialiser la matrice U.
- *calcul des centres :* calculer la position des C centroïdes P_c (équation (3.3)).
- *partitionnement :* mémoriser U dans U_{svgd} et re-calculer U par les équations 3.4.
- *arrêt :* recommencer à l'étape *calcul des centres* tant que $\|U - U_{svgd}\| \geq \epsilon$.

3.2.2.2 Avantages et inconvénients de l'algorithme

Le principal avantage de l'algorithme par rapport aux algorithmes plus classiques provient de l'introduction des degrés d'appartenance μ_{jc}. Grâce à eux, le processus

d'optimisation itératif est rendu beaucoup plus robuste, notamment en permettant de prendre en compte les recouvrements entre les regroupements. Il permet ainsi d'obtenir des partitions plus pertinentes et plus proches de la réalité. En outre, ces degrés permettent de prendre des décisions nuancées pour l'assignation d'une forme à un regroupement, ce qui s'avère très intéressant pour toute forme de classification.

Parmi les autres avantages de l'algorithme, on peut noter que sa complexité algorithmique est relativement réduite par rapport à d'autres algorithmes de classification non supervisée. Cela le rend plus facilement exploitable pour traiter des problèmes de taille importante (avec beaucoup de données).

Outre ces avantages très généraux, l'algorithme des CMF est aussi particulièrement adapté au cadre de notre étude. Il permet en effet d'extraire automatiquement des sous-ensembles flous décrivant d'une façon robuste et synthétique la structure des données. Ces sous-ensembles flous peuvent être définis à partir de l'équation 3.4 donnant directement la fonction d'appartenance. Ils ne dépendent alors que de la connaissance des centroïdes P, du degré de flou m et de la métrique d. Les paramètres nécessaires à leur représentation sont donc peu nombreux, ce qui les rend peu coûteux à utiliser et ce qui va dans donc le sens de notre objectif de *compacité*.

Malgré tout, l'algorithme possède aussi quelques inconvénients. On peut citer par exemple le problème de la sensibilité à l'initialisation (différentes initialisations peuvent aboutir à différentes partitions), la nécessité d'imposer le nombre de regroupements C *a priori* ou encore le manque de flexibilité sur la forme des regroupements qu'il peut détecter. La plupart de ces difficultés peuvent cependant être contournées. Un exemple relatif à la forme des regroupements est donné dans le paragraphe suivant.

En dehors de ces inconvénients, l'algorithme possède aussi certaines spécificités qui peuvent le rendre inadapté pour certains usages. Nous reportons celles-ci un peu plus loin parce qu'elles ont un impact important dans le cadre de notre étude.

3.2.3 Les algorithmes dérivés des C-moyennes floues

Les CMF ont été très largement utilisées dans de nombreux domaines. Différentes adaptations ont aussi été faites, notamment pour pouvoir traiter les problèmes dans lesquels les regroupements ont des formes variées. Ces adaptations se fondent essentiellement sur une modification de la métrique utilisée [98] et de la forme des prototypes.

Pour pouvoir effectuer des regroupements avec des formes variées, une solution simple consiste à modifier la distance utilisée. Rappelons que dans le cas des CMF, la distance utilisée est la distance Euclidienne. Les regroupements trouvés possèdent donc une forme hypersphérique.

Une variante proposée par Gustafson et Kessel [70] consiste à adapter cette distance de manière à pouvoir obtenir des regroupements de formes hyperellispoïdales quelconques. Pour cela, la distance Euclidienne est remplacée par une distance générale

de Mahalanobis (cf. annexe A) et en définissant une par regroupement C_c :

$$d_c^2(e_j, P_c) = [\det(Cov_c)]^{1/n} \, (e_j - P_c)^t Cov_c^{-1} (e_j - P_c) \tag{3.5}$$

avec n la dimension de l'espace des données et Cov_c la matrice de covariance floue associée au regroupement c. Celle-ci est calculée à chaque pas de l'itération par :

$$Cov_c = \frac{1}{\sum_{j=1}^{N} \mu_{jc}^m} \sum_{j=1}^{N} \mu_{jc}^m (e_j - P_c)(e_j - P_c)^t \tag{3.6}$$

Dans cet algorithme, les prototypes décrivant les regroupements ne sont plus uniquement déterminés par les centroïdes P_c mais aussi par la matrice de covariance Cov_c associée.

D'une façon similaire, Gath et Geva [61] partent de l'hypothèse selon laquelle les données ont une distribution Gaussienne dans chaque regroupement. La mesure de dissimilarité entre une donnée e_j et un regroupement C_c est alors considérée comme étant inversement proportionnelle à la probabilité *a posteriori* que e_j appartienne à C_c. Pour cela, la distance utilisée est adaptée de la manière suivante :

$$d_c^2(e_j, P_c) = \frac{[\det(Cov_c)]^{1/2}}{p_c} \exp\left(\frac{(e_j - P_c) Cov_c^{-1} (e_j - P_c)^t}{2}\right)$$

où p_c est la probabilité *a priori* qu'une donnée appartienne à C_c. Celle-ci est estimée à chaque pas de l'itération par : $p_c = \frac{\sum_{j=1}^{N} \mu_{jc}^m}{N}$. Cette mesure de dissimilarité permet d'obtenir des regroupements hyperellipsoïdaux de formes et de densités variées. Dans ce contexte, les prototypes sont définis à la fois par les centroïdes P_c, les matrices de covariance Cov_c et la probabilité p_c.

D'autres types de modifications ont été apportées à l'algorithme initial des CMF, notamment pour modifier son comportement en agissant sur la fonction objectif. C'est un point important que nous exploitons ici pour pouvoir extraire des sous-ensembles flous correspondant aux différentes nature de connaissances que nous utilisons dans notre système. Le paragraphe suivant présente l'adaptation utilisée pour extraire et représenter les connaissances intrinsèques.

3.3 Niveau intrinsèque : modélisation des connaissances et décomposition

Le niveau de modélisation intrinsèque a pour fonction de décrire chacune des classes du problème en s'appuyant sur leurs propriétés intrinsèques. Cette modélisation doit notamment tirer parti du caractère multimodal des classes afin de faire ressortir une structuration horizontale en sous-classes. Les connaissances utilisées pour décrire cette structure doivent donc représenter les caractères les plus représentatifs et les plus stables

des classes. Ces propriétés sont essentielles pour l'interprétabilité de la modélisation mais aussi pour le bon fonctionnement du système puisque le mécanisme de décomposition ainsi que celui de rejet de distance s'appuient directement sur cette description.

Nous avons vu précédemment que les algorithmes de classification floue non supervisée étaient des outils particulièrement intéressants dans le cadre de notre problématique pour extraire et modéliser les connaissances. Cependant, pour une modélisation intrinsèque les CMF ainsi que les extensions présentées dans le paragraphe 3.2 sont en fait assez mal adaptées. Nous montrons ci-dessous pourquoi et proposons comme alternative de nous appuyer sur un algorithme de type possibiliste [98, 100, 99, 101] pour extraire les modèles intrinsèques aux classes. Celui-ci est semblable à celui qui est utilisé dans *ResifCar* pour modéliser les primitives des tracés [16].

3.3.1 Inconvénients des C-moyennes floues pour l'extraction de connaissances intrinsèques

Dans les CMF, la contrainte imposée aux degrés d'appartenance des individus (cf. équation (3.4)), fait que les prototypes et les regroupements sont établis les uns par rapport aux autres. La conséquence directe est la forme particulière des fonctions d'appartenance déduites de l'algorithme. Elles sont définies les unes par rapport aux autres (cf. équation 3.4) et traduisent la notion de « partage » des individus entre les regroupements.

La figure 3.7 montre l'impact sur le comportement des fonctions d'appartenance lorsque celles-ci sont déterminées en une dimension (a) ou bien en deux dimensions (matérialisées par leurs lignes de niveaux) (b).

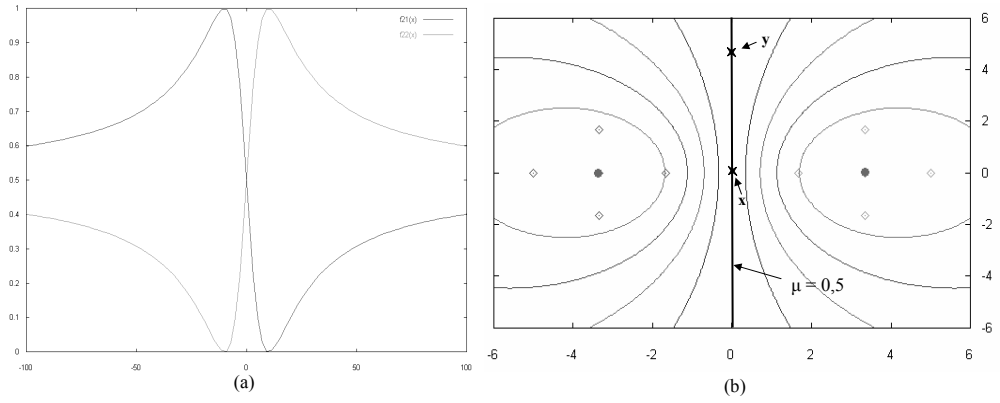

FIG. 3.7 – *Forme et comportement des fonctions d'appartenance issues des C-moyennes floues lorsque deux regroupements sont recherchés : en une dimension (a) ; en deux dimensions après projection des lignes de niveaux ($\mu = 0,9$ à $\mu = 0,5$). Les degrés d'appartenance tendent vers* $0,5$ *lorsqu'on est à égale distance des centroïdes, que ce soit près de ceux-ci (individu 'x') ou loin (individu 'y') (b).*

On se rend compte en observant la forme des fonctions loin des centroïdes (a),

qu'avec cette représentation il est impossible de distinguer les individus se situant près de la frontière tout en étant proches des centroïdes des individus se situant près de la frontière mais loin de ceux-ci. Ainsi, sur (b), les deux observations notées x et y sont toutes les deux à une distance égale des deux centroïdes ($d(x, P_1) = d(x, P_2)$ et $d(y, P_1) = d(y, P_2)$) et ont donc le même degré d'appartenance à chacun d'eux ($\mu_{x1} = \mu_{x2} = \mu_{y1} = \mu_{y2} = 0.5$). Pourtant, l'individu y est clairement plus loin des centroïdes que x ($d(y, P_1) > d(x, P_1)$).

Même si cette particularité tend à s'estomper quand le nombre de regroupements recherchés (C) augmente puisque les degrés tendent vers $1/C$, celle-ci est particulièrement gênante et ne correspond pas du tout à une caractérisation intrinsèque des données. En effet, ce que l'on désire, c'est que les individus éloignés d'un centroïde possèdent un degré d'appartenance à celui-ci qui soit faible. C'est la notion de « typicalité » telle qu'elle est utilisée dans [100] et qui s'oppose à celle de « partage » des CMF.

Une autre conséquence de cette notion de partage est que l'algorithme reste assez sensible aux bruit. Comme une donnée appartient nécessairement à un ou plusieurs regroupements, la présence de données non représentatives influe sur la position et la forme des sous-ensembles flous. La figure 3.8 illustre ce point : la donnée notée x provoque une « rotation » des sous-ensembles flous par rapport à leur axe naturel. Elle induit aussi un « décalage » des centroïdes P_1 et P_2 qui ne représentent plus la valeur la plus typique des données associées à chacun des regroupements.

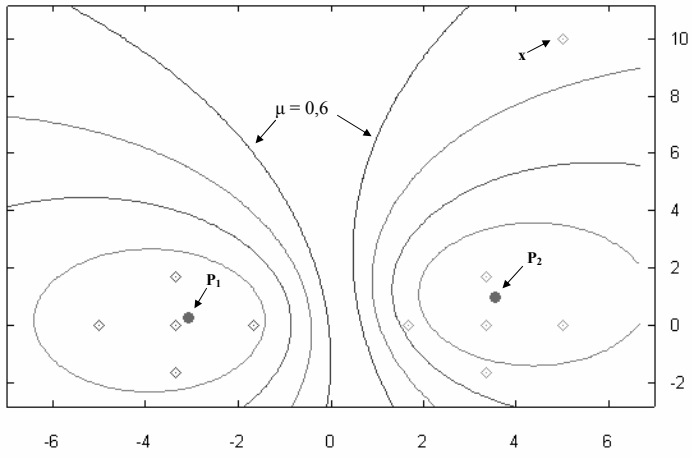

FIG. 3.8 – *Problèmes liés à la définition relative des fonction d'appartenance dans les C-moyennes floues : la donnée 'x' est du bruit mais elle déforme et déplace les sous-ensembles flous.*

Pour toutes ces raisons, l'algorithme des CMF ne satisfait pas nos besoins pour l'extraction et la modélisation de connaissances intrinsèques aux classes. Même s'il peut

être utilisé pour extraire une structure de sous-classes, il est incapable de déterminer et de représenter les valeurs les plus typiques. Afin de remédier à ces différents problèmes, il faut modifier le mode de fonctionnement de l'algorithme. L'algorithme des *C-moyennes possibilistes* correspond à une telle évolution.

3.3.2 Les C-moyennes possibilistes

Le principe de l'algorithme des *C-moyennes possibilistes* (CMP) [98, 100, 99, 101] est de caractériser les regroupements non plus de manière relative, les uns par rapport aux autres, mais au contraire de façon « absolue », ce qui correspond exactement à la définition d'un caractère intrinsèque.

Pour cela, l'idée consiste à relâcher la contrainte sur la sommation à 1 des degrés d'appartenance de l'algorithme des CMF (cf. équation 3.2). Mais si cette contrainte est simplement supprimée, la minimisation de la fonction objectif initiale (équation (3.1)) a comme solution triviale l'attribution de degrés d'appartenance nuls pour chaque donnée. Pour éviter ce problème, la fonction objectif est modifiée comme suit :

$$J_{P,U,\mathcal{E}} = \sum_{c=1}^{C}\sum_{j=1}^{N}(\mu_{jc})^m d^2(e_j, P_c) + \sum_{c=1}^{C}\eta_c \sum_{j=1}^{N}(1-\mu_{jc})^m \tag{3.7}$$

La première partie de la fonction est identique à celle des CMF. Elle vise toujours à minimiser la distance au sein de chacun des regroupements. Le second terme vise quant à lui à maximiser les degrés d'appartenance pour qu'il ne soient pas tous nuls.

Dans cette fonction, la matrice U représente toujours une matrice de partition floue $C \times N$ mais plus au sens de Ruspini [149]. Elle est constituée des degrés d'appartenance μ_{jc} des N individus au C regroupements. Ceux-ci satisfont les contraintes suivantes :

$$\forall j \in \{1,\ldots,N\},\ c \in \{1,\ldots,C\},\ \mu_{jc} \in [0,1]\,; \quad \forall c,\ \max_c \mu_{jc} > 0\,; \quad \forall c,\ 0 < \sum_{j=1}^{N}\mu_{jc} \leq N \tag{3.8}$$

La contrainte sur la sommation à 1 des degrés d'appartenance dans (3.2) a donc été remplacée par une simple contrainte sur l'existence, pour chaque centroïde, d'au moins une donnée y appartenant. Pour que ces contraintes soit satisfaites et que $J_{P,U,\mathcal{E}}$ puisse être minimisée, les centroïdes sont calculés de la même façon qu'avant (équation 3.3) et les degrés d'appartenance sont définis par :

$$\mu_{jc} = \frac{1}{1 + \left(\frac{d^2(e_j,P_c)}{\eta_c}\right)^{\frac{1}{m-1}}} \tag{3.9}$$

Ces derniers permettent de mesurer de manière absolue l'adéquation (ou encore la correspondance) entre un individu et un prototype puisqu'ils ne dépendent que de la distance au centroïde. Ils correspondent donc aux concepts associés à la notion de typicalité.

Le paramètre η_c qui apparaît à la fois dans la fonction objectif (3.7) et dans le calcul des degrés d'appartenance permet, pour chaque centroïde, de déterminer sa zone

d'influence. Plus précisément, il définit à quelle distance du centroïde P_c une donnée aura un degré d'appartenance de $0,5$. Il permet aussi de pondérer la deuxième partie de la fonction objectif par rapport à la première. Ainsi, pour que la minimisation des distances au sein des regroupements soit considérée comme un objectif aussi important que la maximisation des degrés d'appartenance, il faut que η_c soit de l'ordre de $d^2(e_j, P_c)$. Pour déterminer sa valeur de façon automatique, Krishnapuram [98, 100] propose de l'évaluer en lui affectant une valeur proportionnelle à la distance moyenne au sein du regroupement :

$$\eta_c = K \frac{\sum_{j=1}^{N} \mu_{jc}^m \, d^2(e_j, P_c)}{\sum_{j=1}^{N} \mu_{jc}^m} \tag{3.10}$$

Ici, K est souvent choisi égal à 1. Une variante pour évaluer η_c consiste à n'utiliser que les individus représentatifs :

$$\eta_c = \frac{\sum_{e^j \in (\Pi_c)_\alpha} d^2(e_j, P_c)}{|(\Pi_c)_\alpha|} \tag{3.11}$$

où $(\Pi_c)_\alpha$ représente l'α-coupe associée au sous-ensemble flous définit par μ_{jc} et $|(\Pi_c)_\alpha|$ est le nombre d'individus de cette α-coupe. Généralement, α prend une valeur entre $0,1$ et $0,4$.

Il est théoriquement possible de réévaluer η_c à chaque pas de l'algorithme. Cependant, il existe des risques d'instabilité. La meilleure approche consiste alors à approximer η_c à partir d'une partition initiale en utilisant (3.10) puis à les ré-estimer de manière plus précise après la convergence de l'algorithme avec (3.11). L'algorithme est ensuite relancé une seconde fois avec ces nouvelles valeurs, ce qui a pour effet de limiter encore plus l'impact du bruit sur la position et la forme des regroupements. Cette deuxième étape converge généralement très vite et n'est nécessaire que si l'on souhaite obtenir la véritable forme des fonctions d'appartenance (dans certains cadres de travail, les centroïdes peuvent suffire).

De par son mode de fonctionnement, l'algorithme est très sensible à l'initialisation. D'autres algorithmes comme celui de Gath et Geva [61] (cf. paragraphe 3.2.3) sont donc très souvent utilisés dans un premier temps pour obtenir la partition initiale. L'algorithme se déroule alors selon les deux phases suivantes :

- **Phase 1 :**
  - *Initialisation :* déterminer le nombre de regroupements C, le degré de flou m, la distance d. Initialiser la matrice U à partir du résultat d'un algorithme d'initialisation comme celui de Gath et Geva et évaluer les η_c à partir de (3.10) ;
  - *Mise à jour des centres :* calculer la position des C centroïdes P_c (équation (3.3)).
  - *Mise à jour du partitionnement :* mémoriser U dans U_{svgd} et re-calculer U par les équations (3.9).
  - *Arrêt :* recommencer à la *mise à jour des centres* tant que $\|U - U_{svgd}\| \geq \epsilon$.

- **Phase 2 (optionnelle) :**
  - *Initialisation :* recalculer les η_c avec (3.11) ;
  - *Mise à jour des centres :* calculer la position des C centroïdes P_c (équation (3.3)).
  - *Mise à jour du partitionnement :* mémoriser U dans U_{svgd} et re-calculer U par l'équation 3.9.
  - *Arrêt :* recommencer la *mise à jour des centres* tant que $\|U - U_{svgd}\| \geq \epsilon$.

3.3.3 Propriétés de l'algorithme pour une modélisation intrinsèque

Avec les modifications apportées aux CMF, l'algorithme des CMP permet d'obtenir des fonctions d'appartenance aux différents regroupements qui ne sont plus dépendantes de leurs interactions mais uniquement de la distance d'une donnée aux centres. Cela se traduit au niveau de l'algorithme, lors des itérations par une mobilité plus réduite des prototypes qui n'ont qu'un champ d'action limité par η_c. C'est une des raisons principales qui rend la phase d'initialisation de l'algorithme très importante. En revanche, cela permet de distinguer clairement les individus proches des centroïdes de ceux qui en sont éloignés. La forme des fonctions d'appartenance est illustrée sur la figure 3.9.

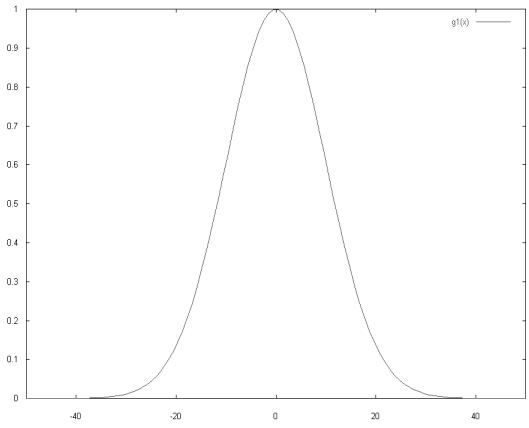

FIG. 3.9 – *Forme d'une fonction d'appartenance issue des C-moyennes possibilistes en une dimension.*

En reprenant le même problème que celui de la figure 3.7, le résultat de l'algorithme possibiliste présenté sur la figure 3.10 montre bien que cette fois, l'individu y possède un degré d'appartenance très nettement inférieur à celui de x.

De plus, comme annoncé, l'algorithme est beaucoup moins sensible à la présence de bruit dans les données, ce qui est illustré sur la figure 3.11 qui est à comparer avec la figure 3.8.

Grâce à ces propriétés, l'algorithme des CMP permet d'extraire automatiquement des sous-ensembles flous caractérisant de manière intrinsèque l'organisation spatiale

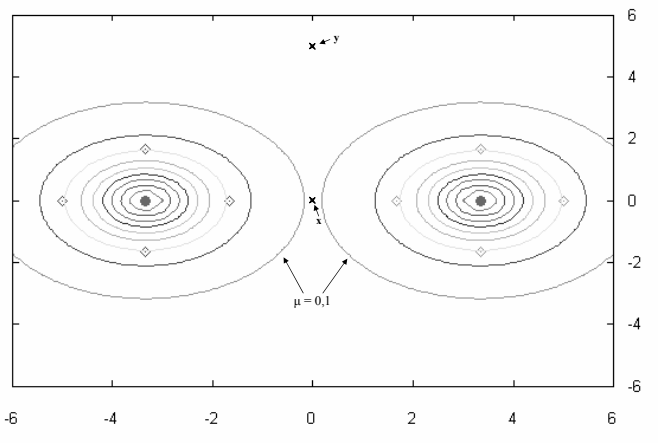

FIG. 3.10 – *Caractérisation intrinsèque par les C-moyennes possibilistes.*

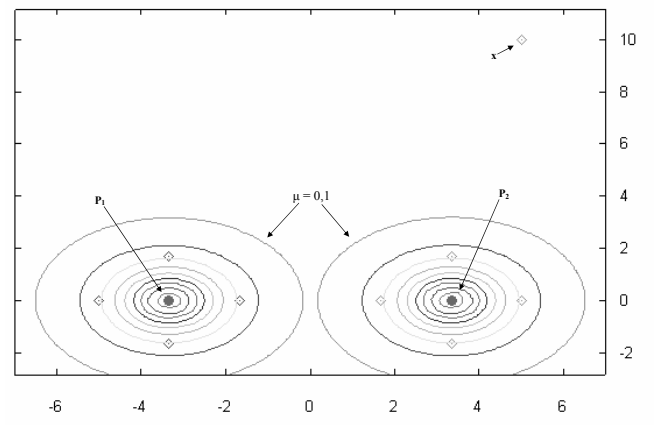

FIG. 3.11 – *Immunité au bruit des prototypes issus des C-moyennes possibilistes.*

d'un ensemble de données, notamment en s'appuyant sur les valeurs les plus typiques. Nous nous appuyons donc sur celui-ci pour faire la description intrinsèque des classes.

3.3.4 Extraction des modèles intrinsèques aux classes

Pour chaque classe ω_i, nous voulons obtenir un modèle intrinsèque $MI(\omega_i)$ (cf. figure 3.5). Pour faire cette modélisation et respecter le caractère intrinsèque de la classe, il ne faut considérer que les échantillons appartenant à celle-ci.

Soit B_{app} la base d'apprentissage telle que nous l'avons déjà définie :

$$B_{app} = \{e_j | j = 1, \ldots, N\},$$

$$\forall e_j \in B_{app},\ e_j = [e_j^1, \ldots, e_j^k, \ldots, e_j^n, e_j^{(n+1)}],$$

avec N le nombre d'individus de la base, n le nombre de caractéristiques de l'espace de représentation et $e_j^{(n+1)} \in S$ ($S = \{\omega_1, \ldots, \omega_i, \ldots, \omega_{\mathcal{S}}\}$) l'étiquette de e_j. B_{app} est divisée en autant de sous-bases B_{app}^i qu'il y a de classes de la façon suivante :

$$\forall i = 1, \ldots, \mathcal{S} \quad B_{app}^i = \{e_j \in B_{app} | e_j^{(n+1)} = \omega_i\}$$

Après avoir configuré l'algorithme des CMP comme indiqué dans le paragraphe suivant, il est utilisé *indépendamment sur chaque sous-base* pour obtenir les modèles intrinsèques correspondant à chaque classe (cf. paragraphe 3.3.4.2).

3.3.4.1 Configuration des C-moyennes possibilistes pour la caractérisation des classes

Afin d'exploiter au mieux l'algorithme des CMP dans notre contexte d'utilisation, certains paramètres doivent être fixés.

Le premier concerne la forme des regroupements ou plus précisément la forme des fonctions d'appartenance décrivant ces regroupements. Celle-ci dépend à la fois de la distance d et des valeurs η_c. Nous avons vu dans le paragraphe 3.3.2 que dans la version standard de l'algorithme, la distance est de la forme générale des distances de type Mahalanobis et que η_c pouvait être déterminé par l'équation (3.11). Mais l'estimation de ce paramètre est particulièrement sensible. Une possibilité pour s'en affranchir consiste à déterminer directement la forme des fonctions d'appartenance désirée [99], ce qui permet en même temps d'intégrer directement la distance. Par exemple, il est possible d'utiliser des fonctions exponentielles comme :

$$\mu_{jc} = \exp\left(-\frac{1}{2}(e_j - P_c)^t Cov_c^{-1}(e_j - P_c)\right) \tag{3.12}$$

où Cov_c est la matrice de covariance des individus appartenant au regroupement C_c. Dans ce cas, Cov_c est estimée une première fois lors de l'initialisation (phase 1) à partir de la matrice de covariance floue (cf. équation (3.6)) résultant de l'algorithme d'initialisation utilisé. Elle est ensuite ré-estimée lors de l'initialisation de la phase 2, mais cette fois-ci de manière classique (non floue) à partir des données appartenant au regroupement C_c uniquement (matrice de partition non floue) et après avoir supprimé les données ayant un degré d'appartenance faible à chacun des regroupements (élimination du bruit).

Pour la modélisation de sous-classes, les fonctions exponentielles tendent vers 0 assez rapidement, limitant la portée des prototypes et rendant la modélisation assez stricte. Dans notre cas de figure, comme les modèles intrinsèques aux classes sont utilisés pour décomposer le problème et établir le lien entre le module de description intrinsèque par prototypes et le module de discrimination focalisée, il est important que la description soit robuste et pas trop restrictive. Nous nous sommes donc tourné vers des fonctions d'appartenance à base radiale (forme de « cloche ») plus souples [16] (cf. figure 3.12) :

$$\mu_{jc} = \frac{1}{1 + (e_j - P_c)^t Cov_c^{-1}(e_j - P_c)} \tag{3.13}$$

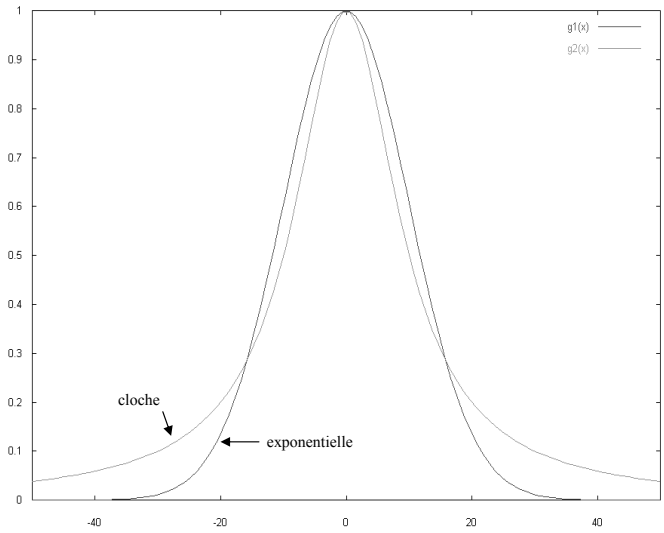

FIG. 3.12 – *Différence entre les fonctions d'appartenance exponentielles et les fonctions de type « cloche » utilisées.*

Ce problème peut aussi être adressé par le biais du paramètre m, comme expliqué dans [101].

Le deuxième point important pour le bon fonctionnement de l'algorithme concerne le nombre de regroupements C recherchés. Ce paramètre est très difficile à obtenir si l'on ne possède aucune information *a priori* sur le nombre de modalités de chacune des classes. Deux méthodes sont alors fréquemment utilisées pour le déterminer automatiquement.

La première méthode consiste à utiliser un critère de validité mesurant la qualité d'une partition et à choisir celle qui optimise ce critère pour des valeurs de C comprises entre C_{min} et C_{max}. De nombreuses études ont été faites sur le sujet, que ce soit dans le cas des algorithmes classiques [123] ou bien dans la cas des algorithmes flous [22, 176, 98, 185, 174]. La plupart se fondent sur une évaluation des distances intra-regroupements c'est-à-dire sur la compacité des groupes, ou encore sur la distance inter-regroupements c'est-à-dire sur leur séparation. D'autres utilisent aussi des critères géométriques. Nous ne répertorions ici que quelques unes des approches les plus classiques en environnement flou :

- l'entropie d'une partition U, qui dérive de l'entropie de Shannon est définie par :

$$Entropie(U) = -\frac{\sum_{c=1}^{C}\sum_{j=1}^{N}\mu_{jc}log(\mu_{jc})}{N}$$

- le cœfficient de partition [22] $coefPart$ qui mesure le recouvrement entre les différents regroupements, s'évalue de la manière suivante :

$$coefPart(U) = \frac{\sum_{j=1}^{N}\sum_{c=1}^{C}\mu_{jc}^2}{N} \tag{3.14}$$

Ce critère est compris entre $[0, 1]$ avec une partition « optimale » (sans recouvrement) obtenue pour $coefPart(U) = 1$.

L'inconvénient de ces méthodes provient de leur sensibilité vis-à-vis des paramètres utilisés. Ainsi, la valeur du cœfficient de partition et de l'entropie varie en fonction des valeurs de m et de C notamment. [174] propose alors un critère mieux adapté pour l'utilisation des CMF et insensible aux paramètres. D'autres critères ont aussi été élaborés afin de prendre en compte à la fois les propriétés liées au partitionnement flou (par le biais des fonctions d'appartenance) et les propriétés géométriques des données. Ce point est là encore très important dans le cadre de notre étude pour que les prototypes extraits reflètent bien la structure des données.

Le critère que nous utilisons est l'*indice de Xie et Beni* [176] qui se base sur le rapport entre la compacité et la séparabilité des regroupements. Ce critère, noté $indice_{XB}$, doit être minimisé pour obtenir une bonne partition :

$$indice_{XB}(U) = \frac{\sum_{c=1}^{C} \sum_{j=1}^{N} \mu_{jc}^2 \, d^2(e_j - P_c)}{N \ \min_{l,c} \, d^2(P_l - P_c)} \tag{3.15}$$

Il existe bien entendu d'autres critères comme par exemple celui proposé dans [185].

Outre l'utilisation d'un critère de validité pour élaborer une partition correcte, il est aussi possible de procéder par agglomération ou division des regroupements. L'idée consiste à rechercher une partition sur-découpée puis à fusionner les différents regroupements en fonction de leur similarité, ou au contraire à partir d'un seul regroupement puis à essayer de le diviser. Pour plus d'informations sur ces méthodes, le lecteur peut se référer à [98, 99].

3.3.4.2 Définition des modèles intrinsèques aux classes

Il convient de souligner ici que l'extraction des modèles intrinsèques s'effectue par défaut en utilisant l'espace de représentation complet (c'est-à-dire en n dimensions). Cependant, lorsque n est grand, il peut être intéressant de travailler sur un sous-espace de dimension réduite pour accélérer les traitements. Lorsque cela est nécessaire [5], nous effectuons la recherche de ce sous-espace en nous appuyant sur un algorithme génétique standard [67] (cf. annexe C). Celui-ci cherche à optimiser les performances en reconnaissance (sur une base de validation) du module de décision associé à la modélisation intrinsèque (cf. module de pré-classification dans le chapitre 4). On remarquera alors que même si des informations de discrimination sont utilisées pour choisir ce sous-espace, les connaissances extraites représentent toujours le caractère intrinsèque des classes (seul le contexte associé à ces connaissances est différent). [6]

Pour obtenir les modèles intrinsèques, l'algorithme des CMP est lancé pour chaque classe ω_i à partir de la base B_{app}^i correspondante.

5. Ce choix est fait *a priori*.

6. On pourrait aussi choisir d'accentuer ce caractère intrinsèque en recherchant un sous-espace qui favorise à la fois la compacité et la séparabilité des prototypes flous en utilisant un critère comme $indice_{XB}$.

Un modèle intrinsèque $MI(\omega_i)$ est constitué d'autant de sous-classes que de regroupements déterminés en utilisant le critère $indice_{XB}$. Soit L_i ce nombre. Chaque sous-classe est représentée par un sous-ensemble flou $F^l_{\omega_i}$, $l = 1, \ldots, L_i$ dont les fonctions d'appartenance $\mu_{F^l_{\omega_i}}$ dérivent de l'équation (3.13). Elles sont définies par leur centre $P^l_{\omega_i}$ et leur matrice de covariance $Cov^l_{\omega_i}$. Pour un individu e_j, son degré d'appartenance à $F^l_{\omega_i}$ est donc déterminé par :

$$\mu_{F^l_{\omega_i}}(e_j) = \frac{1}{1 + (e_j - P^l_{\omega_i})^T (Cov^l_{\omega_i})^{-1} (e_j - P^l_{\omega_i})} \tag{3.16}$$

La figure 3.13 illustre un exemple de modèle intrinsèque en deux dimensions d'une classe constituée de trois sous-classes.

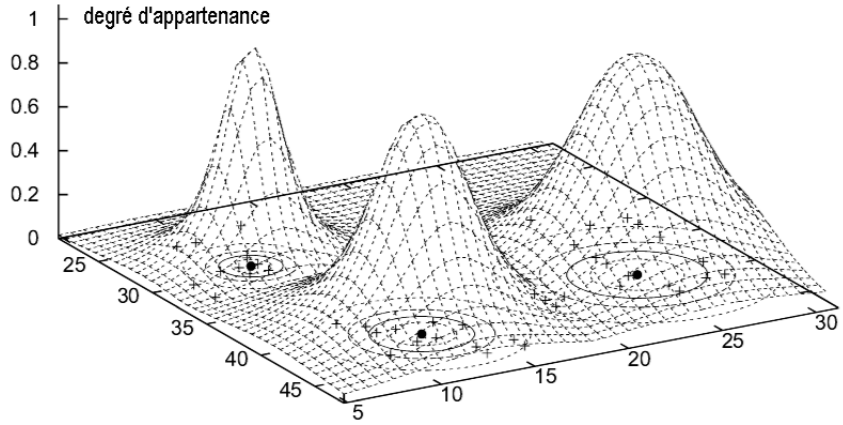

FIG. 3.13 – *Exemple de modèle intrinsèque pour une classe constituée de trois sous-classes.*

On peut facilement déduire de cette description l'adéquation d'un point de vue intrinsèque d'un individu e_j à une classe ω_i, c'est-à-dire son adéquation au modèle $MI(\omega_i)$. Pour cela, nous utilisons la t-conorme max comme opérateur de disjonction :

$$\mu_{MI(\omega_i)}(e_j) = \max_{l=1,\ldots,L_i} (\mu_{F^l_{\omega_i}}(e_j)) \tag{3.17}$$

L'interprétation intuitive correspondante est qu'un individu appartient d'autant plus au modèle d'une classe qu'il appartient à une de ses sous-classes, c'est-à-dire si ses propriétés sont compatibles avec les propriétés intrinsèques de la classe.

3.3.5 Principes de la décomposition en sous-problèmes

Pour déterminer les sous-problèmes sur lesquels doit se concentrer le module de discrimination focalisée, la base d'apprentissage initiale B_{app} est divisée en sous-bases B^i_{app2}. Celles-ci sont constituées de l'ensemble des formes ayant des propriétés intrinsèques en adéquation[7] avec le modèle intrinsèque de la classe ω_i. La décomposition

7. Ce mécanisme est donc assez similaire au principe du filtrage flou [27].

n'est donc pas stricte : un même exemple peut se retrouver dans plusieurs sous-bases (cf. figure 3.14).

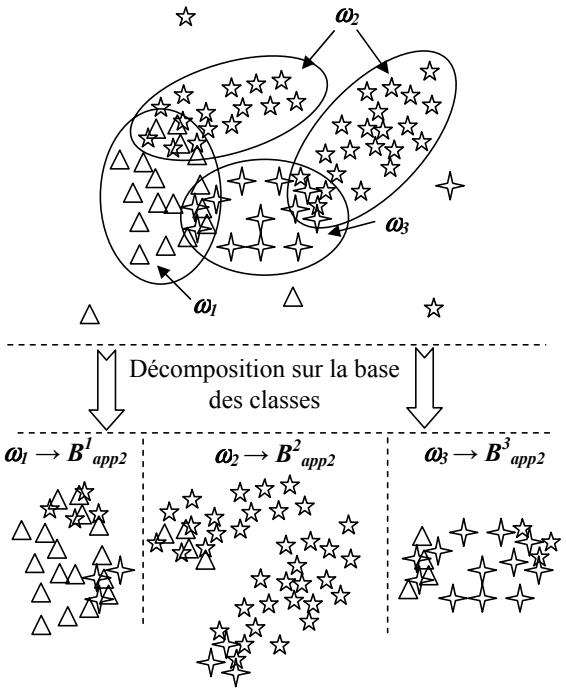

FIG. 3.14 – *Exemple du mécanisme de décomposition à partir des modèles intrinsèques de trois classes.*

Une première solution pour constituer celles-ci consiste à utiliser un mécanisme d'α-coupe :

$$B^i_{app2} = \{e_j \in B_{app} | \mu_{MI(\omega_i)}(e_j) \geq \alpha_i\} \tag{3.18}$$

où α_i est à déterminer pour chaque classe soit *a priori* soit par apprentissage ce qui s'avère en général assez complexe.

Une solution plus simple et qui expérimentalement s'avère efficace consiste à utiliser non pas un seuillage absolu par classe mais un seuillage relatif à chaque individu. Ainsi, un exemple sera considéré comme pouvant appartenir à toutes les classes pour lesquelles son adéquation sera suffisamment importante par rapport à la classe privilégiée :

$$B^i_{app2} = \{e_j \in B_{app} | \mu_{MI(\omega_i)}(e_j) \geq \frac{\max_{s=1,\ldots,\mathcal{S}}(\mu_{MI(\omega_s)}(e_j))}{\alpha}\} \tag{3.19}$$

Dans cette approche, il n'y a plus qu'une seule valeur α à déterminer, contrairement aux $\mathcal{S}$ valeurs α_i de la méthode précédente. Expérimentalement, une valeur comprise entre 2 et 3 donne de bons résultats. On notera cependant que cette façon de faire comporte certains risques, notamment si la base d'apprentissage contient du bruit (des exemples n'appartenant à aucune classe). En effet, même si un individu possède un

degré d'appartenance faible à tous les modèles intrinsèques, celui-ci sera tout de même ajouté dans les sous-bases. Toutefois, l'intégration du mécanisme de rejet de distance (qui est un des objectifs) devrait permettre d'éviter ces effets de bords.

Une fois les sous-bases constituées, il faut réétiqueter les individus pour que les experts du module de discrimination focalisée puisse opérer convenablement : les individus de B^i_{app2} appartenant effectivement à ω_i ($e_j^{(n+1)} = \omega_i$) sont réétiquetés comme des exemples positifs : $e_j^{(n+1)} = \omega_{i+}$, et tous les autres comme des contre-exemples : $e_j^{(n+1)} = \omega_{i-}$.

Les connaissances discriminantes peuvent à présent être extraites dans chacune de ces sous-bases pour discriminer les exemples positifs des exemples négatifs.

3.4 Niveau discriminant : extraction, modélisation et structuration des connaissances de nature discriminante

Le niveau discriminant est composé d'un ensemble de modèles discriminants, un pour chaque modèle intrinsèque $MI(\omega_i)$. Ils représentent les connaissances permettant de différencier les formes appartenant à ω_{i+} de celles appartenant à ω_{i-}. Nous avons vu dans le paragraphe 3.1.2 que ces connaissances possédaient des propriétés spécifiques à leur nature qui nécessitaient notamment de les structurer en fonction de leur pertinence pour la discrimination. Nous avons aussi souligné qu'elles étaient fortement dépendantes de leur contexte d'utilisation. L'élaboration de ce niveau nécessite donc de pouvoir résoudre l'ensemble des points suivants :

- comment extraire automatiquement un ensemble de connaissances discriminantes dans un contexte donné et comment les représenter par des sous-ensembles flous ;
- comment choisir automatiquement les connaissances les plus pertinentes dans ce même contexte ;
- comment passer d'un contexte à un autre et faire le lien entre les connaissances (c'est-à-dire les structurer) pour rendre la modélisation à la fois robuste et interprétable.

Dans ce contexte, les *arbres de décision flous* constituent une approche de modélisation tout à fait intéressante. En effet, ils permettent à la fois d'extraire des connaissances discriminantes et de les organiser hiérarchiquement en fonction de leur pouvoir de discrimination. De plus, ils opèrent de façon récursive en se focalisant peu à peu sur des sous-problèmes de plus en plus simples représentant des contextes différents. Enfin, en adoptant un mode de représentation à base de sous-ensembles flous, la modélisation est rendue plus robuste et elle possède de meilleures propriétés d'interprétabilité.

Dans les paragraphes suivants, nous présentons dans un premier temps comment la structure d'arbre permet d'organiser les connaissances discriminantes. Nous faisons ensuite un bref rappel des mécanismes d'apprentissage associés aux arbres de décision

classiques. Ceux-ci sont importants car ils nous servent de base pour pouvoir présenter notre nouvelle approche pour la construction d'arbres de décision flous. Ceux-ci ont été élaborés de façon à s'intégrer à notre problématique pour obtenir les propriétés requises. Ils reposent pour cela sur l'utilisation des CMF pour extraire des connaissances discriminantes robustes et interprétables par rapport aux données.

3.4.1 La structure d'arbre de décision pour organiser les connaissances discriminantes

Le principe des arbres de décision pour la classification a été présenté très brièvement en introduction (cf. paragraphe 1.3.7). Rappelons simplement que leur objectif est de mettre en relation un ensemble de connaissances relatives aux propriétés des formes dans l'espace de représentation de façon à pouvoir discriminer les classes auxquelles elles appartiennent. Ces connaissances se présentent sous la forme de conditions sur la valeur des attributs et leurs relations sont décrites par une arborescence facilement interprétable.

De manière formelle, la structure d'un arbre de décision est un graphe $G = (\mathcal{N}, \mathcal{A})$ composé d'un ensemble de *nœuds* (ou encore sommets) $\mathcal{N}$ et d'*arcs* $\mathcal{A}$. Chaque nœud $\mathcal{N}_{Id}$ de l'arbre possède un identifiant que nous notons Id d'une façon générale. Cet identifiant est en fait une séquence d'indices repérant la position du nœud dans l'arbre. Chaque arc $(\mathcal{N}_{Id}, \mathcal{N}_{Id.l})$ est orienté et relie un nœud *père* $\mathcal{N}_{Id}$ et un nœud *fils* $\mathcal{N}_{Id.l}$ où $Id.l$ est la concaténation (.) de l'identifiant Id du père avec l'indice l du fils (son numéro parmi l'ensemble des fils du nœud père). Chaque nœud possède exactement un père, à l'exception de la *racine* de l'arbre qui n'en possède aucun. Son identifiant est 1 et elle est notée $\mathcal{N}_1$. Les nœuds terminaux, appelés *feuilles*, ne possèdent pas de fils. Un *chemin* (ou branche) dans l'arbre est une suite d'arcs qui relie deux sommets. Un nœud non terminal $\mathcal{N}_{Id}$ est étiqueté par un des attributs $F_k, k = 1, \ldots, n$ de l'espace de représentation. Les arcs sont quant à eux étiquetés par une condition F_{kl} portant sur les valeurs de F_k et qui permet de passer du nœud père Id au nœud fils $Id.l$ si elle est satisfaite. Cette condition représente en général une modalité de l'attribut ou un ensemble de valeurs. Enfin, les feuilles sont étiquetées par une classe $\omega_i \in \mathcal{S}$ représentant la décision qui doit être prise si une forme satisfait l'ensemble des conditions le long du chemin allant de la racine à la feuille.

La figure 3.15 présente un exemple partiel et simplifié d'arbre de décision pour la discrimination de lettres latines, tracées en-ligne. Les attributs de l'espace de représentation portent ici sur les propriétés morphologiques des tracés et les classes sont les lettres de l'alphabet latin.

Grâce à la structuration des connaissances sous cette forme arborescente, l'interprétation est rendue relativement aisée. Ainsi, sur l'exemple précédent, si un caractère possède trois traits descendants alors c'est un 'm'. De la même manière, si un caractère ne possède qu'un trait descendant et qu'il possède un signe diacritique dont le rapport hauteur/largeur est supérieur ou égal à $0,5$ alors il s'agit d'un 'i' ou d'un 'j'. On notera que cet exemple n'est que partiel et qu'ils ne permet pas de faire une discrimination

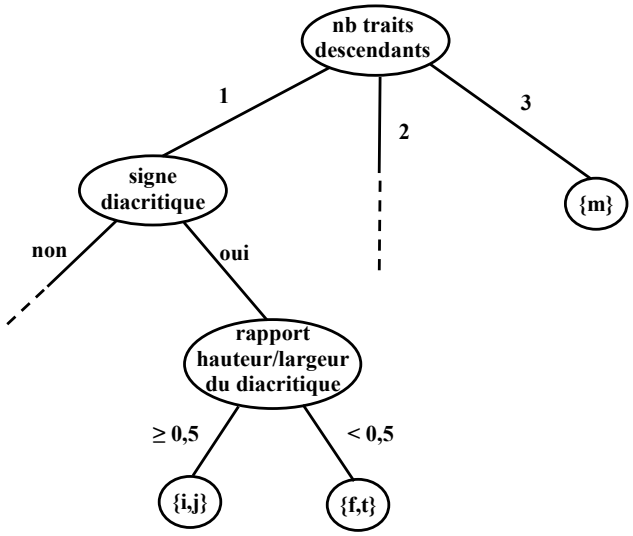

FIG. 3.15 – *Exemple d'arbre de décision (partiel) pour la discrimination de caractères latins.*

complète des classes (les feuilles ne sont pas étiquetées par une seule classe). Pour pouvoir discriminer le 'i' du 'j' et le 'f' du 't', il faudrait raffiner l'arbre en y incorporant d'autres connaissances discriminantes.

Ce type de structure a été utilisé dans de nombreux domaines applicatifs où son interprétabilité ainsi que son habileté à manipuler des attributs à la fois numériques et symboliques en ont fait une approche privilégiée par rapport à d'autres approches de classification. Cependant, dans le cadre de la reconnaissance de formes, sa capacité à gérer efficacement les données numériques reste limitée. En effet, comme la plupart des données réelles manipulées sont sujettes aux imprécisions et au bruit, l'utilisation de conditions strictes comme « $\geq 0,5$ » provoque parfois des comportements indésirables et sont source d'erreurs. Un des intérêts des arbres de décision flous, comme celui que nous présentons plus loin, est justement de limiter ces effets.

Dans le paragraphe suivant, nous décrivons un processus d'apprentissage classique permettant d'élaborer automatiquement un arbre de décision, c'est-à-dire d'extraire les connaissances discriminantes et de les structurer sous la forme d'une arborescence. Cela nous permet de mettre en lumière un certain nombre de points importants qui nous ont guidés dans notre approche.

3.4.2 Mécanisme d'apprentissage général des arbres de décision

Les arbres de décision ont fait l'objet d'un grand nombre d'études, selon différentes considérations, qu'elles soient d'ordre statistique, liées à l'intelligence artificielle, à la reconnaissance des formes, etc. Il en résulte un vaste choix d'algorithmes, chacun ayant

ses particularités. Le lecteur peut se référer à [30, 31, 37, 59, 82, 108, 141, 142, 115]. Parmi les principales approches, on peut citer l'algorithme *CART* (Classification and Regression Trees) de Breiman *et al.* [30] et l'algorithme *ID3* de Quinlan [141] qui sont à l'origine de beaucoup d'autres méthodes de construction d'arbres de décision classiques et flous. Il existe aussi un certain nombre d'articles qui font un état de l'art de ces différentes méthodes [17, 97, 129, 150, 121, 188]. La présentation qui suit s'inspire des approches descendantes et plus particulièrement de l'algorithme du *C4.5* [142] qui dérive lui-même de l'*ID3*.

La construction de l'arbre s'effectue à partir d'une base d'apprentissage B_{app} initiale étiquetée[8] telle que nous l'avons définie dans le paragraphe 1.2.4.4. L'objectif est d'extraire de cette base les connaissances pertinentes permettant de partitionner la base d'apprentissage en sous-ensembles d'individus représentatifs d'une unique classe, opérant ainsi une discrimination des classes. Ce partitionnement repose sur le principe de la stratégie « diviser pour régner » afin de diviser le problème de discrimination initial en sous-problèmes plus faciles à traiter au lieu de le considérer dans sa globalité. Pour cela, les partitionnements s'effectuent de manière récursive sur une base d'exemples B ($B = B_{app}$ au départ) à partir de conditions sur la valeur des attributs $F_k, k = 1, \ldots, n$ de l'espace de représentation des formes. Ces attributs prennent un ensemble de valeurs (ou modalités) $F_{kl},\ l = 1, \ldots, L_k$. L'extension aux attributs numériques continus sera discutée dans le paragraphe suivant (cf. paragraphe 3.4.2.2).

L'algorithme d'apprentissage se résume alors de la manière suivante :

INITIALISATION :
$B = B_{app}$.
CONSTRUIRE-ARBRE(B)
DEBUT
  SI le critère d'arrêt est vérifié
    ALORS créer une feuille
  SINON
    choisir un attribut F_k *et créer un nœud*
    partitionner B *en* L_k *sous-ensembles* $B_{kl},\ l = 1, \ldots, L_k$
    Pour $l = 1, \ldots, L_k$ *CONSTRUIRE-ARBRE*(B_{kl})
  FINSI
FIN

Cet algorithme récursif est caractérisé par quatre notions fondamentales qui permettent de distinguer les différentes approches :

– le choix d'un attribut pour partitionner B ;

8. Il existe aussi certaines méthodes d'apprentissage utilisant d'autres sources d'informations comme les probabilités *a priori*, une base de règles, un réseau de neurones, etc. (cf. [129] pour une revue sur les arbres de décision non flous)

- une stratégie de partitionnement ;
- un critère d'arrêt ;
- la création d'une feuille.

Chacune d'elle est décrite en détails dans les paragraphes suivants.

3.4.2.1 Le choix d'un attribut pour partitionner

Lors de la construction d'un nœud non terminal, il faut déterminer l'attribut qui doit permettre de partitionner la base courante B d'individus. Ce point est crucial dans le mécanisme de construction d'arbre de décision puisqu'il influe directement sur la structure finale de l'arbre et donc sur ses propriétés. Rappelons que les connaissances discriminantes, telles que nous les avons définies (cf. paragraphe 3.1.2) ne sont pas toutes aussi pertinentes puisqu'elles sont caractérisées par leur pouvoir de discrimination. De plus, cette pertinence est fortement dépendante du contexte dans lequel elle est évaluée. Choisir les attributs les plus discriminants localement à chaque nœud permet donc d'obtenir une modélisation plus robuste mais aussi plus compacte et lisible [121, 33].

Pour évaluer la pertinence d'un attribut relativement à la distribution en classes d'un ensemble d'individus, on utilise le plus souvent une mesure de discrimination. De nombreux travaux ont été effectués sur l'élaboration de ces mesures et les propriétés qu'elles doivent posséder [30, 121, 111, 142].

La mesure la plus utilisée est probablement *l'entropie de Shannon* [156] qui repose sur les principes de la théorie de l'Information. Dans le cas général où $P = (p_1, p_2, \ldots, p_n)$ est une distribution de probabilité, l'entropie est donnée par la formule suivante :[9]

$$H(P) = -\sum_{i=1}^{n} p_i \log p_i \tag{3.20}$$

Ainsi, $H(P)$ sera maximale si $p_1 = p_2 = 0,5$ c'est-à-dire si les distributions sont fortement corrélées. Elle sera au contraire minimale si p_1 ou p_2 est nul.

Intégrons à présent cette mesure dans le processus d'apprentissage des arbres de décision. Rappelons tout d'abord que $S = \{\omega_1, \ldots, \omega_i, \ldots, \omega_{\mathcal{S}}\}$ est l'ensemble des classes et que B désigne un ensemble de N individus[10] à partitionner au niveau d'un nœud. Parmi ces échantillons, N^i appartiennent à la classe ω_i et la probabilité associée est $p(\omega_i) = \frac{N^i}{N}$. L'information relativement à la distribution en classes S sur B s'écrit :

$$H_B(S) = -\sum_{i=1}^{\mathcal{S}} p(\omega_i) \log p(\omega_i) \tag{3.21}$$

Considérons maintenant un attribut F_k prenant les modalités $F_{kl}, l = 1, \ldots, L_k$. Si cet attribut est choisi pour partitionner B, les N exemples sont partagés en L_k

9. Le logarithme est ici en base deux.

10. Pour éviter une multipication des notations, N représente à la fois le nombre d'individus de la base d'apprentissage initiale B_{app} ainsi que celui des différentes sous-bases B résultant des partitionnements.

sous-ensembles $B_{kl}, l = 1, \ldots, L_k$ suivant la modalité à laquelle ils appartiennent (cf. équation (3.24) dans le paragraphe 3.4.2.2 pour plus de détails sur le mode de partitionnement). Les effectifs associés aux B_{kl} sont notés N_{kl}. Parmi eux, N_{kl}^i appartiennent à la classe ω_i. L'entropie associée à cette division relativement aux classes est appelée *entropie conditionnelle* :

$$H_B(S|F_k) = -\sum_{l=1}^{L_k} p(F_{kl}) \sum_{i=1}^{\mathcal{S}} p(\omega_i|F_{kl}) \log p(\omega_i|F_{kl}) = \sum_{l=1}^{L_k} p(F_{kl}) H_{B_{kl}}(S) \tag{3.22}$$

avec

$$p(\omega_i|F_{kl}) = \frac{N_{kl}^i}{N_{kl}}$$

et

$$p(F_{kl}) = \frac{N_{kl}}{N}.$$

Prendre l'attribut F_k le plus discriminant revient alors à choisir celui qui maximise le *gain d'information* :

$$G_B(F_k) = H_B(S) - H_B(S|F_k) \tag{3.23}$$

La *mesure d'impureté* de *Gini* [30] ou encore le critère du χ^2 sont d'autres exemples bien connus de mesures de discrimination parmi les nombreuses autres qui existent [111, 122, 157, 115]. Étant donné le nombre important de mesures, il est tout naturel de s'interroger sur laquelle choisir. De nombreuses études ont été conduites, la plupart du temps de manière empirique, pour essayer d'établir une hiérarchisation des mesures de discriminations. La plupart de ces travaux convergent vers la même constatation : il semblerait qu'aucune mesure ne soit supérieure à une autre de manière significative [30, 126] et que le choix de ce critère soit nettement moins important pour l'efficacité de l'arbre de décision que d'autres paramètres comme le critère d'arrêt [30]. Mais cela ne signifie nullement que les attributs peuvent être choisis au hasard et ce pour toutes les raisons évoquées plus haut.

Il existe malgré tout des différences entre les mesures, différences qui peuvent aider à choisir en fonction du type d'arbre que l'on souhaite construire. Par exemple, le critère de *Gini* se révèle mal adapté pour mesurer l'impureté quand plusieurs classes doivent être séparées [30]. De plus, ce critère tend à favoriser les attributs produisant une partition dans laquelle les sous-ensembles sont de taille et de pureté équivalente. Au contraire, l'entropie favorise les partitions les plus pures, ce qui peut aboutir à des arbres déséquilibrés. Certains auteurs ont aussi montré que cette mesure était biaisée en faveur des attributs possédant un grand nombre de modalités [141]. Quinlan a alors proposé le *gain ratio* [142] pour réduire ce biais. Marsala [121, 120] a aussi présenté un cadre de travail pour l'étude et la construction de mesures de discrimination à partir d'un modèle hiérarchique de fonctions. Celui-ci permet d'étudier le comportement des mesures de discrimination les unes par rapport aux autres afin de les choisir en fonction

des propriétés désirées.

Dans le cadre de nos arbres de décision flous, la mesure de discrimination doit être adaptée à l'environnement flou afin de prendre en compte l'imprécision des données. Celle que nous utilisons est une extension de l'entropie de Shannon.

3.4.2.2 La stratégie de partitionnement

La procédure de construction des arbres de décision vise à partitionner l'ensemble d'apprentissage jusqu'à obtenir des sous-ensembles d'individus appartenant tous à la même classe. On dit alors que ces sous-ensembles sont « purs ». Les partitionnements s'effectuent dans l'espace de représentation, suivant les différentes modalités des attributs. Il existe alors différentes possibilités pour effectuer un partitionnement au niveau d'un nœud, selon que les attributs sont nominaux ou continus.

Dans le cas des attributs nominaux, un attribut F_k possède un nombre fini L_k de modalités. On peut donc effectuer un partitionnement suivant ces valeurs. Soit B la sous-base de N individus à un nœud donné et F_k un attribut utilisé pour faire le partitionnement. B est divisée en L_k sous-bases B_{kl}, $l = 1, \ldots, L_k$ en répartissant les échantillons $e_j, j = 1, \ldots, N$ en fonction de la modalité qu'ils possèdent pour l'attribut considéré :

$$B_{kl} = \{e_j \in B \mid e_j^k = F_{kl}\} \tag{3.24}$$

Dans certains cas, en fonction de la méthode utilisée pour choisir les attributs, ou bien pour éviter une division trop rapide de la base d'apprentissage, ou encore pour des raisons de lisibilité, il peut être souhaitable de se ramener à des partitionnements binaires. Pour cela, un attribut à L_k modalités est dérivé en L_k attributs caractérisés chacun par une modalité de l'attribut d'origine et une autre correspondant à la réunion des modalités restantes. Par exemple, si l'attribut *couleur* prend les modalités {*rouge, vert, bleu*}, trois attributs binaires peuvent en être déduits. Le premier aura les modalités {*rouge, autre*}, le deuxième {*vert, autre*} et le troisième {*bleu, autre*}. C'est l'un des deux processus utilisé par l'algorithme CART [30] lorsque le nombre d'attributs est grand. La deuxième méthode, utilisée quand le nombre d'attributs est réduit, consiste à dériver un attribut pour chaque paire de modalités possible ({*rouge, vert*}, {*rouge, bleu*}, {*vert, bleu*} dans le cas de notre exemple).

Le problème est plus difficile avec les attributs continus. Il est théoriquement possible de discrétiser un espace de représentation numérique afin de créer artificiellement des modalités, mais il est plus difficile d'obtenir une discrétisation qui ait un sens par rapport au problème et qui soit robuste. Ce sont donc souvent des experts qui effectuent cette tâche.

Il existe cependant plusieurs techniques de partitionnement automatique à partir d'attributs numériques continus. Dans le *C4.5*, Quinlan [142] propose d'ordonner les valeurs prises sur un attribut F_k par les N individus de la base B, c'est-à-dire localement à un nœud. Soit $(e_1^k, e_2^k, \ldots, e_N^k)$ cette liste avec $e_1^k \leq e_2^k \leq \ldots \leq e_N^k$. À partir de celle-ci,

les seuils $Sj = (e_j^k + e_{(j+1)}^k)/2, \quad j = 1, \ldots, N-1$ sont calculés, générant autant de partitionnements binaires reposant sur les nouvelles modalités : $\{F_k \leq Sj, F_k > Sj\}$.

De nombreuses autres études ont été faites pour élaborer des techniques de partitionnement reposant sur les attributs continus [143, 30, 55, 121]. Certains auteurs ont aussi étudié la possibilité de faire des partitionnements non pas en se basant sur un seul attribut, mais en en utilisant plusieurs afin de bénéficier de l'interaction entre les attributs, notamment du point de vue de leur pouvoir de discrimination. Les techniques utilisées reposent sur leur combinaison linéaire [31, 115] ou non linéaire [82] ou encore sur l'utilisation de réseaux de neurones [69, 151]. Les arbres de décision résultant sont souvent plus efficaces mais ils sont aussi plus coûteux à apprendre et à construire. De plus, l'interprétabilité de la modélisation s'en trouve fortement réduite alors que c'est un des atouts des arbres de décision.

Ces techniques de partitionnement ont montré que lorsque l'espace de représentation utilisé est continu, le partitionnement s'effectue très souvent localement à un nœud afin d'exploiter le caractère contextuel des connaissances discriminantes. De plus, l'utilisation de plusieurs attributs en même temps permet d'être plus efficace pour la discrimination. Ce sont donc deux points importants que nous exploitons dans notre méthode par arbres de décision flous.

3.4.2.3 Critère d'arrêt et élagage d'un arbre de décision

Le critère d'arrêt est un point essentiel de la procédure de construction de l'arbre car il détermine en grande partie son efficacité, tant au niveau de sa compacité que de ces capacités de généralisation [30].

Une première idée simple consiste à grossir l'arbre (en réitérant la procédure CONSTRUIRE-ARBRE) tant que les nœuds ne sont pas purs ; c'est-à-dire tant qu'ils contiennent des exemples de classes différentes. L'arbre de décision ainsi obtenu décrit exactement l'ensemble d'apprentissage et ne fait aucune erreur de classification sur cette base. Cette approche possède deux inconvénients majeurs : d'une part les capacités de généralisation de l'arbre de décision seront très souvent mauvaises, conduisant à un sur-apprentissage ; d'autre part, la taille de l'arbre risque de devenir très importante si les classes sont difficilement séparables. L'impact sur la taille de l'arbre dépend en général du nombre d'exemples dans la base d'apprentissage : plus le nombre d'individus utilisé en apprentissage est important et plus l'arbre risque d'être de taille importante. Ce phénomène est accru par la présence de bruit dans les données. Outre les problèmes d'utilisation de ressources mémoire et CPU que cela suscite, un arbre trop grand est difficilement interprétable, ce qui n'est pas souhaitable en général.

Pour limiter ces effets néfastes, d'autres critères d'arrêt sont introduits dans la procédure de construction. On distingue principalement deux approches : l'*arrêt « précoce »* et l'*élagage*.

La première agit pendant la construction de l'arbre tout comme le critère d'arrêt sur la pureté d'un nœud. Classiquement, une feuille est construite lorsque le nombre

d'échantillons de la base B d'un nœud est trop faible ou encore lorsque la pureté de ce sous-ensemble d'individus est jugée suffisante. Cette estimation peut se déduire directement de la mesure de discrimination utilisée pour les partitionnements. Cependant, ces mesures sont calculées localement à un nœud et sont souvent fortement dépendantes du nombre d'échantillons. Il est donc difficile d'établir un seuil unique qui puisse être valable pour tous les nœuds de l'arbre. La construction de l'arbre risquerait alors de s'arrêter prématurément pour certains nœuds ou au contraire tardivement pour d'autres. D'autres variantes ont été conçues pour pallier à ces problèmes comme par exemple une mesure d'impureté qui se calcule sur l'arbre complet [122].

Au lieu d'arrêter la construction de l'arbre de manière « précoce », certains auteurs comme Breiman [30] suggèrent de construire l'arbre de décision de taille maximale puis de réduire sa taille dans une seconde phase par une méthode d'*élagage*. Le principe général consiste à supprimer des nœuds ou des sous-arbres peu représentatifs ou encore à les regrouper. De nombreuses techniques ont été étudiées montrant pour la plupart une efficacité réelle par rapport à l'arbre non élagué. Dans [30], les auteurs procèdent en deux étapes. La première consiste à construire une série d'arbres de plus en plus petits à partir de la base d'apprentissage en supprimant les nœuds en fonction d'un critère basé sur le rapport entre le taux d'erreur de l'arbre sur la base d'apprentissage et sa taille (*cost complexity*). La seconde utilise une base de validation pour choisir l'arbre qui a le taux d'erreur le plus faible sur cette base. Comme il est parfois difficile d'obtenir une base de validation, les auteurs proposent une variante reposant sur le principe de la validation croisée (cf. paragraphe 1.2.6.1). Cette dernière méthode est cependant assez coûteuse. Une autre solution proposée par Quinlan [142], consiste à utiliser le taux d'erreur sur la base d'apprentissage ajusté de manière statistique pour prendre en compte le biais introduit. C'est aussi le principe adopté dans [117, 92] où l'algorithme d'élagage, qualifié de « pessimiste », à l'avantage de ne fonctionner qu'en une seule passe, contrairement à beaucoup d'autres.

Dans les arbres de décision flous, le critère d'arrêt est moins important que dans les arbres de décision classiques. En effet, grâce à la modélisation par sous-ensembles flous, les données sont synthétisées et l'expansion de l'arbre implicitement limitée. Nous verrons cependant lors des expérimentations que l'influence de ce critère n'est pas à n'égliger (cf. paragraphe 5.2).

3.4.2.4 Création d'une feuille

Une feuille diffère d'un nœud non terminal par le fait qu'elle marque l'arrêt de l'expansion d'une branche, parce qu'un des critères d'arrêt a été satisfait.

La feuille signifie alors que l'on peut (doit) être capable d'identifier la classe la plus représentative dans la sous-base B locale. Lorsque cette sous-base est pure, il n'y a qu'une seule classe possible et c'est elle qui étiquette la feuille. C'est le cas idéal mais il est rarement satisfait en pratique. Dans tous les autres cas, la règle générale consiste à étiqueter la feuille avec la classe possédant le plus de représentants.

3.4.2.5 Inconvénients des arbres de décision classiques dans le cadre de notre approche

Le gros inconvénient des arbres de décision tels qu'ils ont été présentés ici provient de la façon dont les conditions sur les attributs sont utilisées pour faire les partitionnements. Ces conditions sont strictes, ne laissant aucune alternative possible pour attribuer un individu à un des fils du nœud considéré. Les partitionnements résultants peuvent ainsi devenir arbitraires, rendant la modélisation moins robuste voire incohérente (et donc difficilement exploitable). Ce phénomène est encore accru si les données manipulées sont imprécises. Plus particulièrement, pour une approche de reconnaissance de forme travaillant dans un espace de représentation numérique, les conditions sur les attributs se traduisent essentiellement par des frontières séparatrices nettes. Celles-ci étant déterminées de façon à optimiser la séparation des classes (par le biais de la mesure de discrimination), elles ne possèdent bien souvent aucun lien explicite avec les propriétés des formes telles qu'elles apparaissent dans l'espace de représentation. Les connaissances qui en résultent manquent donc souvent de robustesse et sont difficiles à interpréter, ce qui ne satisfait pas nos objectifs. Les arbres de décision flous apportent alors une solution intéressante à ces différents problèmes.

3.4.3 Les arbres de décision flous

Afin d'améliorer la robustesse des arbres de décision classiques, certains auteurs comme Quinlan [142] ont modifié le mode de représentation de leurs arbres afin d'utiliser des conditions plus souples sur les attributs numériques. Les arbres de décision flous sont alors une extension directe reposant sur le formalisme bien établi de la théorie des sous-ensembles flous. Les premiers travaux dans le domaine sont ceux de Chang et Pavlidis [35]. L'idée principale consiste à décrire les attributs par des sous-ensembles flous, que nous appellerons modalités floues, et à utiliser celles-ci dans les conditions permettant de faire les partitionnements. Par exemple, dans la figure 3.15, les deux conditions portant sur le rapport hauteur/largeur du diacritique peuvent être définies par les deux propositions floues suivantes : « la forme du diacritique est un point » (rapport hauteur/largeur proche de 1) et « la forme du diacritique est un trait » (rapport hauteur/largeur faible). Au niveau de la structure de l'arbre, les arcs sont alors étiquetés par les sous-ensembles flous (termes linguistiques) correspondants.

Les méthodes liées à l'apprentissage des arbres de décision flous [36, 91, 89, 121, 130, 172] peuvent être considérées comme une extension de celles des arbres de décision classique. En s'appuyant sur la méthode générale présentée dans le paragraphe précédent, on peut distinguer trois évolutions majeures :

- une « fuzzification » de l'espace de représentation c'est-à-dire la détermination de sous-ensembles flous pour décrire les attributs, qu'ils soient à l'origine symboliques ou numériques et continus ;
- l'adaptation de la méthode de partitionnement ;
- l'adaptation de la mesure de discrimination.

Nous présentons dans le paragraphe suivant, une nouvelle approche pour construire de tels arbres afin que celle-ci produise une modélisation en accord avec notre définition des connaissances discriminantes et avec nos objectifs.

3.4.4 Une nouvelle approche de construction d'arbres de décision flous pour l'extraction de connaissances discriminantes

Nous présentons ici un nouvel algorithme d'apprentissage pour les arbres de décision flous qui diffère des autres notamment par la stratégie de partitionnement et la manière d'extraire les sous-ensembles flous. Ces adaptations ont été faites dans le but d'extraire des connaissances discriminantes avec les propriétés que nous avons définies et de pouvoir intégrer l'ensemble dans notre architecture pour la reconnaissance de formes. Dans ce cadre, chaque chemin allant de la racine à une feuille peut être interprété comme un processus ou concept de discrimination.

3.4.4.1 L'algorithme d'apprentissage

L'algorithme que nous présentons ici est une extension de la procédure **CONSTRUIRE-ARBRE** du paragraphe 3.4.2. Il permet d'élaborer un arbre de décision flou à partir d'une base d'exemples B. Pour son utilisation dans notre architecture, il doit être appliqué sur chacune des sous-bases B^i_{app2} déterminées à partir des modèles intrinsèques. On notera ici que quelque soit l'espace de représentation utilisé par le niveau de modélisation intrinsèque (partiel ou complet), les arbres du niveau de modélisation discriminante choisissent les attributs qui les intéressent parmi l'espace complet.

L'algorithme de construction d'un arbre se présente de la manière suivante :

INITIALISATION :
$B = B^i_{app2}$; $\forall j = 1, \ldots, N\ \mu_1(e_j) = 1$
CONSTRUIRE-ARBRE-FLOU(B)
DEBUT
  SI le critère d'arrêt est vérifié
    ALORS créer une feuille $\mathcal{N}_f$ *et calculer la représentativité* d_{fi+} *et* d_{fi-} *des classes* ω_{i+} *et* ω_{i-}
  SINON
    Choisir un sous-espace F_k *à deux dimensions adapté à la discrimination des exemples de* B
    Créer un nœud $\mathcal{N}_{Id}$
    Partitionner B *en* L_k *sous-ensembles :* B_{kl}, $l = 1, \ldots, L_k$ *par les C-moyennes floues*
    Pour chaque sous-ensemble B_{kl}, *déterminer les degrés* $\mu_{\mathcal{N}_{Id.l}}(e_j)\ \forall e_j \in B_{kl}$
    Pour $l = 1, \ldots, L_k$ *CONSTRUIRE-ARBRE-FLOU*(B_{kl})
  FINSI
FIN

La différence essentielle entre cet algorithme et le précédent résulte de l'introduction du formalisme des sous-ensembles flous pour décrire les conditions étiquetant les arcs. Elle se matérialise par le fait que les individus qui sont propagés à travers l'arbre n'appartiennent plus de façon stricte aux différents nœuds mais au contraire à chacun d'eux avec différents degrés. Chaque observation e_j est donc caractérisée par un degré d'appartenance $\mu_{\mathcal{N}_{Id}}(e_j)$ au nœud courant $\mathcal{N}_{Id}$. À la racine de l'arbre, $\mu_{\mathcal{N}_1}(e_j) = 1 \; \forall j = 1, \ldots, N$. Au niveau d'une feuille f, $\mu_{\mathcal{N}_f}(e_j)$ indique à quel degré l'observation e_j correspond au processus de discrimination modélisé par le chemin allant de la racine à la feuille.

Une autre différence importante concerne le choix de l'espace utilisé au niveau d'un nœud pour faire le partitionnement. Dans un cadre classique, cet espace est monodimensionnel : il s'agit d'un seul attribut F_k. Dans notre approche, nous autorisons que cet attribut soit remplacé par un sous-espace de dimension fixe que nous notons aussi F_k pour éviter d'introduire des notations supplémentaires. Bien entendu, si la dimension des sous-espaces vaut 1, on retrouve le cas classique. Cette manière de faire procure une plus grande souplesse à l'algorithme et nous montrerons dans le paragraphe 3.4.4.4 que nous travaillerons dans $L_k = 2$ dimensions. La figure 3.16 illustre la structure d'un tel arbre de décision flou.

3.4.4.2 Obtention des modalités floues

Dans les arbres de décision classiques, les partitionnements au niveau des nœuds de l'arbre s'effectuent en s'appuyant sur les modalités existantes des attributs, sur le résultat de leur discrétisation ou encore par la détermination de frontières discriminantes s'ils sont numériques et continus. Dans les arbres de décision flous, les partitionnements se font à partir de modalités floues décrivant les attributs. La question que l'on se pose ici est de savoir comment obtenir ces modalités floues.

Plusieurs possibilités sont envisageables mais on distingue deux origines principales. La première est d'origine *humaine* et consiste à faire appel à un expert du domaine. En s'appuyant sur son expérience, il peut déterminer à la fois la forme et la position des sous-ensembles flous relatifs aux attributs de l'espace de représentation. Ce procédé a l'avantage de produire un partitionnement en parfaite adéquation avec le problème traité en bénéficiant de l'expérience déjà acquise. Les attributs sont ainsi transcrits sous forme de variables linguistiques et les modalités floues correspondent alors aux termes linguistiques.[11] Ce procédé a l'avantage de rendre la modélisation parfaitement interprétable par les experts du domaine. Cependant, il est coûteux en temps, en « homme », et doit être répété pour chaque nouveau problème à traiter, ce qui n'est pas toujours possible et va à l'encontre de notre objectif de *généricité*.

Cela nous amène directement à la deuxième origine qui consiste à extraire *automatiquement* les sous-ensembles flous à partir des données. Cette extraction peut se faire soit lors de prétraitements, soit au cours de la construction de l'arbre. Cependant, effectuer l'extraction lors de prétraitements est en contradiction avec la stratégie sur laquelle repose le processus de construction des arbres de décision. À chaque nœud correspond

11. Ce sont les hypothèses de travail faites dans [89, 172, 68, 36].

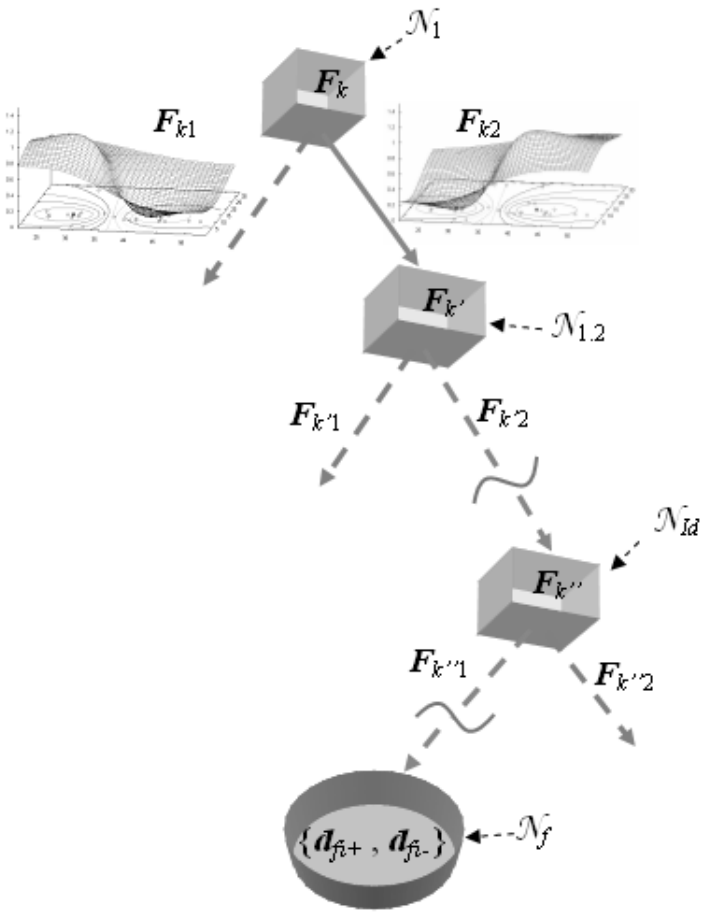

FIG. 3.16 – *Structure des arbres de décisions flous utilisés.*

un sous-problème différent qu'il faut résoudre. Il est donc naturel d'utiliser les outils les mieux adaptés pour traiter ce sous-problème. Ce principe correspond parfaitement à la nature des connaissances discriminantes que l'on veut extraire puisque nous avons vu que la pertinence de celles-ci dépend du contexte. On préférera donc une *extraction dynamique* des modalités floues au cours de la construction de l'arbre, localement à chaque nœud comme par exemple dans les approches de Marsala [121] ou encore de Hsu et Chiang [91] et plus récemment de Olaru et Wehenkel [130].[12] On notera aussi qu'il est possible de partir de sous-ensembles flous prédéterminés puis de les optimiser localement à chaque nœud [90, 88].

Après avoir déterminé quand extraire les modalités floues, il reste à établir com-

12. On remarquera d'ailleurs que c'était déjà la stratégie employée dans les arbres de décision classiques pour traiter les attributs continus [30, 143].

ment les extraire. Là encore, différents processus ont été mis en œuvre, s'appuyant sur la recherche de points de coupure [186, 187, 130], sur la théorie de la morphologie mathématique [121], etc. Dans notre étude, ce sont ces modalités floues qui représentent les connaissances discriminantes que l'on recherche. Elles doivent donc respecter les propriétés relatives à la discrimination et à la structure des données. Dans ce cadre, les *C-moyennes floues* sont particulièrement intéressantes et c'est le choix que nous avons adopté.

3.4.4.3 Les C-moyennes floues pour représenter des connaissances discriminantes

Les sous-ensembles flous issus des *C-moyennes floues* ont des propriétés particulières qui résultent de la façon dont sont déterminées les fonctions d'appartenance (cf. équation (3.4)) et notamment de la contrainte imposée aux degrés μ_{jc} dont la somme doit valoir 1 pour tout individu e_j (cf. équation (3.2)). La conséquence immédiate dont nous avons déjà parlé est le caractère relatif des centroïdes trouvés et des sous-ensembles flous dérivés (cf. paragraphe 3.3.1). Alors que cette propriété était particulièrement gênante pour l'extraction et la représentation de connaissances intrinsèques, elle devient ici très intéressante. En effet, des frontières implicites se forment automatiquement entre les différents regroupements. Ainsi, ce sont les propriétés de séparation de ces regroupements qui sont réellement modélisées. Cela apparaît clairement sur la figure 3.17 où l'algorithme est utilisé pour rechercher deux regroupements dans un espace à deux dimensions. La frontière entre ceux-ci est matérialisée par la ligne de niveau correspondant à un degré d'appartenance de $0,5$ à chacun des sous-ensembles flous.

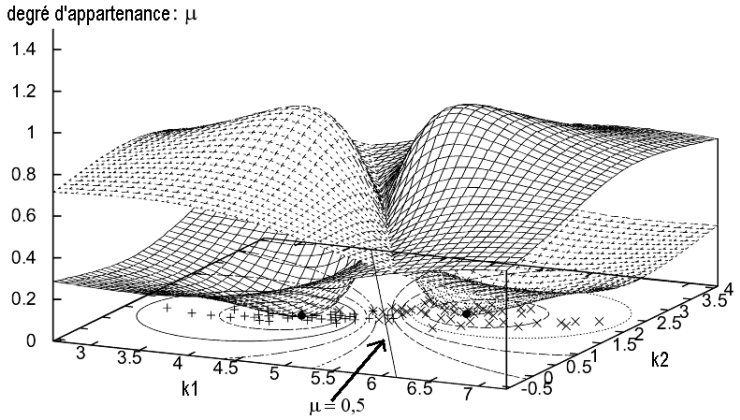

FIG. 3.17 – *Frontière discriminante résultant d'une modélisation par les C-moyennes floues.*

De plus, comme l'algorithme repose sur une fonction objectif visant à minimiser la distance intra-groupe, les centroïdes et donc les frontières de séparation qui en sont déduites sont automatiquement positionnées de façon stable par rapport à la distribution

« naturelle » (la structure) des données dans l'espace de représentation. De par leur aspect non supervisé, les *C-moyennes floues* mettent donc l'accent sur la recherche de frontières naturelles de séparation, ce qui va à la fois dans le sens de la robustesse mais aussi dans celui de l'interprétabilité.

Nous avons vu ci-dessus l'intérêt d'utiliser des sous-ensembles flous résultant des CMF pour trouver des frontières séparatrices inter-regroupements. Mais il ne faut pas oublier non plus de considérer la forme des fonctions d'appartenance lorsque l'on s'éloigne des centroïdes. En effet, nous avons vu que dans cette situation, celles-ci tendaient vers $1/C$ (où C est le nombre de regroupements cherchés) et sont donc non-convexes. Cette particularité a bien évidemment une incidence, à la fois sur le résultat de la modélisation mais aussi sur le processus de décision. Il est donc légitime de s'interroger sur la cohérence de la représentation par rapport à son utilisation. Nous allons voir dans le paragraphe suivant que dans le cadre du processus de partitionnement, et par extension pour la prise de décision, cette propriété s'interprète de façon tout à fait logique. [13]

À présent que nous avons déterminé un mode d'extraction et de représentation adapté pour la modélisation de connaissances discriminantes, il reste à voir comment l'intégrer dans le processus de construction de nos arbres de décision flous, notamment pour que la discrimination s'opère sur les classes du problème. C'est l'objet des deux paragraphes suivants.

3.4.4.4 Partitionner par les C-moyennes floues

Il y a plusieurs questions auxquelles il faut pouvoir répondre pour intégrer les *C-moyennes floues* dans le processus de construction de nos arbres de décision flou. La première était de déterminer quand extraire les sous-ensembles flous et nous avons vu que l'extraction dynamique, dans le contexte associé à chaque nœud, permettait d'être en accord à la fois avec le mode de fonctionnement des arbres mais aussi avec la nature des connaissances que l'on souhaite extraire. Un deuxième point à considérer est de savoir sur quelles données utiliser les CMF.

Partitionner sur quelles données ?

Récemment, dans [91], les auteurs ont utilisé les CMF pour l'élaboration d'arbres de classification flous. Ils effectuent l'extraction à chaque nœud sur les individus de chaque classe séparément. L'algorithme détermine ensuite le meilleur partitionnement parmi l'ensemble des classes avec une mesure de discrimination. Cette méthode offre de bons résultats mais elle ne correspond pas à notre point de vue sur la représentation des connaissances. En effet, en procédant ainsi, les CMF sont utilisées dans le but de décrire les classes de façon explicite, ce qui correspond davantage à une vision intrinsèque du

13. Le lecteur notera aussi qu'il existe une variante récente des CMF permettant de générer des sous-ensembles flous convexes [113] si l'on souhaite se ramener à une situation plus ordinaire.

problème qu'à une vision discriminante.[14] Or nous avons vu que les CMF ainsi que les sous-ensembles flous résultants étaient assez mal adaptés pour ce type de modélisation. Dans notre étude, nous souhaitons extraire des connaissances discriminantes robustes et dans ce contexte, les frontières discriminantes « naturelles » entre les classes sont sans doute les plus stables. C'est pourquoi nous utilisons les CMF sur l'ensemble des données de la base B localement à un nœud et indépendamment de leur classe, pour en extraire des frontières naturelles. La mesure de discrimination permettra ensuite de choisir celles qui sont les plus discriminantes par rapport aux classes, en fonction du sous-espace F_k dans lequel elles ont été extraites (cf. paragraphe suivant).

Après avoir déterminé quand et sur quelles données il fallait utiliser l'algorithme, il reste à voir comment le configurer, et notamment dans combien de dimensions faire les extractions (puisque nous avons vu que F_k pouvait être un sous-espace de dimension ≥ 1) et combien de regroupements chercher (paramètre C des CMF).

Partitionner dans quel espace ?

Classiquement, dans les arbres de décision flous, l'extraction des sous-ensembles flous se fait en une seule dimension ce qui facilite l'interprétabilité. Cependant, dans un espace de représentation numérique, il est évident que plusieurs attributs peuvent être corrélés et que cette corrélation peut aider pour faire la discrimination des classes. On peut reprendre l'exemple de la discrimination des lettres pour illustrer ce principe. Ainsi, pour distinguer le 'i' de la plupart des autres lettres, la présence du signe diacritique et sa forme sont deux caractères discriminants (cf. figure 3.15). Cependant, si la forme du diacritique est traduite dans l'espace de représentation par deux attributs (hauteur et largeur), seule l'utilisation des deux, ensemble, est réellement intéressante. Ce point a déjà été vérifié avec les arbres de décision classiques reposant sur une combinaison linéaire des attributs (cf. paragraphe 3.4.2.2). Les *C-moyennes floues* ont l'avantage de pouvoir fonctionner directement en x dimensions, $x \leq n$. Il semble donc intéressant d'essayer de tirer parti de cette propriété. Cependant, pour ne pas aller trop à l'encontre de la compacité et de l'interprétabilité, se restreindre à des partitionnements dans un sous-espace F_k à deux dimensions est une nécessité. Expérimentalement, cela permet un gain en performance important par rapport à l'utilisation d'une seule dimension et le résultat reste compact et relativement interprétable, pour le concepteur, puisque le partitionnement est facilement visualisable. Une autre solution aussi envisageable consiste à effectuer les partitionnements dans un sous-espace à x dimensions ($x \geq 2$) puis à projeter les sous-ensembles flous obtenus sur chacun des attributs concernés. Mais là encore, le nombre de modalités floues risque de croître de manière considérable. De plus, le mécanisme de projection provoque une perte d'information assez importante. Nous nous limitons donc à des partitionnements en deux dimensions F_{k1} et F_{k2} (cf. figure 3.18).

14. Même si le choix du partitionnement permet ensuite d'agir de façon discriminante. On se rapproche dans ce cas de l'élaboration de prototypes au sens de [147]. L'analogie a d'ailleurs été soulignée dans [121].

Partitionner en combien de parties ?

Abordons à présent le problème du nombre de regroupements C à rechercher, c'est-à-dire le nombre de modalités floues que l'on souhaite déterminer à chaque nœud. Une première solution consiste, comme dans le cas de l'algorithme des *C-moyennes possibilistes*, à utiliser un critère de partition comme $coefPart(U)$ (cf. paragraphe 3.3.4.1). Cependant, comme celui-ci nécessite de construire toutes les partitions de $Cmin$ à $Cmax$, le coût calculatoire peut vite devenir prohibitif. De plus, comme pour les arbres de décision classiques, un trop grand nombre de modalités au niveau d'un nœud entraîne une expansion horizontale de l'arbre qui peut conduire à une subdivision trop hâtive de la base d'apprentissage. Enfin, nous avons aussi vu que certaines mesures de discrimination comme l'entropie de Shanon étaient sensibles au nombre de modalités (cf. paragraphe 3.4.2.1). Il faudrait donc s'assurer que la mesure utilisée dans ce contexte soit indépendante. Pour l'ensemble de ces raisons, nous avons choisi d'opérer des partitionnements flous binaires au niveau des nœuds de l'arbre.

Une fois ces éléments déterminés, le processus d'extraction des sous-ensembles flous et de partitionnement associé se passe de la manière suivante.

Mécanisme de partitionnement

Les CMF sont utilisées localement à un nœud $\mathcal{N}_{Id}$ sur la sous-base d'apprentissage courante B, indépendamment des classes, et dans un sous-espace F_k à deux dimensions qu'il restera à déterminer (cf. paragraphe 3.4.4.5). Deux modalités floues F_{k1} et F_{k2} sont ainsi déduites (cf. figure 3.18) et deux nœuds fils vont être construits à leur tour en s'appuyant sur les sous-bases B_{k1} et B_{k2} résultant du partitionnement flou de B. Dans les arbres de décision classiques, le partitionnement produit des sous-bases disjointes, suivant la valeur e_j^k de l'individu pour l'attribut F_k. Dans nos arbres, ce processus doit être adapté pour prendre en compte les degrés d'appartenances des individus aux différentes modalités floues.

Rappelons que lors de la construction de l'arbre, la création d'un nœud correspond à un changement de contexte et nécessite la résolution d'une sous-tâche correspondant à la discrimination des classes représentées dans B. Dès lors, il est inutile (et même risqué) de travailler systématiquement sur la base d'apprentissage complète (B_{app}), même si les individus possèdent des degrés d'appartenance $\mu_{\mathcal{N}_{Id}}(e_j)$ différents à chacun des nœuds. En effet, l'ensemble des individus qui ont un degré d'appartenance faible à un nœud donné ne sont que peu pertinents dans ce contexte. Ils représentent donc à la fois un surcoût calculatoire mais peuvent aussi perturber le processus d'extraction des modalités floues par les CMF. Il y aura aussi des risques d'effets de bords sur le processus de sélection du sous-espace F_k pour la discrimination. D'une façon générale, un nombre important d'individus avec un degré faible ne peut être considéré comme négligeable. Pour limiter cet impact, les sous-bases B_{kl} sont constituées en utilisant un mécanisme d'α-coupe afin de ne conserver que les échantillons les plus pertinents pour

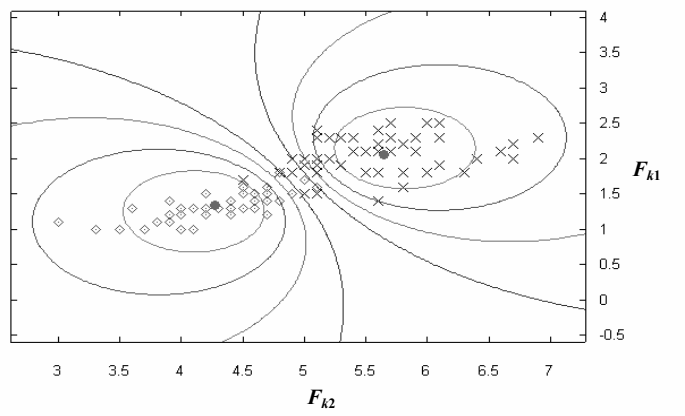

FIG. 3.18 – *Exemple de visualisation d'un partitionnement en deux dimensions par les C-moyennes floues. Le contexte de discrimination est représenté à la fois par les données locales au nœud (base B constituée ici d'exemples de 2 classes : '×' et '◇') et l'espace* $\{F_{k1} \times F_{k2}\}$ *permettant la discrimination. Les deux modalités floues sont représentées par les lignes de niveaux associées aux fonctions d'appartenance.*

chaque sous-base : [15]

$$B_{kl} = \{e_j \in B \mid \mu_{F_{kl}}(e_j) \geq \alpha\} \tag{3.25}$$

Avec une valeur de α réduite (inférieure à $0,5$ pour des partitions binaires avec les CMF), les individus situés près des frontières de deux regroupements sont dupliqués dans chacun des nœuds fils.

Ce mode de fonctionnement peut paraître surprenant puisqu'il duplique l'information qui localement est la plus sensible. Mais cela s'interprète assez facilement dans notre contexte. En effet, l'algorithme des CMF optimise le positionnement des centroïdes, et donc des frontières qui en résultent, de façon à séparer au mieux [16] les données du sous-problème B. Dès lors, si un individu se situe près d'une frontière, que ce soit près des centres ou loin de ceux-ci [17], cela signifie que l'espace dans lequel l'algorithme a effectué le partitionnement n'est pas adapté pour faire la discrimination de cet individu et l'attribuer de façon robuste à l'un des regroupements. En fait, le contexte local n'apporte pas d'informations suffisantes pour une prise de décision relative à cet individu.

La solution consistant à l'attribuer de façon plus ou moins arbitraire à l'un des regroupements ou même à l'éliminer pourrait devenir préjudiciable pour la suite. En effet, les sous-ensembles flous sont extraits dans un sous-espace F_k favorisant la discrimination des classes. Un choix est donc fait, en accord avec l'objectif local de discrimination, mais le résultat ne sera que rarement parfait. Il se fera donc nécessairement en faveur de certains exemples (ceux qui participent fortement à la détermination des centroïdes) et au détriment d'autres (ceux qui se trouvent près des frontières). Hors ces derniers sont

15. Ce processus est aussi utilisé dans [121].
16. Au sens de la fonction objectif.
17. Cela est équivalent ici de par la non convexité des fonctions d'appartenance des CMF.

peut-être porteurs d'informations intéressantes mais qui n'apparaîtront que dans un autre contexte (avec d'autres exemples et dans un autre sous-espace F_k). L'utilisation de fonctions d'appartenance non convexes permet de traduire cette forme d'imperfection et d'éviter les prises de décisions trop hâtives.

Une autre solution classique consiste à créer un nœud fils supplémentaire $\mathcal{N}_{Id.(L_k+1)}$ regroupant l'ensemble de ces individus problématiques. Mais là encore, cette solution n'est pas exempte de risques. En effet, il ne faut pas oublier que les CMF reposent très fortement sur les propriétés des individus dans l'espace de représentation et leurs interactions. Effectuer des regroupements arbitraires d'individus qui n'ont peut être que peu de points communs et les séparer au contraire de ceux avec qui ils peuvent représenter une information importante semble donc être une mauvaise solution.

C'est pour toutes ces raisons que les individus près des frontières inter-regroupements sont dupliqués dans chaque sous-base correspondante, laissant aux étages inférieurs de l'arbre la possibilité de les utiliser d'une meilleure façon.

Finalement, une fois les sous-bases $B_{kl},\ l = 1, 2$ définies, les degrés d'appartenance $\mu_{\mathcal{N}_{Id.l}}(e_j)$ des individus e_j sont adaptés pour prendre en compte à la fois leur appartenance au nœud père $\mathcal{N}_{Id}$ ainsi que celle à la modalité floue F_{kl}. Cette opération est effectuée par une t-norme $\top$ marquant la conjonction :

$$\mu_{\mathcal{N}_{Id.l}}(e_j) = \top(\mu_{\mathcal{N}_{Id}}(e_j), \mu_{F_{kl}}(e_j)) \tag{3.26}$$

où $\mu_{F_{kl}}(e_j)$ est le degré d'appartenance de e_j à F_{kl} que l'on déduit du résultat des CMF (cf. équation (3.4)). Le choix de la t-norme utilisée étant lié à l'exploitation de l'arbre, nous précisons celui-ci au paragraphe 4.3.3.

Jusqu'à présent, nous avons laissé en suspens le choix du sous-espace F_k au niveau d'un nœud, précisant juste qu'il était déterminé de façon à optimiser la discrimination des classes. Nous détaillons ce point ci-dessous.

3.4.4.5 Choix d'un sous-espace pour discriminer les classes

Dans les arbres de décision classiques, les mesures de discrimination sont utilisées afin de choisir le meilleur partitionnement au niveau d'un nœud dans le but de séparer les individus des différentes classes (cf. paragraphe 3.4.2.1). Le principe est similaire pour les arbres de décision flous. Ceux-ci se fondent sur une mesure adaptée pour prendre en compte l'imprécision relative à l'appartenance d'un individu e_j au nœud $\mathcal{N}_{Id}$ considéré. Plusieurs mesures ont ainsi été définies, à partir de mesures de discrimination classiques, de la notion d'entropie, etc. [162, 179, 121, 144, 172] (cf. [121, 118, 119, 172] pour un panorama de plusieurs d'entre elles).

Dans notre travail, nous avons utilisé comme mesure de discrimination la *mesure d'entropie étoile* [162] qui généralise l'entropie de Shannon. Celle-ci a été largement utilisée dans les arbres de décision flous, donnant de bons résultats [121, 89, 91]. Elle a aussi été comparée dans [118] à la mesure de Yuan et Shaw [179]. Les auteurs ont

ainsi montré que la mesure d'entropie étoile avait tendance à favoriser la création de feuilles pures dans l'arbre avec le risque d'obtenir un arbre déséquilibré. Au contraire, la mesure de Yuan et Shaw favorise la construction d'arbres équilibrés,en général plus gros (ils possèdent plus de feuilles).

En reprenant les notations de l'équation (3.20), l'entropie étoile est définie par :

$$H^*(P^*) = -\sum_{i=1}^{n} p_i^* \log p_i^* \tag{3.27}$$

Dans cette équation, la distribution de probabilité P est remplacée par une distribution de probabilité d'événement flou p^* [181]. Si cette distribution est déterminée à partir d'un ensemble de N individus, la probabilité d'événement flou $p^*(\omega_i)$ relative à l'obtention de la classe ω_i est :

$$p^*(\omega_i) = \sum_{j=1}^{N} \mu_{\omega_i}(e_j) p(e_j) \tag{3.28}$$

avec $p(e_j)$ la probabilité classique d'occurrence de l'individu e_j.

L'entropie conditionnelle floue relativement aux classes ω_{i+} et ω_{i-} correspondant au choix d'un attribut F_k est alors donnée par :

$$H_B^*(S|F_k) = -\sum_{l=1}^{L_k} p^*(F_{kl}) \sum_{i=\{i+,i-\}} p^*(\omega_i|F_{kl}) \log p^*(\omega_i|F_{kl}) \tag{3.29}$$

On peut alors calculer le gain d'information $G_B^*(F_k)$ associé au choix de F_k :

$$G_B^*(F_k) = H_B^*(S) - H_B^*(S|F_k) \tag{3.30}$$

Une fois le choix de la mesure de discrimination établi, il faut mettre en œuvre une procédure de sélection du sous-espace optimisant la discrimination des classes. La solution la plus simple consiste à calculer le gain d'information à partir du partitionnement résultant des CMF pour chaque sous-espace F_k possible. Cette solution est tout à fait envisageable si F_k est un sous-espace à une dimension (un attribut) et que la dimension totale de l'espace de représentation reste petite. Pour nos arbres, comme nous utilisons des sous-espaces de dimension 2 d'un espace qui peut être éventuellement très grand, cette recherche exhaustive peut s'avérer coûteuse en temps de calculs. Il est donc intéressant de mettre en place une stratégie de sélection alternative évitant une explosion combinatoire. Elle pourra ainsi s'appliquer à des sous-espaces de dimension supérieure à deux (si on souhaite sacrifier la lisibilité). Plusieurs types d'algorithmes sont utilisables et notre choix s'est porté sur l'utilisation d'un algorithme génétique [67].[18] Celui-ci repose sur l'optimisation d'une fonction objectif (la fonction de fitness) qui utilise directement le gain d'information (cf. équation (3.30)).

Finalement, pour terminer de décrire notre mécanisme d'apprentissage d'arbre de décision, il reste à voir le problème de la création des feuilles.

18. Les principes des algorithmes génétiques sont reportés dans l'annexe C.

3.4.4.6 Création des feuilles

Lorsqu'un critère d'arrêt est rencontré (nous utilisons ici les critères ordinaires tels que le nombre d'individus minimal à un nœud, ou encore la valeur du gain d'information), une feuille f est construite. Celle-ci est ici caractérisée par les degrés de représentativité d_{fi+} et d_{fi-} des classes ω_{i+} et ω_{i-}. La façon la plus simple et la plus courante pour déterminer ces degrés consiste à leur assigner la *probabilité conditionnelle floue* [181] d'obtenir la classe à laquelle ils correspondent sachant que les conditions permettant d'arriver jusqu'à la feuille f sont satisfaites :

$$d_{fi+} = P^*(\omega_{i+}|f) = \frac{\sum_{e_j \in \omega_{i+}} \mu_{\mathcal{N}_f}(e_j)}{\sum_{e_j} \mu_{\mathcal{N}_f}(e_j)} \tag{3.31}$$

où $\mu_{\mathcal{N}_f}(e_j)$ est le degré d'appartenance de e_j à la feuille f. Le degré d_{fi-} est calculé de façon similaire.

Si on considère l'arbre de décision flou uniquement dans le cadre de la modélisation discriminante, ces degrés constituent simplement une source d'information sur la qualité de la discrimination au niveau de chaque feuille. Cependant, dans le cadre du processus de décision associé à notre système, ces degrés prennent une importance toute particulière. Nous détaillons cet aspect dans le chapitre 4.

3.5 Bilan

Dans ce chapitre, nous avons décrit les fondements sur lesquels reposent la modélisation de notre approche de reconnaissance de formes. En partant de nos besoins, nous avons défini le type de connaissances à utiliser pour un usage donné, ainsi que les propriétés qu'elles devaient posséder. Dans notre modèle constitué de deux niveaux de modélisation associés au module de description intrinsèque par prototypes et au module de discrimination focalisée, nous avons identifié deux types fondamentaux de connaissances : les connaissances de nature intrinsèque aux classes utilisées pour décrire le problème de façon explicite et robuste et pour pouvoir fournir une base solide pour le mécanisme de décomposition ; les connaissances de nature discriminante entre classes pour pouvoir représenter les éléments locaux qui permettent de faire une distinction fine des classes.

Les connaissances intrinsèques et discriminantes sont extraites automatiquement à partir d'algorithmes de classification floue non supervisée permettant de s'appuyer sur la structure des données pour extraire des sous-ensembles flous robustes et explicites. Ainsi, les connaissances intrinsèques aux classes sont déduites de l'utilisation de l'algorithme des *C-moyennes possibilistes*. Les connaissances discriminantes entre classes sont quant à elles extraites et structurées par une nouvelle approche de construction d'arbres de décision flous. Ceux-ci ont été conçus spécifiquement pour répondre à nos objectifs et à notre vision des connaissances discriminantes. Ils reposent sur l'utilisation des *C-moyennes floues* afin d'extraire des frontières « naturelles » et robustes permettant la discrimination dans des sous-espaces à deux dimensions choisis par algorithme génétique.

Ces connaissances sont structurées entre elles par un mécanisme de décomposition permettant de passer du niveau intrinsèque au niveau discriminant. Les connaissances discriminantes sont de plus hiérarchisées en fonction de leur pertinence afin de rendre leur utilisation plus robuste.

Une fois que les modèles associés au niveau intrinsèque et au niveau discriminant ont été extraits, ceux-ci sont utilisés pour définir le mécanisme de décision du système. Afin d'obtenir une bonne compréhension de ce dernier, les connaissances des deux niveaux sont intégrées dans des systèmes d'inférence floue permettant de faire un lien explicite entre celles-ci et les classes. Le chapitre suivant décrit en détails le processus de décision qui en découle.

Chapitre 4

Exploitation du modèle pour la décision : combinaison de systèmes d'inférence floue

Grâce aux mécanismes d'apprentissage vus dans le chapitre 3, il nous est possible de générer automatiquement une modélisation pour un problème donné. Cette modélisation, qui repose sur la structuration de connaissances intrinsèques et discriminantes formalisées de façon homogène par des sous-ensembles flous, est exploitée pendant la phase de généralisation pour l'identification de formes inconnues. Elle est pour cela associée à une fonction de décision.

Celle-ci se matérialise par deux éléments : la mise en relation explicite des connaissances du modèle avec les différentes classes et un processus qui permet, à partir de ces relations et d'une entrée e, de classer cette dernière. Les connaissances de la modélisation étant représentées par des sous-ensembles flous, nous adoptons ici un mécanisme de décision à base de règles floues. Celles-ci sont intégrées dans des systèmes d'inférence floue qui sont combinés de façon à bénéficier au maximum de la structuration de la modélisation.

Dans ce chapitre, nous présentons tout d'abord la structure générale sur laquelle repose le mécanisme de décision. Nous décrivons ensuite en détails le formalisme à base de systèmes d'inférence floue utilisé pour associer les connaissances du modèle aux classes. Nous terminons enfin en explicitant le processus de décision dans son intégralité.

4.1 Principe du mécanisme de décision et complémentarité des connaissances pour la décision

Dans la modélisation que nous avons décrite, le niveau de modélisation discriminante regroupe l'essentiel des connaissances utiles pour la classification. En effet, il bénéficie directement des connaissances intrinsèques par le biais de la décomposition guidée par les données et intègre les connaissances discriminantes permettant de distinguer les classes. Il est donc associé à un module de décision appelé *module de classification principale*.

Cependant, en plus du mécanisme de décomposition, la modélisation intrinsèque des classes fournit sa propre vision du problème en utilisant notamment des connaissances plus générales et plus stables que les connaissances discriminantes qui sont particulièrement sensibles à cause de leur caractère contextuel. Or nous avons vu dans le paragraphe 2.2.1 que l'utilisation de points de vue différents était un moyen efficace pour accroître les performances d'un système de reconnaissance. Il est donc intéressant de s'appuyer sur le niveau intrinsèque non seulement pour la structuration qu'il apporte mais aussi pour les informations qu'il possède et qui peuvent s'avérer utiles pour la classification.

En partant de ce principe, le mécanisme de décision du système *Mélidis* (cf. figure 4.1) s'articule autour des trois modules de classification comme évoqué au paragraphe 1.4.3.3 : le module de *pré-classification*, le module de *classification principale* et le module de *classification finale*.

Comme les connaissances de la modélisation sont représentées par des sous-ensembles flous, nous appliquons les principes vus dans le paragraphe 2.1 pour formaliser l'ensemble de façon homogène par des systèmes d'inférence floue (SIF). Ceux-ci permettent de faire un lien explicite entre les connaissances de la modélisation et les classes par des règles de décision plus facilement compréhensibles par le concepteur. Comme le niveau intrinsèque travaille sur l'ensemble du problème modélisé, il est associé à un seul SIF. En revanche, comme chaque modèle discriminant opère de façon indépendante sur un sous-problème, chacun d'eux est associé à un SIF. Les résultats de ceux-ci doivent donc être combinés pour fournir une sortie homogène à la pré-classification afin que les deux puissent à leur tour être fusionnés pour la classification finale.

Dans le paragraphe suivant, nous décrivons chacun de ces SIF de façon détaillée. Le processus de décision complet se rapportant à la figure 4.1 est ensuite expliqué dans le paragraphe 4.3.

4.2 Intégration des connaissances de la modélisation dans des systèmes d'inférence floue

À la fin de l'apprentissage de la modélisation, les connaissances extraites sont intégrées automatiquement dans des règles de décision. Leur formalisme est décrit ci-dessous.

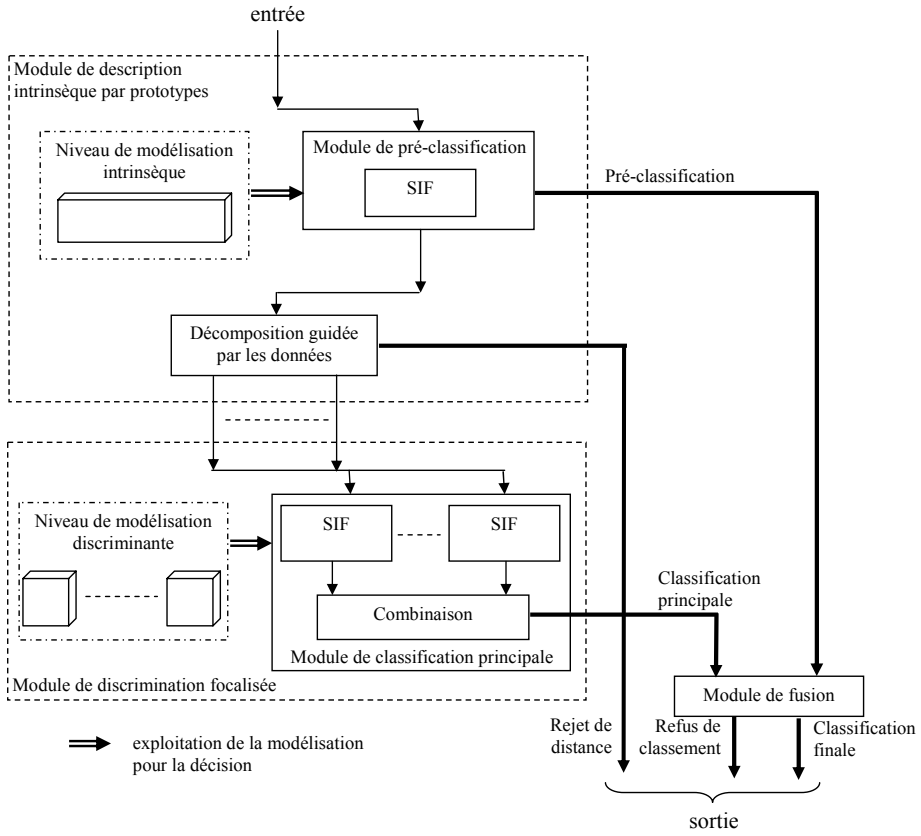

FIG. 4.1 – *Principe du mécanisme de décision du système Mélidis.*

4.2.1 Module de pré-classification

À partir de la modélisation du niveau intrinsèque, un ensemble de règles de décision relatives au problème sont déduites automatiquement. Pour cela, il suffit de définir un lien explicite entre chaque modèle $MI(\omega_i)$ et la classe ω_i qu'il décrit par une règle floue R_{ω_i}.

Comme nous l'avons montré dans le paragraphe 2.1.4, une règle peut prendre différentes formes. Nous proposons ici trois formalismes.

4.2.1.1 Règles à une conclusion

La façon la plus intuitive et la plus simple pour construire les règles associées aux connaissances intrinsèques est la suivante :

$$\boldsymbol{R_{\omega_i}} : \textbf{SI } e \text{ correspond à } MI(\omega_i) \textbf{ ALORS} \text{ la classe est } \omega_i \tag{4.1}$$

Dans cette règle, la prémisse correspond à l'adéquation d'un individu e au modèle intrinsèque $MI(\omega_i)$, c'est-à-dire à au moins une des sous-classes de ω_i. Elle peut être détaillée de la façon suivante :

$$\textbf{SI } e \text{ est } F_{\omega_i}^1 \textbf{ ou ... ou } e \text{ est } F_{\omega_i}^l \textbf{ ou ... ou } e \text{ est } F_{\omega_i}^{L_i} \tag{4.2}$$

L'inconvénient de cette forme de règle est qu'elle ne rend pas du tout compte de la pertinence de la description des modèles intrinsèques aux classes. En effet, ils n'auront vraisemblablement pas tous la même robustesse. On peut alors modifier la règle pour la pondérer par un degré d_i :

$$\boldsymbol{R_{\omega_i}} : \textbf{SI } e_j \text{ correspond à } MI(\omega_i) \textbf{ ALORS la classe est } \omega_i \text{ avec un degré } d_i \tag{4.3}$$

La façon la plus simple pour définir d_i consiste à lui affecter la probabilité conditionnelle floue : $P^*(\omega_i|MI(\omega_i))$ déterminée sur la base d'apprentissage.

4.2.1.2 Règles à conclusions multiples

La forme des règles décrites dans le paragraphe précédent ne correspond que partiellement à la réalité. Elle considère en effet que chaque modèle ne décrit que sa classe, de façon exclusive. Or il est évident que les modèles intrinsèques, et plus précisément leurs sous-classes, vont se recouvrir dans certaines zones de l'espace de représentation (sauf peut-être pour les cas les plus simples). Ceci est d'autant plus vrai que les modèles sont déterminés indépendamment les uns des autres. Pour obtenir une description plus complète de ce que les modèles intrinsèques englobent, il convient de modifier les conséquences des règles de façon à les rendre multiples :

$$\begin{array}{lll} \boldsymbol{R_{\omega_i}} : & \textbf{SI } e \text{ correspond à } MI(\omega_i) & \\ & \textbf{ALORS} & \text{la classe est } \omega_1 \text{ avec le degré } d_{i1} \textbf{ et ... et} \\ & & \text{la classe est } \omega_s \text{ avec le degré } d_{is} \textbf{ et ... et} \\ & & \text{la classe est } \omega_{\mathcal{S}} \text{ avec le degré } d_{i\mathcal{S}} \end{array} \tag{4.4}$$

où les d_{is} sont toujours les probabilités conditionnelles floues associées aux classes. On remarquera alors que la première règle (règle (4.1)) se retrouve aisément en fixant $d_{ii} = 1$ et $d_{is} = 0$ pour $s \neq i$.

4.2.1.3 Règles optimisées pour la classification

Malgré ces différentes améliorations, les règles précédentes conservent un inconvénient majeur dans le cadre de notre processus de décision. Elles permettent de représenter le pouvoir de description des modèles intrinsèques de façon précise mais elles ne permettent pas de prendre en compte leurs interactions pour la discrimination des classes. Or dans une approche de reconnaissance de formes, les performances restent un objectif majeur. On a donc tout intérêt d'essayer de tirer parti au maximum des modèles intrinsèques pour maximiser les performances de la pré-classification en vue de la fusion et de la classification finale.

Pour cela, on considère le SIF comme une fonction de décision qui à une entrée e associe un score s_i^1 pour chaque classe ω_i. L'objectif consiste ensuite à optimiser cette fonction de façon à ce qu'elle produise un score $s_i^1 = 1$ si ω_i est la vrai classe de e, et de $s_i^1 = 0$ sinon. Les règles peuvent alors être exprimées sous la forme suivante, en reprenant le formalisme des systèmes de type Takagi-Sugeno d'ordre 0 (cf. paragraphe 2.1.4) :

$$\begin{array}{lll} \boldsymbol{R_{\omega_i}} : & \textbf{SI } e \text{ correspond à } MI(\omega_i) & \\ & \textbf{ALORS} \quad s_1^1 = a_{i1} \textbf{ et } \ldots \textbf{ et} & \\ & \qquad\qquad\quad s_s^1 = a_{is} \textbf{ et } \ldots \textbf{ et} & \\ & \qquad\qquad\quad s_{\mathcal{S}}^1 = a_{i\mathcal{S}} & \end{array} \tag{4.5}$$

où les a_{is} sont des constantes qui mesurent la participation de la règle dans le résultat vis à vis de chaque classe. Elles peuvent être déterminées directement par une méthode d'optimisation comme la méthode de la pseudo-inverse ou encore initialisées (en prenant par exemple $a_{ii} = 1$ et $a_{is} = 0$ pour $s \neq i$) puis optimisées par des méthodes s'appuyant sur une descente de gradient [66].

Grâce à ces possibilités d'optimisation des conclusions, ce formalisme de règle offre à notre avis un très bon compromis entre lisibilité et performances. En effet, même si les conclusions perdent en lisibilité, l'essentiel de l'interprétabilité du système est conservée puisqu'elle réside dans les modèles intrinsèques eux-mêmes. De plus, comme nous l'avons montré dans le paragraphe 2.1.4, ce type de SIF simplifie le mécanisme d'inférence et permet de le rendre moins coûteux (en mémoire et en temps de calcul), ce qui dans la perspective de portage sur des machines aux ressources limitées peut devenir très intéressant. C'est donc ce type de règle que nous avons choisi d'utiliser dans notre approche.

NOTE:
On remarquera que quelle que soit la forme des règles utilisées, il est encore possible d'utiliser une description plus précise des relations entre les modèles intrinsèques et les classes en s'appuyant sur la décomposition en sous-classes. Dans ce cas, le SIF ne contiendra plus une règle par modèle intrinsèque $MI(\omega_i)$ (et donc par classe) mais une pour chaque prototype flou $F_{\omega_i}^l$ (pour chaque sous-classe) les composant.[1]

4.2.2 Module de classification principale

Afin de conserver un formalisme homogène avec le module de pré-classification, le niveau de modélisation discriminant est à son tour utilisé pour générer automatiquement un ensemble de systèmes d'inférence floue. Chaque arbre est formalisé par un SIF qui se déduit naturellement de sa structure. La formalisation des arbres sous la forme d'une base de règles de décision n'est pas récente [142] et a largement été utilisée pour la manipulabilité et l'interprétabilité qu'elle offre [89, 121, 179].

1. Rappelons en effet que le modèle intrinsèque d'une classe est constitué de la réunion d'un ensemble de prototypes flous caractérisant les sous-classes.

Pour un arbre donné, une règle floue R_f est extraite pour chaque chemin qui va de la racine à une feuille $\mathcal{N}_f$:

$$\begin{aligned} \boldsymbol{R_f}: \quad & \textbf{SI } e \text{ satisfait les conditions conduisant à } \mathcal{N}_f \\ & \textbf{ALORS} \quad \text{sa classe est } \omega_{i+} \text{ avec le degré } d_{fi+} \textbf{ et} \\ & \qquad\quad \text{sa classe est } \omega_{i-} \text{ avec le degré } d_{fi-} \end{aligned} \tag{4.6}$$

La prémisse correspond à l'adéquation d'un individu e à la feuille $\mathcal{N}_f$ et représente donc la conjonction des conditions sur le chemin de la racine $\mathcal{N}_1$ à la feuille $\mathcal{N}_f$. Elle peut donc être détaillée de la façon suivante, en s'appuyant sur le schéma de la figure 4.2 :[2]

$$\textbf{SI } e \text{ est } F_{1l} \textbf{ et } \ldots \textbf{ et } e \text{ est } F_{Id.l'} \textbf{ et } \ldots \tag{4.7}$$

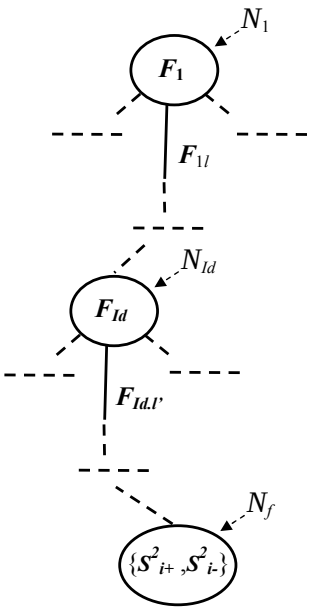

FIG. 4.2 – *Chaque chemin de l'arbre qui va de la racine à une feuille est traduit par une règle.*

La conclusion utilise quant à elle les *degrés de représentativité* des classes d_{fi+} et d_{fi-} comme nous les avons définis dans l'équation (3.31).

Comme pour le module de pré-classification, les règles peuvent aussi s'exprimer en utilisant le formalisme des règles de Takagi-Sugeno d'ordre 0 de façon à les optimiser

2. Cette notation est simplifiée dans le sens où il serait plus convenable de remplacer l'individu e (et donc le vecteur de caractéristique associé) par son sous-vecteur de caractéristiques correspondant à l'espace de représentation utilisé à chaque nœud.

pour la reconnaissance. Elles prennent alors la forme suivante :

$$\begin{array}{ll} \boldsymbol{R_f}: & \textbf{SI } e \text{ satisfait les conditions conduisant à } \mathcal{N}_f \\ & \textbf{ALORS} \quad s^2_{i+} = b_{fi+} \textbf{ et} \\ & \qquad\quad s^2_{i-} = b_{fi-}. \end{array} \tag{4.8}$$

Là encore, c'est le formalisme que nous adoptons.

Nous allons à présent voir comment ces SIF sont exploités et leurs résultats combinés.

4.3 Processus de décision pour la classification

Une fois les SIF des modules de pré-classification et de classification obtenus, ceux-ci peuvent être exploités en phase de généralisation pour déterminer à quelle classe appartient un individu e dont l'étiquette est inconnue.

Le processus de décision se déroule en quatre étapes (cf. figure 4.1). Le module de pré-classification détermine, en se basant sur les connaissances intrinsèques, quels SIF vont être exploités au deuxième niveau (cf. paragraphe 4.3.1). Il délivre aussi un score s^1_i à chaque classe ω_i, l'ensemble de ceux-ci formant la pré-classification (cf. paragraphe 4.3.2). Le module de classification principale fournit à son tour un score s^2_i pour chaque classe, résultant de la combinaison des SIF précédemment sélectionnés (cf. paragraphe 4.3.3). Finalement, le résultat de la pré-classification et de la classification principale sont fusionnés de façon à obtenir la classification finale (cf. paragraphe 4.3.4).

4.3.1 Sélection des connaissances discriminantes et rejet de distance

La structuration hiérarchique sur deux niveaux du classifieur permet d'opérer une sélection des connaissances discriminantes à mettre en jeu lors du processus de décision. Elle assure donc que les éléments pris en compte pour la classification sont en adéquation avec la forme présentée en entrée. Elle limite ainsi les perturbations qui pourraient survenir à cause des interactions trop complexes entre les connaissances. Cela rend le processus de décision plus robuste tout en favorisant son intelligibilité et sa vitesse d'exécution.

Cette phase de sélection s'appuie sur les connaissances des modèles intrinsèques afin de déterminer quelles sont les classes auxquelles l'individu présenté en entrée s'apparente le plus. Pour cela, le mécanisme utilisé lors de l'apprentissage pour la décomposition en sous-problèmes est à nouveau employé (cf. paragraphe 3.3.5). Ainsi, en reprenant l'équation (3.19), un individu e sera propagé au niveau du SIF du deuxième niveau correspondant à la classe ω_i seulement si :

$$\mu_{MI(\omega_i)}(e) \geq \frac{\max_{s=1,\ldots,\mathcal{S}}(\mu_{MI(\omega_s)}(e))}{\alpha} \tag{4.9}$$

avec une valeur de α identique à celle utilisée lors de l'apprentissage. L'adéquation de e au modèle de la classe $MI(\omega_i)$ est toujours déterminé par l'équation (3.17). Finalement,

on note $\mathcal{I}$ l'ensemble des SIF (le numéro de la classe à laquelle ils correspondent) du module de classification principale qui sont ainsi activés :

$$\mathcal{I} = \left\{ i,\ i = 1, \ldots, \mathcal{S} \mid \mu_{MI(\omega_i)}(e) \geq \frac{\max_{s=1,\ldots,\mathcal{S}}(\mu_{MI(\omega_s)}(e))}{\alpha} \right\}$$

Comme cette sélection s'appuie sur un mécanisme de seuillage relatif défini par α, une entrée activera systématiquement au moins un SIF ($\mathcal{I} \neq \emptyset$). La conséquence directe de cette manière de procéder est que si e est une entrée invalide, c'est-à-dire ne correspondant à aucune classe d'un point de vue intrinsèque, celle-ci sera malgré tout dirigée vers le module de classification principale pour être identifiée, produisant nécessairement une erreur. Une solution intéressante pour pallier ce problème, consiste à exploiter les propriétés des connaissances intrinsèques aux classes afin de mettre en œuvre un mécanisme de rejet de distance.

Pour introduire celui-ci, une possibilité consiste à déterminer (*a priori* ou par apprentissage) des seuils α_i pour chaque classe.[3] À partir de ceux-ci, on considère qu'un individu e doit être rejeté si :

$$\forall i = 1, \ldots, \mathcal{S} \quad \mu_{MI(\omega_i)}(e) < \alpha_i \tag{4.10}$$

En effet, si l'adéquation $\mu_{MI(\omega_i)}(e)$ d'un individu à un modèle de classe est inférieure au seuil α_i correspondant, e peut être considéré comme trop différent de $MI(\omega_i)$ pour pouvoir appartenir à ω_i. Si c'est le cas pour toutes les classes, l'individu est rejeté et non classé. Les seuils α_i ainsi définis pourront aussi remplacer efficacement le seuillage relatif dans l'équation (4.9) employé pour la sélection des SIF. Dans ce cas, un SIF du module de classification principale sera ajouté à $\mathcal{I}$ si :

$$\mu_{MI(\omega_i)}(e) \geq \alpha_i$$

4.3.2 Pré-classification

Suite au processus de sélection, si la forme n'est pas rejetée, le module de pré-classification produit un vecteur de sortie en exploitant le SIF généré. Nous avons vu qu'il existait différentes façons de formuler les règles (cf. paragraphe 4.2.1). Ici, nous ne présentons que le mécanisme aux règles de la forme (4.5) utilisées pendant les expérimentations.[4]

La sortie est représentée par un ensemble de scores $s_i^1, i = 1, \ldots, \mathcal{S}$ relatifs à chaque classe. Ils sont obtenus à partir de l'inférence *somme-produit* définie par l'équation (2.22) :

$$s_i^1 = \frac{\sum_{s=1}^{\mathcal{S}} \mu_{MI(\omega_s)}(e) \cdot a_{si}}{\sum_{j=1}^{\mathcal{S}} \mu_{MI(\omega_j)}(e)} \tag{4.11}$$

3. Les moyens d'obtenir ces seuils sont toujours à l'étude actuellement et nous ne les reportons donc pas ici.

4. Les règles de la forme (4.3) et (4.4) peuvent être inférées par différents mécanismes comme ceux présentés dans le paragraphe 2.1.3.3.

Ce choix *a priori* est motivé par l'expérience qui a montré qu'il permettait d'obtenir des performances intéressantes. C'est en effet le mécanisme d'inférence utilisé dans *ResifCar* et, comme nous l'avons souligné auparavant dans les paragraphes 2.1.4 et 4.2.1.3, il permet d'optimiser les conclusions du SIF par des méthodes numériques simples. Ce choix va donc le sens de notre objectif de performances. Cependant, d'autres mécanismes peuvent aussi être envisagés et leur étude fait partie des perspectives.

4.3.3 Classification principale

Une fois la sélection des SIF effectuée par le module de pré-classification, l'individu e (non rejeté) est présenté à chacun des SIF de $\mathcal{I}$.

Quelle que soit la forme des règles utilisées parmi celles présentées dans le paragraphe 4.2.2, la prémisse d'une règle R_f est obtenue en utilisant la t-norme *produit* pour représenter la conjonction des conditions le long de la branche menant à la feuille f :

$$\mu_{\mathcal{N}_f}(e) = \mu_{F_{1l}}(e) \cdot \ \ldots \ \cdot \mu_{F_{Id.l'}}(e) \cdot \ \ldots \tag{4.12}$$

Ce choix va de pair avec celui du mode d'inférence. En effet de par la nature même des fonctions d'appartenance des CMF et par le choix du produit comme opérateur de conjonction, chaque individu est « partagé » entre les différentes branches de l'arbre. Ainsi, si on considère $\mu_{\mathcal{N}_{Id}}(e_j)$ non plus comme l'appartenance de e_j à $\mathcal{N}_{Id}$ mais comme sa représentativité localement à ce nœud[5], ce partage se traduit par le fait que pour toutes les feuilles f de l'arbre, $\sum_f \mu_{\mathcal{N}_f}(e_j) = 1$. Il convient alors, lors de la classification, de prendre en compte ce partage de l'individu entre les différentes branches et de cumuler leurs effets. Dans ce cadre, l'utilisation d'une inférence *somme-produit* sur les règles (4.8) est donc tout à fait adaptée. De plus, elle procède à une normalisation implicite des scores et permet ainsi de les rendre homogène pour chacun des SIF du niveau discriminant, facilitant leur fusion. Enfin, ce mécanisme facilite à nouveau les optimisations pour l'objectif de performances. Bien entendu, d'autres mécanismes d'inférence et opérateurs de conjonction peuvent aussi être utilisés. Ainsi, les règles (4.6) peuvent être exploitées de manière classique par une inférence type *max-produit* (cf. paragraphe 2.1.4, équation (2.21)) ou encore par la méthode de classification de Marsala [121]. Il s'agit là encore de perspectives à étudier.

Le score associé à la classe ω_{i+} se calcule finalement de la manière suivante :

$$s^2_{i+}(e) = \frac{\sum_{f=1}^{F} \mu_{\mathcal{N}_f}(e) \cdot b_{fi+}}{\sum_{g=1}^{F} \mu_{\mathcal{N}_g}(e)} \tag{4.13}$$

où F et le nombre de règles du SIF (feuilles de l'arbre). Le score s^2_{i-} s'obtient de façon similaire.

5. Pour tout individu, sa représentativité vaut 1 à la racine de l'arbre.

Pour obtenir le vecteur de sortie du module de classification principale, il faut combiner les résultats de chacun des SIF du module. Comme tous n'ont pas été activés suite à la phase de sélection, les scores correspondants sont fixés de la manières suivante :

$$\forall j \notin \mathcal{I}\ s^2_{j+} = 0\ et\ s^2_{j-} = 1$$

La combinaison peut alors s'effectuer de différentes manières. Par exemple, on peut combiner à la fois les scores associés aux classes du problèmes (s^2_{i+}) et les scores associés aux « autres classes » (s^2_{i-}) avec avec la règles de Dempster-Shafer reposant sur la théorie de l'évidence [155]. C'est le choix fait par exemple dans [121] pour gérer les forêts d'arbres de décision flous. Mais ici, la combinaison s'effectue simplement en composant directement le vecteur de sortie avec les scores s^2_{i+} ($\{s^2_{1+}, \ldots, s^2_{i+}, \ldots, s^2_{\mathcal{S}+}\}$). Ce principe est simple et s'est avéré efficace à l'usage. Cependant, d'autres mécanismes peuvent aussi être envisagés.

On notera enfin qu'il est possible d'introduire à ce niveau des mécanismes permettant de fiabiliser le processus de décision. Un exemple simple repose sur la non-contradiction des SIF. Par exemple, si deux SIF associés aux classes ω_i et ω_j fournissent respectivement les scores s^2_{i+} et s^2_{j+} et que ceux-ci sont élevés (supérieurs à un seuil à déterminer), on peut considérer qu'ils se contredisent (les deux classes ω_i et ω_j sont en concurrence). D'une façon similaire, si aucun score $s^2_{i+},\ i = 1, \ldots, \mathcal{S}$ n'est supérieur à un seuil, alors aucune classe ne semble vraiment être privilégiée. Dans ces deux cas, on peut décider soit de ne pas classer la forme (refus de classement), soit de faire confiance au module de pré-classification. Dans ce dernier cas, étant donné le mode de fusion adopté (cf. paragraphe 4.3.4), les scores $s^2_{i+},\ i = 1, \ldots, \mathcal{S}$ sont fixés à 1.

4.3.4 Classification finale

Le module de classification finale utilise en entrée les scores $s^1_i,\ i = 1, \ldots, \mathcal{S}$ issus du module de pré-classification ainsi que les scores $s^2_{i+},\ i = 1, \ldots, \mathcal{S}$ issus du module de classification principale. La décision finale repose sur la fusion de ces deux informations qui sont complémentaires pour chaque classe étant donné leurs origines différentes. La première chose à faire pour pouvoir les fusionner est de s'assurer que ces deux vecteurs sont bien normalisés. Plusieurs possibilités sont bien sûr envisageables mais expérimentalement, le procédé qui consiste à normaliser par la somme des scores (équation (2.25)) semble donner les résultats les plus stables.

Pour le choix de la méthode de fusion en elle-même, celle-ci doit tenir compte des aspects suivants :

- s^1_i est une mesure stable mais peu discriminante par rapport à s^2_{i+} qui est plus précise mais aussi plus instable aux abords des frontières. Il semble donc difficile *a priori* de privilégier, lors de la fusion, une information plutôt qu'une autre ;
- s^1_i et s^2_{i+} mesurent toutes les deux l'adéquation à la classe ω_i mais en se basant sur des connaissances complémentaires (intrinsèques et discriminantes). Pour que la classe ω_i reste un choix privilégié pour la classification finale, les scores doivent

être forts en même temps. Si l'un des deux est faible, la classe correspondante doit être écartée au profit de celles qui sont plus favorables ;
- le résultat de la combinaison doit rendre compte de la participation des deux mesures de façon précise afin d'éviter les comportements ératiques.

Parmi les nombreuses méthodes de combinaison couramment utilisées [19, 127, 164, 25, 95, 7], le *produit* est un opérateur de fusion intéressant puisqu'il a un comportement proche de celui désiré ici. En effet, en reprenant la caractérisation des opérateurs de fusion faites dans [25], le produit est : commutatif, indépendant du contexte, conjonctif, continu. De plus, il a l'avantage d'être peu coûteux, à l'inverse de méthodes comme la règle de combinaison de Dempster-Shafer ou encore de la méthode du BKS, ce qui va toujours dans le sens de nos objectifs. Il permet aussi de conserver une bonne séparation entre le module de fusion et les deux autres modules, ce qui permet de faciliter les optimisations de ces derniers. Cela serait plus difficile avec une fusion résultant d'un apprentissage automatique comme par exemple une solution utilisant un réseau de neurones. L'inconvénient de l'opérateur produit est sa sensibilité par rapport aux scores qui lui sont fournis : si le score pour une classe est faible, celle-ci est irrémédiablement exclue de la décision (cf. paragraphe 2.2.2.4) [8]. Ici, grâce à l'utilisation des connaissances intrinsèques en premier lieu et à la sélection des connaissances discriminantes ensuite, ces effets sont limités. Nous avons donc opté pour ce mode de combinaison.

Une fois le produit choisi comme opérateur de fusion, la décision finale est représentée par le vecteur de sortie $s = \{s_1, \ldots, s_{\mathcal{S}}\}$ défini par :

$$\forall i = 1, \ldots, \mathcal{S} \quad s_i = s_i^1 \cdot s_{i+}^2 \tag{4.14}$$

Finalement, la classe attribuée à e est celle pour laquelle la valeur de s_i est maximale.

À la suite de l'obtention du vecteur s, il est à nouveau possible de fiabiliser les réponses du système en refusant de classer une entrée qui provoque une décision trop ambiguë. Comme pour les mécanismes de rejet de distance, le refus de classement est encore à l'étude et nous ne donnons ici que quelques éléments de réponse. Le principe est le suivant. Soit ω_i et ω_j les deux classes ayant les scores s_i et s_j les plus élevés pour un individu e. Si ceux-ci sont trop « proches » l'un de l'autre, c'est-à-dire si :

$$\|s_i - s_j\| < \theta$$

avec θ à déterminer, alors le classifieur refuse de classer e pour signifier que le risque d'erreur de confusion est trop important.

4.4 Bilan

Dans ce chapitre, nous avons présenté le mécanisme de décision associé à notre système. Celui-ci utilise directement les connaissances des niveaux de modélisation intrinsèque et discriminant et les intègre dans des règles de décision floues. Celles-ci ont

été conçues de façon à optimiser les performances en terme de taux de reconnaissance. Il s'agit d'un choix guidé par le cadre applicatif lié à la reconnaissance de formes et orienté vers la phase d'exploitation. Pour l'étude des différents modules du système et leur optimisation, nous avons proposé des formalismes intermédiaires plus facilement interprétables.

Le processus de décision a été organisé de façon à bénéficier au mieux de la structuration de la modélisation. Il repose sur trois modules de classification. Le module de pré-classification fournit un premier ensemble de scores reflétant la correspondance entre la forme et les différentes classes du point de vue de ses caractéristiques intrinsèques. En s'appuyant sur celles-ci, ce module sélectionne aussi les connaissances discriminantes qui seront exploitées. Cette sélection permet à la fois d'accélérer les traitements et de les rendre plus robustes et fiables. Le module de classification discriminante exploite ensuite les connaissances sélectionnées pour fournir un deuxième ensemble de scores. Finalement, ceux-ci sont fusionnés par l'opérateur produit pour produire la classification finale.

Il reste des points à approfondir dans cette partie relative au mécanisme de décision. Il reste notamment l'intégration des mécanismes de rejet de distance et de « refus de classement » à ajouter pour pouvoir améliorer la fiabilité des réponses. Le choix des modes de combinaison et de fusion sont aussi à étudier plus en détails. Il serait en effet intéressant d'essayer de bénéficier un peu plus des informations disponibles (comme les scores s^2_{i-}) mais aussi d'essayer de se placer entièrement dans le contexte du raisonnement approximatif. L'introduction de la logique possibiliste ou encore d'éléments de la théorie de l'évidence sont des voies de recherche à privilégier.

Cependant, avec la configuration présentée, le système obtient déjà des performances intéressantes permettant de valider les différents modules ainsi que la façon dont ils sont combinés. C'est ce que nous présentons au chapitre suivant.

Chapitre 5

Expérimentations

Au cours de ces travaux, un ensemble d'expérimentations a été conduit afin de valider les propriétés du système *Mélidis*. Celles qui sont reportées ici ont deux objectifs essentiels :

- montrer les performances brutes du classifieur par rapport aux objectifs principaux que nous nous sommes fixés (notamment en terme de généricité, de performances et de compacité) ;
- valider les différents modules du système ainsi que leurs apports relatifs dans l'ensemble.

Pour effectuer cette validation et notamment évaluer la généricité du système, nous nous sommes appuyé sur la résolution de plusieurs problèmes. Nous avons travaillé d'une part sur des benchmarks classiques couramment utilisés en classification, et d'autre part sur des problèmes plus complexes et spécifiques liés à la reconnaissance d'écriture manuscrite qui est une des origines de l'approche.

Dans ce chapitre, nous présentons dans un premier temps l'ensemble des bases utilisées pour les expérimentations. Nous décrivons ensuite les performances générales du classifieur en donnant quelques éléments comparatifs avec d'autres méthodes de classification et de reconnaissance. Nous terminons enfin en détaillant les apports de chacun des modules ce qui permet de valider la structuration hiérarchique.

5.1 Benchmarks et bases utilisées pour les expérimentations

Afin de valider la propriété de généricité, il était essentiel de faire des expérimentations sur plusieurs problèmes de complexité variable. Cette complexité peut se traduire de différentes façons. Il peut s'agir de difficultés inhérentes au problème en lui-même comme le nombre de classes, la complexité des formes, leur variabilité, leur ressemblance, etc. Les difficultés peuvent aussi venir de la représentation du problème, c'est-à-dire de la base de données : espaces de représentation mal choisis, bruits dans les formes et dans la base (présence de formes inconnues), quantité de données peu importante ou au contraire très importante, etc. Tous ces éléments influent de manière importante sur les processus d'apprentissage et de décision d'un classifieur, ce qui rend leur évaluation très délicate. Utiliser des bases issues de différents problèmes et de composition variée est donc un élément essentiel.

Dans un premier temps, nous nous sommes focalisé sur quatre types de problèmes de différentes natures. Nous avons utilisé deux benchmarks classiques : les *formes d'ondes de Breiman* ainsi que le problème des *images satellites*. Les deux autres problèmes sont relatifs à la reconnaissance en-ligne de caractères manuscrits. L'un porte sur les chiffres, l'autre sur les lettres minuscules. Nous présentons chacun d'eux dans les paragraphes suivants.

5.1.1 Les benchmarks classiques

On peut trouver à différents endroits un certain nombre de benchmarks permettant d'évaluer les classifieurs. Le site de l'UCI ML Repository[1] en regroupe plusieurs qui sont très fréquemment utilisés. Nous en avons sélectionné deux.

Les formes d'ondes est un problème artificiel composé de trois classes dont les représentants sont décrits par 21 caractéristiques à valeurs numériques. Il a été introduit par Breiman [30] dans le cadre de l'étude des arbres de décision et a été très largement utilisé par la suite. Il s'agit d'un problème complexe dont le taux de reconnaissance optimal (Bayésien) est de 86 %.

Le deuxième benchmark utilisé est le problème des images satellites du projet Statlog. Il y a 6 classes à identifier correspondant à différentes natures de sols. Les exemples sont caractérisés par 36 attributs numériques. Deux bases sont fournies : la base d'apprentissage contient 4435 individus et la base de test en contient 2000 autres.

5.1.2 Problèmes associés à la reconnaissance en-ligne de caractères manuscrits

Comme nous l'avons mentionné dans le chapitre 1, les difficultés associées à la reconnaissance de formes manuscrites sont particulièrement nombreuses et intéressantes

1. Il est accessible à l'adresse suivante : http://www.ics.uci.edu/~mlearn/MLRepository.html

à considérer. Le problème est en effet complexe en lui-même et la création de bases d'échantillons est elle aussi particulièrement délicate.

Cette autre série d'expérimentations vise à étudier le comportement du système à partir de données issues d'un cadre applicatif réel.

Dans ces expérimentations, nous avons utilisé deux bases de données contenant différents caractères manuscrits. Il s'agit d'une part de la base *IRONOFF* et d'autre part de la base *UNIPEN*.

5.1.2.1 La base IRONOFF

La base IRONOFF [168] a été conçue avec l'objectif de fournir simultanément la version en-ligne et hors-ligne de caractères et de mots manuscrits saisis par différents scripteurs. Les formes ont été acquises à partir d'une tablette électronique et d'un stylet électronique à encre au travers de formulaires papier. Ce processus a permis à la fois de capturer le signal en-ligne tout en conservant le signal hors-ligne (sur le formulaire) qui a été ensuite numérisé pour obtenir l'image correspondante. Dans ces expérimentations, nous avons travaillé uniquement sur la version en-ligne des chiffres manuscrits. Cette base contient environ 4000 chiffres saisis par 400 scripteurs différents. Comme on peut le voir sur la figure 5.1, la qualité des saisies est relativement correcte.

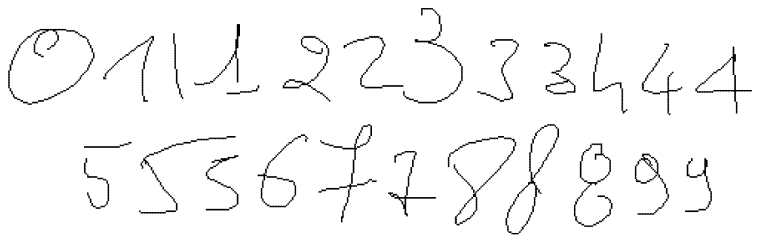

FIG. 5.1 – *Exemples de chiffres manuscrits de la base IRONOFF.*

5.1.2.2 La base UNIPEN

Lors de sa création (1992), le projet *UNIPEN*[2] [71] avait pour objectif de produire une base de grande taille pour le domaine de la reconnaissance d'écriture manuscrite, chose qui n'existait pas alors. Un grand nombre de caractères et de mots ont ainsi été recueillis, en provenance d'une quarantaine d'organismes différents. La conséquence directe de l'ampleur de cette base est qu'il existe une variabilité très importante des formes à reconnaître. Le nombre de scripteurs différents ayant participé au projet est en effet très important (plus de 2200). Une autre conséquence est la variabilité de la qualité de l'acquisition des tracés. Certaines formes ont été acquises sur des périphériques de mauvaise qualité ou mal réglés (résolution insuffisante) produisant des dégradations

2. http://unipen.nici.kun.nl/

importantes. De plus, certains tracés ont été mal segmentés et mal étiquetés, les rendant impossible à identifier.

Dans ces travaux, nous avons utilisé le corpus train r01-v07 de la base et plus particulièrement les sections 1*a* et 1*c* relatives aux chiffres arabes et aux minuscules de l'alphabet latin respectivement. Pour les chiffres, la base initiale contient 15953 exemples. Elle a été partiellement nettoyée en supprimant uniquement les tracés mal étiquetés ou mal segmentés, soit 146 individus. La figure 5.2 montre quelques exemples de cette base.

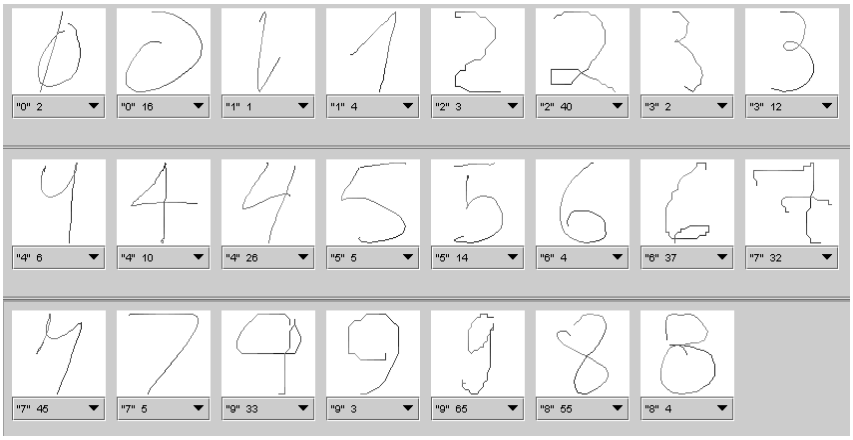

FIG. 5.2 – *Exemples de chiffres manuscrits de la base UNIPEN.*

Quant à la base de lettres, elle est constituée de 61351 exemples dont 265 ont été écartés. La figure 5.3 montre quelques exemples.

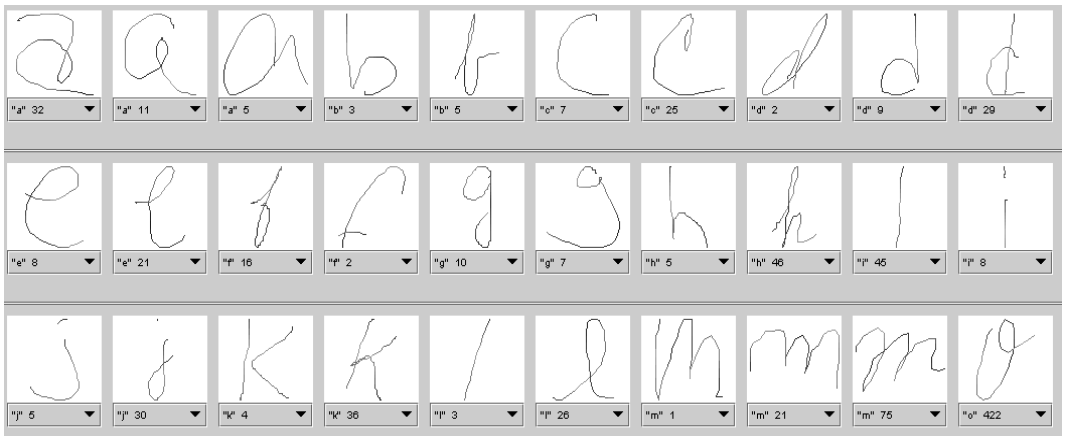

FIG. 5.3 – *Exemples de lettres manuscrites de la base UNIPEN.*

5.1.2.3 Type de caractéristiques utilisées pour décrire les tracés

Les tracés dynamiques issus des bases IRONOFF et UNIPEN ont été caractérisés par un ensemble d'attributs numériques. Ceux-ci ont été déterminés en s'appuyant sur l'expérience acquise grâce au système *ResifCar* et ils reposent donc sur les même principes [13]. Notre objectif n'est pas ici d'évaluer le système *Mélidis* en tant que reconnaisseur de formes manuscrites afin d'obtenir des performances brutes comparables à d'autres systèmes de ce genre. Les comparaisons avec d'autres classifieurs ont donc été effectuées dans les mêmes conditions et sur le même jeu de caractéristiques. Décrire précisément celles-ci n'a donc que peu d'intérêt ici et nous donnons simplement leur principe ci-dessous.

Avant l'extraction, les caractères sont simplement renormalisés pour avoir une taille standard. Le mécanisme d'extraction des caractéristiques s'appuie sur le module de segmentation des tracés de *ResifCar* qui permet de les découper en trois types de primitives : traits descendants, contextes morphologiques associés à ces traits et zones de liaisons (cf. paragraphe 1.4.1). Ce sont ces primitives qui sont ensuite caractérisées par leur position absolue, relative, leur courbure, etc.

Un premier jeu de 44 caractéristiques a ainsi été extrait pour les chiffres et il a été étendu à 73 pour les lettres.

5.2 Expérimentations préliminaires sur notre approche des arbres de décision flous

Nous reportons ici les résultats d'un ensemble d'expérimentations préléminaires qui ont été conduites afin de valider notre principe de construction et d'exploitation des arbres de décision flous comme nous les avons décrits dans les paragraphes 3.4.4, 4.2.2 et 4.3.3.

Les premières expérimentations ont été faites sur le benchmark des formes d'ondes de Breiman. Les résultats ont été obtenus en effectuant la moyenne de 5 validations croisées en 10 parties (cf. paragraphe 1.2.6.1) sur une base de 600 exemples.

L'arbre est configuré pour faire des partitionnements binaires avec les CMF dans des sous-espaces de dimension '*Dim.*'. Lorsque '*Dim.*'$= 1$, le choix de l'attribut se fait par une recherche exhaustive (la mesure d'entropie étoile est évaluée pour chaque attribut). Si '*Dim.*'≥ 2, la recherche s'effectue en utilisant l'algorithme génétique reposant sur la mesure d'entropie étoile. On indique aussi le critère d'arrêt (*Arrêt*) utilisé, qui correspond au nombre d'individus minimum nécessaire à un nœud donné pour continuer à partitionner. Le SIF déduit de l'arbre a été testé de deux façons. La première (*type 1*) correspond à l'utilisation des règles de la forme (4.6) avec une inférence type *max-produit* (équation (2.21)). La deuxième (*type 2*) correspond à la version optimisée du SIF avec les règles (4.8) dont les conclusions ont été déterminées par la méthode de

la pseudo-inverse sur la base d'apprentissage.[3]

Les résultats sont présentés dans le tableau 5.1. On y trouve le taux d'erreur (*Err.*), la taille de l'arbre exprimée en terme de nombre de règles/feuilles (*nb. règles*), profondeur maximale (*Prof. max.*) et profondeur moyenne (*Prof. moy.*), ainsi que l'écart-type du taux d'erreur (*Ec.*).

Dim.	Arrêt	SIF	Err.	nb. règles	Prof. max.	Prof. moy.	Ec.
1	4	*type 1*	23,9 %	72,8	9,1	6,8	5,2
1	10	*type 1*	22,8 %	31,7	6,9	5,3	4,1
1	4	*type 2*	21,9 %	74,3	9,4	6,9	4,7
1	10	*type 2*	22,2 %	32,0	7,0	5,3	4,5
2	4	*type 1*	20,6 %	99,8	10,0	7,5	3,9
2	10	*type 1*	20,7 %	37,1	7,2	5,8	3,7
2	4	*type 2*	19,8 %	98,1	9,7	7,5	4,8
2	10	*type 2*	18,6 %	37,0	7,3	5,8	4,0
3	10	*type 2*	18,4 %	49,1	8,6	6,5	4,4

TAB. 5.1 – *Comportement des arbres de décision flous utilisant les C-moyennes floues comme processus de partitionnement sur le problème des formes d'ondes.*

Ces résultats sont intéressants à plusieurs niveaux. On notera tout d'abord que comme annoncé, le choix du mode de représentation des règles ainsi que de l'inférence associée influe de manière assez nette sur les performances de l'arbre. Ainsi, les SIF optimisés de *type 2* ont des taux d'erreurs qui sont systématiquement inférieurs à ceux des SIF de type 1.

Un autre résultat intéressant concerne le choix de la taille du sous-espace utilisé pour faire les partitionnements. On constate en effet que les résultats sont nettement meilleurs en effectuant les partitionnement dans un sous-espace de dimension 2 plutôt qu'en se basant sur un seul attribut. Cependant, cet effet semble décroître avec l'augmentation de *Dim.* au-delà de deux, puisqu'avec trois attributs, le gain en performance est nettement moins important. De plus, on remarquera que le nombre de règles est aussi en augmentation (et le nombre de paramètres encore plus à cause de l'augmentation de la taille des sous-espaces).

On remarquera enfin l'importance du critère d'arrêt, à la fois sur la taille de l'arbre, mais aussi sur les performances. Pour faire un partitionnement qui soit représentatif, les *C-moyennes floues* nécessitent un nombre d'échantillons suffisamment important (et proportionnel à la taille de l'espace de représentation utilisé). C'est pourquoi les résultats sont meilleurs (en terme de taux d'erreur) lorsque l'on s'arrête plus tôt (moins de 10 individus). Comme expliqué dans la paragraphe 3.4.2.3, le critère d'arrêt était déjà un point important pour les arbres de décision classiques. Il l'est donc aussi dans le cas des arbres de décision flous, même si son impact est nettement moins important du fait de la modélisation granulaire par sous-ensembles flous. Ce phénomène est aussi

3. En théorie, il serait plus intéressant d'utiliser une base de validation mais le nombre d'échantillons pour l'apprentissage même deviendrait alors trop juste.

une illustration du problème du sur-apprentissage : lorsque l'arbre croît de façon trop importante, il fait un apprentissage par cœur et les performances en généralisation sont égales ou inférieures et moins stables (écart-type supérieur).

Nous terminons cette série d'expérimentations en reportant les résultats d'autres approches par arbres de décision sur le même problème. Le protocole de test est le même que celui utilisé dans [114] ainsi qu'au paragraphe 5.3.1 : une base de 600 individus est utilisée pour l'apprentissage des arbres qui sont testés sur une autre base indépendante de 3000 individus. Le tableau 5.2 résume les performances (taux d'erreurs) obtenues par : une approche par arbre de décision classique, le *C4.5* [142, 143] ; une approche d'arbre utilisant une combinaison linéaire des attributs pour effectuer les partitionnements, *Quest* [115] ; et notre approche floue basée sur les CMF dans sa configuration « standard » (partitionnements binaires en deux dimensions, critère d'arrêt fixé à dix individus et SIF de *type 2*).

Arbres de décision	Err.
C4.5	25,4 %
Quest	18,5 %
par les CMF	19,2 %

TAB. 5.2 – *Comparaison de différentes approches de reconnaissance par arbres de décision (classiques et flous) sur les formes d'ondes de Breiman.*

La comparaison entre *Quest* et *C4.5*, confirme que, comme pour nos arbres, se baser sur l'utilisation de plusieurs attributs pour effectuer les partitionnements permet une amélioration significative des performances. On remarquera aussi que l'approche floue (par les CMF) est sensiblement meilleure que l'approche classique du *C4.5*, ce qui justifie l'intérêt de l'introduction de la gestion des imprécisions. Ce point avait déjà été montré dans [121] que le lecteur pourra consulter pour avoir des résultats avec un autre arbre de décision flou sur la même base mais avec un protocole de test différent.

5.3 Évaluation des performances générales du système *Mélidis*

Dans un premier temps, nous donnons quelques résultats sur les performances générales du système afin de l'évaluer par rapport aux objectifs que nous nous sommes posés. Pour cela, nous présentons d'une part les taux de reconnaissance sur les problèmes présentés ci-dessus et, d'autre part, une évaluation de la compacité de la modélisation pour les problèmes liés au manuscrit. Les résultats présentés correspondent aux performances brutes du système : aucune optimisation n'a été faite après apprentissage et nous n'avons pas cherché à obtenir les meilleurs paramètres manuellement par de multiples expérimentations.

Très souvent, comparer objectivement une approche avec d'autres méthodes existantes est quelque chose de difficile. En effet, les conditions de test sont souvent diffé-

rentes (taille des bases utilisées en apprentissage et en test, espaces de représentation des formes différents, etc.) et pas toujours très explicites dans la littérature. Or ce sont des points qui influent nettement sur les propriétés d'un système. C'est pourquoi nous détaillons au maximum nos conditions d'expérimentations. Lorsque c'est possible, nous donnons des résultats comparatifs avec d'autres systèmes en précisant si les conditions étaient identiques (ou suffisamment proches) ou différentes. Pour les bases UNIPEN, nous renvoyons le lecteur à [146] pour un recensement des approches ayant été validées sur ces bases. À titre indicatif, les deux approches les plus performantes sur cette base obtiennent des résultats de $99,1$ % [139] et $98,9$ % [145] pour les chiffres, et de $97,8$ % [139] et $92,3$ % [145] pour les minuscules.

5.3.1 Benchmarks classiques

L'objectif de cette série d'expérimentations est de fournir une comparaison avec d'autres approches de classification citées dans la littérature. Pour que cette comparaison soit la plus informante, nous avons utilisé le même protocole expérimental que celui donné dans [114].

Pour les formes d'ondes, la base d'apprentissage utilisée contient 600 individus et les performances sont évaluées sur une base de test indépendante de 3000 individus.

La base des images satellites est quant à elle utilisée avec le découpage prédéterminé (cf. paragraphe 5.1.1).

Pour l'ensemble de ces tests, notre système a été configuré de manière standard, comme il a été présenté dans les chapitres 3 et 4. Les arbres de décision flous effectuent donc des partitionnements binaires dans des sous-espaces à deux dimensions choisis par l'algorithme génétique. De plus, une optimisation des conclusions des SIF a été faite en minimisant l'erreur de classification sur la base d'apprentissage[4] par la méthode de la pseudo-inverse.

Le tableau 5.3 présente les taux de reconnaissance moyens[5] obtenus par le système *Mélidis* sur les bases de test. Ces résultats sont comparés avec les performances obtenues par deux des meilleurs classifieurs sur ces deux problèmes, parmi l'ensemble des algorithmes recensés dans l'article de Lim et Loh [114] et qui ont été testés avec le même protocole expérimental. Il s'agit de réseaux de neurones de type RBFN [152], de l'approche LVQ (Learning Vector Quantization) [167].

Par rapport aux 33 techniques de classification recensées dans [114], le RBFN se classe en première position (en terme de taux de reconnaissance) sur les formes d'ondes et en deuxième position pour les images satellites. L'approche LVQ se classe quant à elle en première position sur les images satellites et en sixième position sur les formes d'ondes. Le système *Mélidis* s'intercale quant à lui en cinquième position sur les formes

4. En théorie, il faudrait utiliser une base de validation mais le nombre d'échantillons pour l'apprentissage même deviendrait trop juste.

5. Moyenne obtenue sur 5 phases apprentissage/test sur les bases décrites ci-dessus.

	Formes d'ondes	Images satellites
Mélidis	83,4 %	89,8 %
LVQ*	83,0 %	90,2 %
RBFN*	84,9 %	87,9 %

TAB. 5.3 – *Performances comparatives sur deux benchmarks classiques (* Ces taux sont tirés de [114]).*

d'ondes et en deuxième position sur les images satellites. Ses performances sont donc tout à fait intéressantes, ce qui apporte un premier élément de validation en faveur de l'objectif de généricité.

5.3.2 Reconnaissance en-ligne de chiffres manuscrits

Sur le problème de la reconnaissance de chiffres manuscrits, le système a été évalué selon deux modes : en omni-scripteurs sur la base *IRONOFF* et en multi-scripteurs sur la base *UNIPEN*.

5.3.2.1 Reconnaissance omni-scripteurs

Pour ces tests, la base *IRONOFF* a été divisée en deux sous-bases disjointes pour les phases d'apprentissage et de test. Ces deux sous-bases sont constituées chacune d'environ 200 échantillons par classe et les scripteurs sont différents dans les deux bases.

Pour l'apprentissage de notre système, le module de description globale s'appuie sur une présélection d'un jeu de 20 caractéristiques parmi les 44 disponibles en utilisant le principe vu dans le paragraphe 3.3.4.2. Le reste de la configuration est identique à celle présentée dans le paragraphe 5.3.1 (arbres de décision flous binaires effectuant des partitionnement dans des sous-espaces à deux dimensions choisis parmi l'espace de représentation complet ; optimisation des conclusions des SIF).

Le système *Mélidis* est comparé avec trois autres approches de reconnaissance dans les mêmes conditions expérimentales : un MLP (Multi-Layer Perceptron), un réseau de neurones de fonctions à base radiale (RBFN), et une machine à vecteurs de supports (SVM). Le MLP est composé d'une seule couche cachée pour laquelle différentes tailles ont été testées. Les poids ont été appris par rétro-propagation du gradient de l'erreur. Le RBFN [15] ne se base que sur le sous-espace de 20 caractéristiques présélectionnées pour le module de description intrinsèque de notre système, les performances étant meilleures qu'avec l'espace entier. Les poids de la couche de sortie ont été appris en utilisant la méthode de la pseudo-inverse. Enfin, pour la SVM, le logiciel SVMTorch[6] [41] a été utilisé en mode « multi-classes[7] » avec des fonctions de noyaux à base radiale. Les

6. SVMTorch est disponible sur le site : *ftp.idiap.ch/pub/learning/SVMTorch.gz.*

7. Un SVM est construit par classe pour faire la discrimination de cette classe contre toutes les autres.

paramètres de la SVM ont été affinés en effectuant des expérimentations successives.

En dehors de ces approches, nous reportons aussi à titre indicatif les performances de *ResifCar* [14] qui a été évalué dans des conditions similaires mais pas tout à fait identiques (exemples des bases d'apprentissage et de test légèrement différents, jeu de caractéristiques avec quelques variantes).

Le tableau 5.4 présente les résultats de ces différentes approches en précisant : une estimation du nombre de paramètres (*nb. param.*) nécessaire pour la modélisation [8] ; les taux de reconnaissance sur la base de test (*% reco.*).

	nb. param.	% reco.
MLP	5 400	92,7
MLP	8 100	93,5
MLP	10 800	93,3
RBFN	8 600	93,6
SVM	114 075 (137 565∗)	94,6 (95,5∗)
ResifCar	6 084	94,3
Mélidis	12 150 (12 416∗)	94,7 (95,8∗)

TAB. 5.4 – *Performances comparatives sur la reconnaissance en-ligne de chiffres en mode omni-scripteurs (∗ après optimisation du jeu de caractéristiques).*

En considérant dans un premier temps notre principal objectif (les performances), on constate que les approches neuronales (RBFN et MLP) obtiennent des taux de reconnaissance qui sont inférieurs aux autres approches, ce qui les rend moins intéressantes. Parmi les autres approches, le système *ResifCar* parvient à des taux de reconnaissance qui sont voisins de la SVM et de *Mélidis*, avec un nombre de paramètres très réduit. Cependant, il ne faut pas perdre de vue qu'il s'agit d'une approche dédiée, au contraire des deux autres. Finalement, la SVM et *Mélidis* obtiennent des performances comparables en terme de taux de reconnaissance et sont toutes les deux génériques. Cependant, le nombre de vecteurs supports de la SVM et donc le nombre de paramètres nécessaire à sa modélisation est largement supérieur à celui de *Mélidis* (environ 10 fois plus), ce qui peut devenir contraignant dans l'optique d'une intégration sur des machines avec des ressources limitées. Le système *Mélidis* répond donc ici parfaitement à la fois aux objectifs de performances, généricité et compacité (en terme de nombre de paramètres).

8. Il s'agit d'une estimation en fonction du nombre de sous-ensembles flous (neurones, fonctions à base radiale, vecteurs supports), et du nombre de paramètres nécessaires pour les décrire en fonction de la taille de l'espace utilisé pour chacun d'eux. Par exemple, un sous-ensemble flou en n dimensions décrit par son centre et sa matrice de covariance nécessite $n + n \times n$ paramètres.

5.3.2.2 Reconnaissance multi-scripteurs

Nous avons aussi testé notre système sur la base UNIPEN pour évaluer son comportement lorsque le nombre de scripteurs est très grand. La base a été divisée en deux bases disjointes $(2/3, 1/3)$ contenant respectivement 10556 échantillons pour l'apprentissage et 5251 pour le test, chaque scripteur se retrouvant dans les deux bases à la fois.

Pour ces expérimentations, notre système est configuré exactement de la même façon que dans les tests précédents et la comparaison, dans les même conditions, a été faite avec la SVM. Nous ne conservons ici que cette approche pour la comparaison car elle présente des performances en terme de taux de reconnaissance qui sont supérieures aux autres que nous avons testé. De plus, elle est reconnue comme étant une des méthodes de classification parmi les plus performantes à l'heure actuelle. Le tableau 5.5 donne les résultats de ces deux approches en terme de nombre de paramètres et de taux de reconnaissance.

	nb. param.	% reco.
SVM	308 025	97,8
Mélidis	14 600	96,3

TAB. 5.5 – *Performances comparatives avec une SVM sur la reconnaissance en-ligne de chiffres en mode multi-scripteurs.*

Sur cette base contenant beaucoup d'individus, on constate que notre approche obtient des taux de reconnaissance un peu moins bon que la SVM mais celle-ci voit son nombre de paramètres multipliés par presque trois alors que notre système conserve lui approximativement la même taille. Au total, le rapport en terme de compacité entre les deux approches est de plus de 20 en faveur de *Mélidis*.

5.3.3 Reconnaissance en-ligne de lettres manuscrites

Les dernières expérimentations ont porté sur la reconnaissance en-ligne des minuscules issues de la base UNIPEN. Ces tests ont été faits en mode multi-scripteurs en découpant la base en deux bases disjointes : la base d'apprentissage contient 40613 échantillons et la base de test 20473.

La configuration de notre système est toujours la même, sauf que le sous-espaces utilisé pour le module de description globale est de 21 caractéristiques au lieu de 20 pour les chiffres. Celles-ci ont à nouveau été présélectionnées par le même processus que celui décrit dans le paragraphe 3.3.4.2. La comparaison s'effectue là encore avec le classifieur SVM.

Les résultats sont reportés dans le tableau 5.6.

Pour ce problème, on constate à nouveau le même comportement des deux systèmes. Les taux de reconnaissance sont meilleurs avec la SVM mais pour cela, elle utilise environ 30 fois plus de paramètres. Pour compléter ces tests, et obtenir un lien plus étroit

	nb. param.	% reco.
SVM	2 438 784	92,2
Mélidis	82 506	89,7

TAB. 5.6 – *Performances comparatives avec une SVM sur la reconnaissance en-ligne de lettres minuscules en mode multi-scripteur.*

entre le nombre de paramètres et les taux de reconnaissance, d'autres expérimentations devraient être faites en utilisant moins d'échantillons en apprentissage.

5.4 Étude des différents modules du système

L'objectif des résultats reportés dans cette partie est de montrer l'intérêt et la validité de l'architecture du système ainsi que des choix qui ont été faits. Nous reportons donc les détails concernant le comportement des différents modules du système au cours des expérimentations précédentes.

5.4.1 Le système à la loupe

Dans ce paragraphe, nous reprenons les expérimentations qui ont été faites dans les paragraphes précédents en détaillant la participation de chacun des modules du système afin de mettre en évidence la complémentarité de la modélisation intrinsèque et de la modélisation discriminante.

Le tableau 5.7 reprend les résultats sur les deux benchmarks classiques.

	Formes d'ondes	Images satellites
Pré-classification	81,9 %	84,3 %
Classification principale	82,4 %	89,3 %
Classification finale	83,4 %	89,8 %

TAB. 5.7 – *Taux de reconnaissance des différents modules du système sur deux benchmarks classiques.*

Les résultats illustrent de façon assez nette la participation de chacun des modules du système. On constate en effet que chaque module supplémentaire permet d'améliorer les taux de reconnaissance. Ainsi, le module de classification principale permet de réduire les erreurs de pré-classification de $2,76$ % sur les formes d'ondes et de $31,8$ % sur les images satellites. De même, le module de classification finale réduit les erreurs de pré-classification de $8,3$ % sur les formes d'ondes et de 35 % sur les images satellites. C'est donc un premier élément de validation de l'architecture.

De plus, en observant de plus près les individus sur lesquels sont commises les erreurs, il est intéressant de remarquer que le nombre d'erreurs commises sur les mêmes

exemples par les modules de pré-classification et de classification principale est très inférieur au nombre d'erreurs commises indépendamment par chacun d'eux. Par exemple, sur les formes d'ondes, la pré-classification reposant sur les connaissances intrinsèques commet 543 erreurs en moyenne. La classification principale utilisant les connaissances discriminantes commet, elle, 528 erreurs et parmi celles-ci, seules 304 sont communes avec les 543. De la même manière, sur les images satellites, la pré-classification fait 314 erreurs, la classification principale 213 et seules 132 sont communes aux deux. Ces résultats laissent aussi imaginer que, comme annoncé, il doit être possible d'améliorer la méthode de fusion puisque les taux d'erreurs actuels sont supérieurs à ceux que l'on peut théoriquement atteindre en ne considérant que les erreurs communes.

Reprenons à présent les résultats du système sur la base IRONOFF (cf. tableau 5.8).

	nb. param.	% reco.
Pré-classification	8 500	93,8 (94,5∗)
Classification principale	3 650 (3 916∗)	94,0 (94,3∗)
Classification finale	12 150 (12 416∗)	94,7 (95,8∗)

TAB. 5.8 – *Performances des différents modules sur la reconnaissance en-ligne de chiffres en mode omni-scripteurs (base IRONOFF) (∗ après optimisation du jeu de caractéristiques).*

Sur ce problème, les deux premiers niveaux ont des performances en terme de taux de reconnaissance qui sont à peu près équivalentes. Mais là encore, la combinaison des deux par le module de fusion permet de réduire les erreurs de pré-classification de $23,6$ % (sur le jeu de caractéristiques optimisé (∗)).

On remarquera aussi, en ce qui concerne le nombre de paramètres, que c'est la modélisation intrinsèque des classes qui est la plus coûteuse, notamment à cause des matrices de covariance qui sont nécessaires pour définir les sous-ensembles flous. Une solution à envisager consisterait alors à faire une projection des sous-ensembles flous sur chacun des attributs et à évaluer la perte qu'elle engendre au niveau des performances.

Finalement, le tableau 5.9 présente les même types de résultats pour la reconnaissance de chiffres en mode multi-scripteurs sur la base UNIPEN.

	nb. param.	% reco.
Pré-classification	8 600	93,7
Classification discriminante	6 000	96,2
Classification finale	14 600	96,3

TAB. 5.9 – *Performances des différents modules sur la reconnaissance en-ligne de chiffres en mode multi-scripteurs (base UNIPEN).*

Là encore, chaque niveau permet d'améliorer les taux de reconnaissance. Cependant, cela se passe légèrement différemment. En effet, comme la base est plus complexe et le

nombre de scripteurs plus important, le niveau de modélisation intrinsèque semble avoir un peu plus de difficultés pour opérer une bonne caractérisation des classes.[9] Cependant, le niveau de modélisation discriminante permet de rattraper cette faiblesse grâce au mécanisme de décomposition guidée par les données qui permet aux arbres de se focaliser sur les formes difficiles à discriminer. La réduction des erreurs de pré-classification qu'il apporte est de $39,7$ % et l'ensemble réduit les erreurs de pré-classification de $41,3$ %.

Nous avons finalement reporté sur le graphique de la figure 5.4 la réduction relative des erreurs de chacun des niveaux par rapport aux erreurs de pré-classification comme énoncé précédemment.

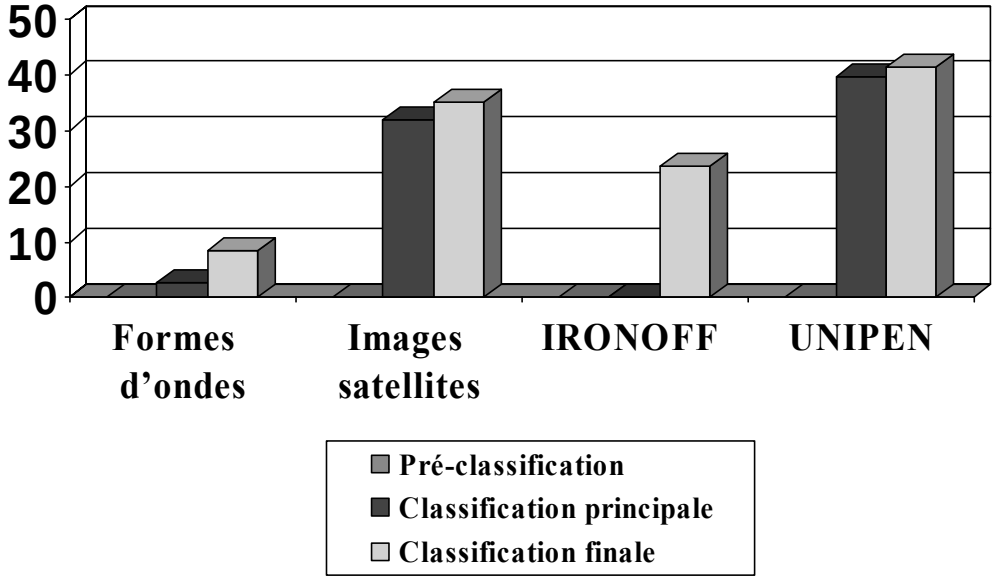

FIG. 5.4 – *Réduction relative des erreurs par rapport à la pré-classification (%).*

Celui-ci illustre parfaitement la complémentarité des différents niveaux de modélisation et l'intérêt de leur combinaison.

5.4.2 Apports de la décomposition guidée par les données

Les résultats précédents ont montré les apports combinés de chacun des modules de décision. On peut aussi essayer d'évaluer quel est l'apport de la décomposition guidée par les connaissances intrinsèques pour le module de discrimination focalisée. Autrement dit, on cherche à savoir si ce module fonctionne aussi bien ou moins bien selon qu'il est utilisé seul ou bien en bénéficiant des connaissances intrinsèques.

9. Cela devrait pouvoir être corrigé en « forçant » l'algorithme des CMP à repérer plus de sous-classes, puisque le critère $indice_{XB}$ tend à en favoriser un nombre limité.

Pour cela, nous avons construit et évalué une forêt[10] d'arbres de décision flous sur la base de chiffres IRONOFF, dans les mêmes conditions expérimentales que celles présentées précédemment. Cette forêt consiste à construire un arbre pour faire la discrimination de chaque classe vis-à-vis de l'ensemble des autres (*one against rest*). Ces arbres sont configurés exactement de la même manière que ceux utilisés par le module de discrimination focalisée et la prise de décision se fait de manière identique à la classification principale (la classe choisie est celle correspondant à l'arbre fournissant le score s^2_{i+} le plus élevé). La seule différence provient donc de la décomposition utilisée dans *Mélidis* et notamment du « filtrage » associé aux connaissances intrinsèques.

Dans cette situation, la forêt obtient un taux de reconnaissance sur la base de test de $92,4$ %, qui est assez nettement inférieur à celui de $94,0$ % obtenu par le module de classification principale de *Mélidis*. C'est donc un élément supplémentaire permettant de valider la structuration hiérarchique du système.

5.5 Implémentation

Le système *Mélidis* présenté dans ce mémoire a été implémenté en langage C++ de façon à offrir une bonne modularité et permettre l'ajout/la modification d'un certain nombre de fonctionnalités. Il ne s'agit donc encore que d'une version prototype destinée à l'expérimentation. La partie qui a été développée représente environ 14000 lignes de code commentées.

En ce qui concerne les vitesses d'exécution sur un Pentium III à 450MHz, le problème de l'apprentissage des chiffres met environ 2h sur la base IRONOFF et 8h sur la base UNIPEN. Les vitesses de reconnaissance sont de 100 chiffres à la seconde en moyenne. Pour les lettres de la base UNIPEN, l'apprentissage nécessite environ 12h sur un Pentium IV à 2,4GHz et la vitesse de reconnaissance est de 10 lettres à la secondes. Dans tous ces traitements, c'est le module de description intrinsèque par prototypes qui est le plus coûteux.

5.6 Bilan

La variété des problèmes traités au cours de ces expérimentations ont permis de valider l'essentiel des propriétés du système *Mélidis*.

Cette phase de tests a montré les bonnes propriétés de généricité et de compacité de la modélisation, notamment face à une approche de type SVM.

L'intérêt de l'architecture structurée autour de la combinaison de connaissances intrinsèques et discriminantes a aussi été validée puisque l'adjonction de chacun des niveaux de classification permet d'accroître les performances de manière sensible sur les problèmes difficiles.

Ces expérimentations ont aussi mis en évidence que c'était le niveau de modélisation intrinsèque qui était le plus coûteux en ressources. Il serait alors intéressant d'envisager le même type d'optimisations que celles qui ont été faites sur le système *ResifCar* et

10. Ce terme est repris de [121].

qui ont permis de diminuer sensiblement l'occupation mémoire et de diviser le temps de traitement moyen d'un caractère par un facteur de 20, le tout sans pertes de performances (d'autres types d'optimisations ont même permis d'accroître les performances). Suite à cela, il faudrait tester les possibilités d'intégration du classifieur sur des systèmes embarqués.

Conclusion et perspectives

La démarche qui a été présentée ici trouve son origine dans la constatation suivante : il existe toujours à l'heure actuelle des besoins importants en terme de classification et de reconnaissance. Cela est vrai d'une façon générale, mais aussi pour des domaines plus précis comme celui de la reconnaissance de formes manuscrites. En effet, bien qu'il existe des systèmes très performants, ceux-ci se trouvent très souvent soit confrontés à la complexité croissante des problèmes à traiter, soit inadaptés pour pouvoir répondre aux contraintes liées à leur intégration dans un cadre applicatif réel. Ce dernier point est particulièrement vrai si l'on considère la volonté de miniaturisation qui est omniprésente dans notre quotidien.

Nous sommes alors partis de la problématique riche et complexe de la reconnaissance de formes manuscrites pour la généraliser et définir un ensemble d'objectifs essentiels que devrait pouvoir satisfaire un système de reconnaissance de formes. Ces objectifs sont :

- la *généricité* afin de pouvoir traiter un grand nombre[11] de problèmes de natures et de complexités variables ;
- les *performances* afin de pouvoir utiliser le système dans des cadres applicatifs réels. Cet objectif inclut les notions de *robustesse*, de *fiabilité* et d'*adaptabilité* ;
- la *compacité* pour l'intégration dans des systèmes embarqués limités en ressources ;
- l'*interprétabilité* afin que le concepteur puisse détecter les problèmes, les corriger et optimiser le système en fonction des contraintes d'utilisation.

Ce dernier point devient à notre avis une nécessité parce qu'il n'est pas possible *a priori* de concevoir un reconnaisseur universel. Dès lors, des choix et des compromis doivent être faits et ils ne peuvent l'être efficacement que si le concepteur est capable de maîtriser et comprendre le système de reconnaissance et ses composants.

Dans ce contexte, nous avons adopté une vision centrée sur les connaissances du système, comment les définir, quels sont leurs rôles et leurs propriétés, que ce soit par rapport au problème (représenté par les données) ou bien par rapport aux objectifs que le concepteur s'est fixé. Nous avons alors dégagé deux natures de connaissances élémentaires qui sont fondamentalement différentes, complémentaires et pourtant rarement exploitées ensemble. Il s'agit d'une part des *connaissances intrinsèques aux classes* et d'autre part des *connaissances discriminantes entre classes*. Les premières permettent

11. Pouvoir les traiter tous avec une même approche reste un problème non résolu !

de décrire les classes indépendamment les unes des autres, de façon robuste et stable, en mettant en avant leur caractère multimodal. Elles peuvent donc être utilisées pour obtenir une première description du problème mais aussi pour le décomposer en sous-problèmes d'une façon « naturelle » et robuste. Elles sont à notre avis nécessaires pour déterminer un ensemble de « contextes » précis dans lesquels les connaissances discriminantes peuvent opérer de façon plus efficace. Ces dernières possèdent en effet une propriété forte de contextualité, ce qui les rend difficiles à extraire et particulièrement sensibles. L'utilisation d'un mécanisme de focalisation et leur organisation de façon hiérarchique sont donc particulièrement adaptés. Nous avons alors associé ces deux types de connaissances en les intégrant dans une architecture fortement structurée de façon à bénéficier au mieux de leurs propriétés et de leurs complémentarités.

À la différence d'autres approches reposant sur un couplage similaire entre une description des classes par modèles (ou encore patrons, ou prototypes) et leur discrimination, nous avons introduit ici un lien explicite entre les connaissances elles-mêmes ainsi qu'entre celles-ci et le problème (au travers des données). L'idée consiste à s'appuyer sur les regroupements naturels des données dans leur espace de représentation de façon à établir des liens robustes entre les connaissances et afin d'obtenir une interprétabilité suffisante pour faciliter le travail du concepteur. Ce point est pour nous essentiel puisqu'il n'existe à l'heure actuelle que bien peu d'informations permettant de choisir, pour un problème donné, le mécanisme de reconnaissance le plus adapté et de le configurer facilement en fonction de son contexte d'utilisation.

Nous avons alors choisi d'utiliser le formalisme des sous-ensembles flous qui permet de décrire les données d'une façon à la fois robuste, synthétique et compréhensible. Ceux-ci sont extraits automatiquement par des algorithmes de classification floue non supervisée qui s'appuient sur la structure naturelle des données. Ils ont été choisis afin de représenter exactement les deux natures de connaissances qui nous intéressent. Ainsi, les connaissances intrinsèques aux classes sont extraites par un algorithme de type *C-moyennes possibilites*. Quant aux connaissances discriminantes, elles sont obtenues par les *C-moyennes floues* qui sont intégrées dans une structure d'arbre de décision flou spécifiquement adapté pour la discrimination.

Pour la reconnaissance, la fonction de décision a été séparée de la modélisation de façon à pouvoir l'optimiser sans remettre en cause toute l'architecture. Elle se traduit par une combinaison de systèmes d'inférence floue qui permettent d'avoir une représentation homogène, plus facilement manipulable et interprétable et qui permettent l'utilisation de différents mécanismes d'inférence en fonction des usages. Cette combinaison exploite la complémentarité des deux natures de connaissances, à la fois du point de vue de la structuration, mais aussi du point de vue des informations différentes qu'elles apportent pour la reconnaissance.

Le système *Mélidis* a été testé sur différents types de problèmes, de natures et de complexités variables, de façon à pouvoir valider le côté *générique*. Les expérimentations ont porté d'une part sur des benchmarks classiques afin d'apporter des éléments de comparaison par rapport à d'autres approches de classification et d'autres part sur des problèmes plus spécifiques et plus riches issus de la reconnaissance en-ligne de caractères

manuscrits.

Les performances brutes obtenues (sans optimisation des paramètres ni de la structure) sont très bonnes sur les benchmarks classiques. Sur les problèmes liés à l'écriture manuscrite, les performances sont similaires ou légèrement inférieures à celles d'une machine à vecteur support. Cependant, celle-ci a nécessité une mise au point délicate pour parvenir à la meilleure configuration (en terme de taux de reconnaissance). De plus, elle utilise un très grand nombre de paramètres. Celui-ci peut être jusqu'à 30 fois supérieur à celui utilisé par notre approche, ce qui la rend difficilement intégrable dans des systèmes embarqués aux ressources limitées.

Ces tests ont aussi permis de mettre en évidence l'intérêt de la structuration du système ainsi que de la complémentarité des connaissances intrinsèques et discriminantes. Chacune d'elle, ainsi que leur combinaison, permet en effet d'améliorer les performances de l'ensemble de façon notable.

Perspectives

Le système tel qu'il a été décrit dans ce manuscrit constitue les fondements d'une approche qui peut être améliorée à plusieurs niveaux.

En ce qui concerne le niveau de modélisation intrinsèque et aux vues des résultats, il apparaît que la description des sous-classes par des prototypes définis par leur centre et leur matrice de covariance est la partie du système qui nécessite le plus de paramètres. Pour accroître encore la compacité de *Mélidis*, différentes possibilités doivent être étudiées pour essayer de réduire la taille de ces modèles sans que les performances n'en soient trop affectées.

Une première solution dans ce sens, que nous avons commencé à explorer, consiste à sélectionner un sous-espace d'attributs dans lequel faire la modélisation intrinsèque. Cette sélection se fait actuellement par algorithme génétique. La fonction de fitness utilisée repose sur l'évaluation des performances de pré-classification du système. On peut aussi envisager de s'appuyer davantage sur une évaluation de la robustesse des modèles intrinsèques en utilisant par exemple des critères de compacité et de séparabilité des prototypes. Le couplage entre ces critères de performances et de robustesse est aussi une piste de recherche intéressante.

Une deuxième solution, qui a été utilisée dans *ResifCar* et qui est très avantageuse, consiste à projeter les sous-ensembles flous correspondant aux sous-classes sur chacun des attributs de l'espace. Il faudrait alors évaluer la perte d'information résultant de cette projection afin que le concepteur puisse faire les choix qui lui conviennent le plus.

Toujours sur ce niveau intrinsèque, la mesure actuelle utilisée pour trouver le bon nombre de sous-classes, semble limiter leur nombre par rapport à ce quoi on s'attendrait (sur la base UNIPEN notamment). Il faut donc déterminer si ce problème vient des caractéristiques définies ou de la mesure en elle-même et comment détecter ces cas et y remédier. Dans ce cadre, de nombreuses études ont été faites sur les mesures d'évaluation des partitionnements [185, 174, 123] et il conviendrait de les étudier plus en détails.

Si l'on considère à présent la méthode de décomposition du problème, on s'aperçoit que lorsque les problèmes deviennent très complexes (reconnaissance des lettres minuscules par exemple), constituer un sous-problème par classe (à partir de la description en sous-classes) limite les performances. Une solution, que nous avons mentionné, consiste à décomposer le problème directement en sous-classes lorsque ceux-ci deviennent trop complexes. Cette modification n'impose pas de remise en cause de l'architecture du système puisque seule une adaptation du mode de combinaison des experts du module de classification principale est nécessaire.

Pour les arbres de décision flous, un certain nombre d'extensions sont envisageables. Celle qui nous parait la plus intéressante dans un premier temps consisterait à remplacer l'algorithme des *C-moyennes floues* par une version un peu plus robuste. Par exemple, les *C-moyennes floues possibilistes* [132], qui combinent le mode de partitionnement relatif des *C-moyennes floues* avec la robustesse face au bruit des *C-moyennes possibilistes*, est à étudier en détails. L'extension vers une supervision partielle ou totale de l'algorithme est aussi un point clé à approfondir afin de pouvoir ajuster le positionnement des frontières discriminantes, notamment lorsque la quantité d'échantillons est très différente entre les deux classes à discriminer. Enfin, l'initialisation de l'algorithme est aussi un point qu'il peut être intéressant d'améliorer afin d'augmenter la robustesse et la vitesse de convergence des algorithmes.

Un deuxième point qui nous semble important concerne la taille des arbres utilisés. Il s'agit en effet d'un élément crucial dans les arbres de décision classique et il conditionne directement leurs performances. Nos expérimentations semblent aussi aller dans ce sens pour les arbres de décision flous même si ce phénomène est implicitement atténué grâce à leur capacité de synthèse. On peut donc envisager d'utiliser des critères d'arrêt plus performants ou bien des algorithmes d'élagages adaptés aux arbres flous (ou aux systèmes d'inférence flous qui en sont déduits).

En ce qui concerne le processus de décision, il semble intéressant d'approfondir les différents mécanismes d'inférence pour évaluer plus précisément leurs apports/manques en terme d'interprétabilité, de performances et par rapport aux connaissances qui sont exploitées. L'introduction de la gestion des incertitudes est un point à étudier plus particulièrement, notamment pour les mécanismes de combinaison. Ceux qui sont actuellement utilisés restent relativement simples et ne permettent pas d'exploiter entièrement la complémentarité des deux niveaux de modélisation.

Un autre point qu'il reste à travailler à propos du processus de décision concerne l'introduction des mécanismes de rejet de distance et de refus de classement afin d'améliorer la fiabilité des réponses. Ces deux notions sont particulièrement importantes, notamment lorsque l'on travaille sur des bases comme UNIPEN. Nous poursuivons actuellement ces travaux.

Enfin, pour revenir sur le contexte applicatif, nos prochaines études porteront sur les possibilités d'intégration du classifieur *Mélidis* sur des systèmes nomades. Cela va nous permettre de valider plus précisément ses propriétés et notamment d'exploiter l'interpré-

tabilité et la modularité du système et de comparer ainsi les possibilités d'optimisation à celles du classifieur dédié *ResifCar*.

Annexes

Annexe A

Principales distances

Soit x et y deux vecteurs définis dans un espace de représentation numérique $\mathcal{R}^n$: $x = [x^1, \ldots, x^k, \ldots, x^n]^T$ et $y = [y^1, \ldots, y^k, \ldots, y^n]^T$. Deux grandes familles de distances sont couramment employées en reconnaissance de formes et en classification : les *distances de Minkowski* et les *distances de Mahalanobis*.

Les distances de Minkowski sont définies de la manière suivante :

$$d(x,y) = \sqrt[p]{\sum_{k=1}^{n} |x_k - y_k|^p}$$

Cette famille permet par le choix de p de retrouver les distances usuelles telles que :

– la distance de Hamming (ou encore *city bloc*) avec $p = 1$:

$$d(x,y) = \sum_{k=1}^{n} |x_k - y_k|$$

– la distance Euclidienne si $p = 2$:

$$d(x,y) = \sqrt{\sum_{i=k}^{n} |x_k - y_k|^2}$$

– la distance de Tschebyshev si $p = \infty$:

$$d(x,y) = \max_{k=1,\ldots,n} |x_k - y_k|$$

La figure A.1 illustre le comportement de ces trois distances dans un espace (a_1, a_2) de $\mathcal{R}^2$ en indiquant l'ensemble des points x à égale distance du point $y = (0,0)$.

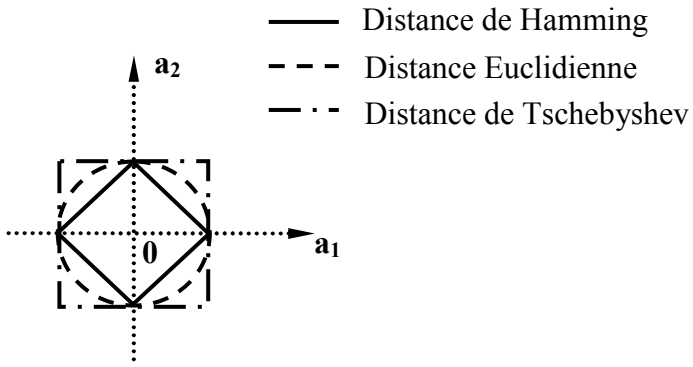

FIG. A.1 – *Illustration des principales distances de Minkowski.*

Les distances de Mahalanobis sont quant à elles définies de manière générale par :

$$d(x,y) = (x-y)^t M^{-1}(x-y) \tag{A.1}$$

où M est une matrice définie positive. On notera, deux cas particuliers des distances de Mahalanobis. Si la matrice M est la matrice identité, on retrouve la distance Euclidienne. Si la matrice M est diagonale (les attributs de l'espace de représentation sont statistiquement indépendants les uns par rapport aux autres), la distance devient équivalente à la distance de Bhattaharaya : $d(x,y) = \sum_{k=1}^{n} \frac{(x_k-y_k)^2}{\sigma_k^2}$ où les σ_k^2 sont les éléments diagonaux de M.

Annexe B

Classification non supervisée

La classification non supervisée a pour objectif de déterminer une structuration des données manipulées. Pour cela, des regroupements (ou classes) sont recherchés à partir d'une base d'exemples non étiquetés. Chaque regroupement est caractérisé par un sous-ensemble d'exemples ayant des propriétés communes et présentant donc une certaine similarité entre eux.

On distingue essentiellement deux grands types d'approches pour la classification non supervisée : les *approches hiérarchiques* et les *approches adaptatives*. Les premières utilisent les données initiales soit pour les regrouper au fur et à mesure (*approches agglomératives*), soit au contraire pour effectuer des divisions successives (*approches divisives*) (cf. paragraphe B.1). Les secondes visent à optimiser une partition initiale de manière itérative en minimisant une fonction objectif (cf. paragraphe B.2).

Quelle que soit l'approche envisagée, toutes reposent sur un même fondement : l'utilisation d'une *mesure de similarité* (ou de dissimilarité) qui permet de déterminer si deux individus de la base et par extension si deux sous-ensembles d'individus (deux regroupements) se ressemblent.

Plusieurs études ont été faites notamment pour pouvoir comparer différents types d'individus. Il existe ainsi des mesures similarité/dissimilarité entre des chaînes de caractères [112, 170], entre des sous-ensembles flous ou des prototypes flous [121, 147], entre des mots (au sens acoustique du terme), des images [140, 3], etc.

Dans les espaces de représentation numériques, les mesures de dissimilarité les plus courantes reposent sur l'utilisation d'une distance (cf. annexe A). Le choix de celle-ci est un élément très important pour le bon fonctionnement des algorithmes de classification non supervisée. Il influe très fortement sur la forme des regroupements trouvés ainsi que sur les propriétés de classification qui en découlent. Ainsi, si la distance Euclidienne fonctionne bien dans des espaces où les données sont réparties de façon homogène dans toutes les dimensions, elle donnera des résultats moins satisfaisants si les données sont étalées selon une direction privilégiée dans l'espace.

B.1 Les algorithmes de classification non supervisée hiérarchiques

Comme pour tout principe de classification non supervisée, la classification hiérarchique [86] détermine une structuration des données en regroupant celles qui possèdent des propriétés similaires. Cependant, elle ne s'arrête pas à cette structuration « horizontale » en classes. Elle cherche aussi à établir un lien hiérarchique entre les regroupements. Ce mécanisme de classification arborescent est très souvent utilisé en sciences naturelles par exemple. Il existe essentiellement deux types d'algorithmes : les algorithmes procédant *par agglomération* et ceux fonctionnant *par division* [45].

Les algorithmes agglomératifs partent de la base d'exemples initiale et considèrent chaque individu comme étant différent des autres. Chaque exemple constitue donc un regroupement à lui seul. Ensuite, à chaque étape, les deux regroupements les plus semblables sont recherchés et sont fusionnés pour former un nouveau regroupement qui remplace les deux précédents. Le processus est réitéré jusqu'à ce que le nombre de regroupements souhaité soit obtenu ou qu'il n'y en ait plus qu'un seul.

Il existe plusieurs méthodes permettant de déterminer quels regroupements il faut fusionner. Beaucoup sont fondées sur l'utilisation d'une distance d. Ainsi, si $\{C_1, \ldots, C_C\}$ représente l'ensemble des regroupements qui ont déjà été établis à l'itération courante et que x et y sont les exemples de la base, on peut utiliser les critères suivants pour fusionner deux regroupements i et j :

- la distance minimum entre les regroupements :

$$d_{min}(C_i, C_j) = \min_{x \in C_i, y \in C_j} d(x, y) \ ;$$

- la distance maximum entre les regroupements :

$$d_{max}(C_i, C_j) = \max_{x \in C_i, y \in C_j} d(x, y) \ ;$$

- la distance moyenne entre les regroupements :

$$d_{moy}(C_i, C_j) = \frac{1}{|C_i| \cdot |C_j|} \sum_{x \in C_i} \sum_{y \in C_j} d(x, y) \ ,$$

  où $|C_i|$ (respectivement $|C_j|$) représente le nombre d'échantillons appartenant à C_i (respectivement C_j) ;
- la distance entre les centroïdes des regroupements :

$$d_{centre}(C_i, C_j) = d(m_{C_i}, m_{C_j}) \ ,$$

  où m_{C_i} et m_{C_j} sont les centroïdes des regroupements C_i et C_j.

Cette dernière méthode évite le calcul de toutes les distances entre les couples (x, y) de données. Cependant, elle n'est applicable que dans le cas où un centre peut être défini.

Cela ne pose pas de problèmes dans un espace de représentation numérique (on peut utiliser par exemple le vecteur moyen) mais devient plus complexe si on travail sur des chaînes de caractères par exemple.

Le résultat de la classification hiérarchique peut être visualisé sous la forme de dendogramme. Cette représentation graphique indique non seulement les regroupements successifs qui ont eut lieu mais aussi la distance entre les groupes fusionnés. Elle permet donc de détecter facilement quand une agglomération semble naturelle (distance peu importante) ou au contraire lorsqu'elle semble avoir été forcée (cf. figure B.1).

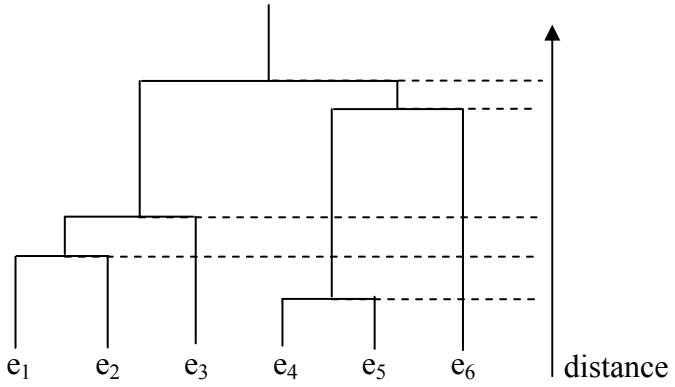

FIG. B.1 – *Exemple de dendogramme résultant d'une classification non supervisée hiérarchique.*

Les algorithmes divisifs fonctionnent à l'inverse des algorithmes précédents : le point de départ est un regroupement unique constitué de l'ensemble des données de la base. Des divisions sont ensuite faites à chaque pas de l'itération. Ces divisions peuvent par exemple s'opérer sur les regroupements de plus grande taille ce qui favorise les partitions équilibrées mais ne reflète pas nécessairement la réalité. Utiliser la distance intra-classe au sein de chaque regroupement comme critère de division donne alors de meilleurs résultats lorsque les regroupements naturels sont de tailles variées [45].

B.2 Les algorithmes basés sur une fonction objectif

Ces algorithmes fonctionnent d'une façon très différente des algorithmes hiérarchiques. Ils cherchent à obtenir la partition qui satisfait au mieux un critère donné : la fonction objectif. Celle-ci repose généralement sur la minimisation de la distance intra-classe et sur la maximisation de la distance inter-classe. Comme une recherche exhaustive de toutes les partitions possibles n'est pas réalisable (pour des raisons combinatoires évidentes), l'idée consiste à optimiser une partition initiale de façon itérative.

Ces algorithmes nécessitent très souvent l'introduction de connaissances *a priori* telles que : la fonction objectif à optimiser [50], la forme des regroupements recherchés,

leur nombre, ainsi que la partition de départ. La plupart fonctionnent sur des espaces de représentation numériques, facilitant la définition de la fonction objectif. Les regroupements sont alors caractérisés par des prototypes correspondant le plus souvent à des centroïdes.

Les différentes étapes de l'algorithme sont les suivantes :
- *initialisation :* déterminer une partition initiale ;[1]
- *calcul des centres :* déduire les prototypes à partir du partitionnement courant ;
- *partitionnement :* effectuer un nouveau partitionnement des données en fonction des prototypes et de la mesure de similarité/dissimilarité ;
- *arrêt :* arrêt si l'algorithme a convergé, sinon reprendre à l'étape de *calcul des centres*.

Pour illustrer de manière plus explicite ce fonctionnement, nous présentons ici l'algorithme général des *C-moyennes* qui se décline en différentes versions suivant les modes d'initialisation, la méthode de calcul des prototypes et la mesure de similarité utilisée [116, 11, 48].

Soit $\mathcal{E} = \{e_j | j = 1, \ldots, N\}$ l'ensemble des données de $\mathcal{R}^n$ utilisées pour établir le partitionnement, C le nombre de regroupements cherchés, $\mathcal{P} = \{P_1, \ldots, P_C\}$ l'ensemble des prototypes correspondants, $d(e_j, P_c)$ la distance Euclidienne utilisée pour mesurer la dissemblance entre une donnée e_j et un prototype P_c et $\{\varphi_{P_c} | c = 1, \ldots, C\}$ les fonctions caractéristiques indiquant l'appartenance (ou non appartenance) d'un élément à chacun des C regroupements. Les fonctions caractéristiques φ_{P_c} sont des fonctions $\mathcal{E} \rightarrow \{0, 1\}$ définies par :

$$\varphi_{P_c}(e_j) = \begin{cases} 1 & si\ c = argmin_{c'=1,\ldots,C} \quad d(e_j, P_{c'}) \\ 0 & sinon \end{cases} \tag{B.1}$$

Le partitionnement à l'itération t est représenté par la matrice $C \times N\ U = [\varphi_{P_c}(e_j)]$. Ce partitionnement est *net* puisqu'une donnée ne peut appartenir qu'à un et un seul regroupement. Leurs intersections deux à deux sont par conséquent vides.

Le critère à optimiser est basé sur le principe des moindres carrés :

$$J_{P,U,\mathcal{E}} = \sum_{c=1}^{C} \sum_{j=1}^{N} \varphi_{P_c}(e_j) d^2(e_j, P_c) \tag{B.2}$$

Il mesure la distance moyenne intra-groupe que l'algorithme doit minimiser.

1. L'initialisation peut aussi être faite en déterminant des prototypes initiaux. Dans ce cas, les étapes suivantes d'adaptation et de partitionnement sont inversées.

Les différentes étapes de l'algorithme sont :

- *initialisation :* initialiser la matrice U ;
- *calcul de centres :* les prototypes, correspondant aux centres des regroupements, sont évalués de la manière suivante :

$$\forall c \in \{1, \ldots, C\}, \quad P_c = \frac{\sum_{j=1}^{N} \varphi_{P_c}(e_j) e_j}{\sum_{j=1}^{N} \varphi_{P_c}(e_j)} \tag{B.3}$$

- *partitionnement :* la matrice U est sauvegardée dans U_{svgd} puis mise à jour en calculant $\varphi_{P_c}(e_j)$ $\forall j \in \{1, \ldots, N\}, \forall c \in \{1, \ldots, C\}$ (cf. équation B.1) ;
- *arrêt :* le partitionnement est réitéré tant que $\|U - U_{svgd}\| \geq \epsilon$.

La complexité de cet algorithme, dont la convergence a été prouvée, est inférieure à celle des algorithmes de classification hiérarchique standards quand N est grand [39]. Il possède cependant plusieurs inconvénients. Ainsi, même si la convergence est assurée, celle-ci ne garantie pas d'atteindre un minimum global. L'initialisation est donc particulièrement importante puisqu'elle peut conduire à des partitionnements différents. De plus, il est souvent difficile de déterminer *a priori* le nombre de regroupements C optimal.

Annexe C

Algorithmes génétiques

Les algorithmes génétiques [67] s'inspirent des principes évolutionnistes pour résoudre des problèmes. Un individu, correspondant à une solution possible au problème, est codé par une chaîne dont les éléments sont appelés gènes. Ces derniers représentent les « paramètres » variables de la solution. Une population de tels individus est créée et elle évolue par des mécanismes de mutations, croisements et reproductions. L'objectif de cette évolution est de permettre l'apparition du meilleur individu pour un critère donné, c'est-à-dire la meilleure solution du problème.

Nous décrivons ci-dessous les principes généraux de ces algorithmes et nous illustrons rapidement les principes de codage, croisement et mutation pour le problème de la sélection de caractéristiques. Pour plus de détails, le lecteur peut se référer à [67, 159].

C.1 Principes généraux des algorithmes génétiques

Pour pouvoir mettre en œuvre un algorithme génétique, il faut commencer par définir un codage du problème sous la forme d'une chaîne de gènes[1] ou chromosome. Il faut aussi définir la fonction de *fitness* qui permet d'évaluer la pertinence d'un individu par rapport au problème. Il peut s'agir d'une simple fonction mathématique ou d'un processus complexe qui fait correspondre à toute chaîne de gènes un score.

Une fois ces éléments déterminés, l'algorithme génétique se déroule de la manière suivante :

- *Initialisation :* Initialiser la population et calculer le fitness de chacun des individus
- *Faire jusqu'à FIN :* Créer une nouvelle génération :
  - Sélectionner les parents dans la population ;
  - Faire des croisements et des mutations ;
  - Calculer le fitness des nouveaux individus ;
  - Sélectionner les survivants.

1. Dans le cadre des méthodes évolutionnistes, ces gènes peuvent prendre un ensemble de valeurs différentes. Cependant, dans le cas des algorithmes génétiques, ils sont normalement binaires.

Dans cet algorithme, l'initialisation s'effectue généralement en créant artificiellement des individus en leur affectant aléatoirement des valeurs pour chacun des gènes.

La sélection des parents qui seront utilisés pour le renouvellement de la population et notamment pour la création de nouveaux individus s'effectue à partir du score de fitness des individus : plus un individu possède un score important (est proche de la meilleure solution) plus il sera favorisé pour engendré un ou plusieurs nouveaux enfants.

Une fois les parents sélectionnés, ceux-ci sont croisés entre eux de façon à obtenir de nouveaux individus. Un croisement s'effectue à partir de la chaîne de gènes des deux parents et en sélectionnant un ou plusieurs points de croisement dans celle-ci. Ces points de croisement déterminent des sous-séquences de gènes qui sont combinées entre parents pour créer de nouveaux individus. Ces derniers peuvent ensuite subir des mutations au niveau d'un ou plusieurs gènes en altérant leurs valeurs.

Les nouveaux individus sont évalués à leur tour en calculant leur fitness et ceux qui possèdent les scores les plus importants (parents compris en général) ont le plus de chance de survivre pour constituer la nouvelle génération.

Le processus est réitéré jusqu'à ce qu'un critère d'arrêt soit rencontré (*FIN*). Il peut s'agir d'un nombre de génération maximal fixé, d'un critère de convergence reposant sur l'évolution du fitness d'une génération à l'autre, etc.

C.2 Exemple pour la sélection de caractéristiques

Les algorithmes génétiques sont fréquemment utilisés lorsque l'on souhaite éviter les problèmes liés à l'explosion combinatoire. En reconnaissance de formes, ils constituent donc un outil particulièrement utile pour paramétrer un système (réseaux de neurones, systèmes d'inférence floue, etc.) ou encore pour faire de la sélection de caractéristiques. Nous décrivons ci-dessous un exemple simple appliqué à ce dernier cas pour illustrer les principes de codage des individus ainsi que les opérations de croisement et de mutation.

Soit B_{app} une base d'apprentissage et B_{val} une base de validation, toutes les deux constituées d'exemples décrits par n attributs. On recherche un sous-espace de n' attributs ($n' \leq n$) permettant d'optimiser les performances d'un classifieur sur la base de validation. On utilise pour cela un algorithme génétique qui fonctionne de la manière suivante.

Un individu, solution particulière de notre problème, est décrit par n gènes qui correspondent à chaque attribut i, $i = 1, \ldots, n$ de l'espace de représentation. Ces gènes sont binaires (booléens) et indiquent si l'attribut i fait partie (marqué V comme vrai) ou non (marqué F comme faux) du sous-espace solution. La figure C.1 donne un exemple de deux individus ainsi codés lorsque $n = 6$. Le premier est la solution initiale (tous les attributs sont sélectionnés) et le deuxième est une solution possible n'utilisant que 3 attributs.

La fonction de fitness utilisée pour évaluer un individu *Ind* consiste à faire l'appren-

tissage du classifieur à partir de B_{app} et à évaluer ses performances sur B_{val}, le tout en se restreignant à l'utilisation du sous-espace d'attributs décrit par les gènes de Ind.

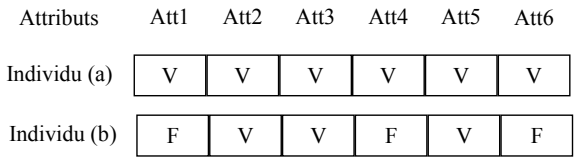

FIG. C.1 – *Exemple de codage des individus dans un algorithme génétique utilisé pour rechercher un sous-espace d'attributs : un individu est représenté par 6 gènes à valeurs booléennes correspondant aux 6 attributs de l'espace complet. L'individu (a) représente l'espace complet, et l'individu (b) représente un sous-espace de 3 attributs.*

À partir de ce codage, l'opération de croisement est illustrée sur la figure C.2. Deux fils sont créés à partir d'un individu (a) et d'un individu (b) en utilisant un seul point de croisement situé entre les gènes correspondants à l'attribut Att2 et à l'attribut Att3.

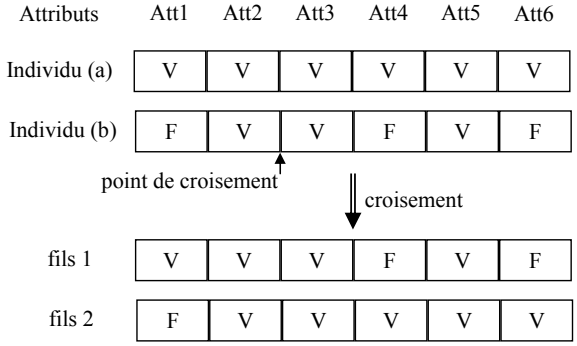

FIG. C.2 – *Exemple de la création de deux fils à partir de deux individus (a) et (b) en utilisant un croisement à un point.*

De même, la figure C.3 illustre la mutation du fils 1 de la figure précédente au niveau du gène correspondant à l'attribut Att4.

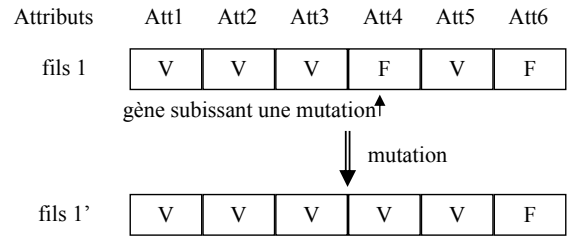

FIG. C.3 – *Exemple de mutation d'un individu sur le gène correspondant à l'attribut Att4.*

Publications personnelles

Articles de revue :

Nicolas Ragot et Éric Anquetil. - Système de classification hybride interprétable par construction automatique de systèmes d'inférence floue. - *Technique et science informatiques*, Hermès, volume 22, numéro 7, pp. 853-878, 2003.

Articles de conférences internationales avec comité de lecture :

Nicolas Ragot et Éric Anquetil. - A Generic Hybrid Classifier Based on Hierarchical Fuzzy Modeling: Experiments on On-Line Handwritten Character Recognition. - In *IEEE International Conference on Document Analysis and Recognition (ICDAR'03)*, volume 2, pp. 963-967, 2003.

Nicolas Ragot et Éric Anquetil. - A New Hybrid Learning Method for Fuzzy Decision Trees. - In *Proceedings of the 10th IEEE International Conference on Fuzzy Systems*, volume 3, pp. 1380-1383, 2001.

Articles de conférences internationales francophones avec comité de lecture :

Nicolas Ragot et Éric Anquetil. - Combinaison hiérarchique de systèmes d'inférence floue : application à la reconnaissance en-ligne de chiffres manuscrits. - In *Actes du congrès CIFED'02, Conférence Fédérative sur l'Écrit et le Document*, pp. 305-314, 2002.

Nicolas Ragot et Éric Anquetil. - Modélisation Automatique des Connaissances par Systèmes d'Inférence Floue Hiérarchisés. - In *Actes du congrès LFA'01, Rencontres Francophones sur la Logique Floue et ses Applications*, pp. 105-111, 2001.

Bibliographie

[1] Site sur les machines à vecteurs supports : http://www.kernel-machines.org/.

[2] Ordre & désordre. – *La Recherche*, (9), novembre-décembre 2002. – Hors série.

[3] Selim Aksoy et Robert M. Haralick. – Probabilistic vs. geometric similarity measures for image retrieval. – In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, volume 2, pp. 357–362, 2000.

[4] Luís A. Alexandre, Aurélio C. Campilho, et Mohamed Kamel. – On combining classifiers using sum and product rules. – *Pattern Recognition Letters*, (22):1283–1289, 2001.

[5] Kamal M. Ali et Michael J. Pazzani. – Error reduction through learning multiple descriptions. – *Machine Learning*, 24(3):173–202, 1996.

[6] F. Alimoğlu et E. Alpaydin. – Combining multiple representations for pen-based handwritten digit recognition. – *ELEKTRIK: Turkish Journal of Electrical Engineering and Computer Sciences*, 9(1):1–12, 2001.

[7] F.M. Alkoot et J. Kittler. – Experimental evaluation of expert fusion strategies. – *Pattern Recognition Letters*, 20:1361–1369, 1999.

[8] F.M. Alkoot et J. Kittler. – Modified product fusion. – *Pattern Recognition Letters*, 23:957–965, 2002.

[9] E. Alpaydin, C. Kaynak, et F. Alimoğlu. – Cascading multiple classifiers and representations for optical and pen-based handwritten digit recognition. – In *Proc. of the 7th International Workshop on Frontiers in Handwriting Recognition*, pp. 453–462, 2000.

[10] Hakan Altinçay et Mübeccel Demireckler. – Why does output normalization create problems in multiple classifier system? – In *Proc. of the 16th International Conference on Pattern Recognition (ICPR-2002)*, volume 2, pp. 775–778, 2002.

[11] M R. Anderberg. – *Cluster Analysis for Applications.* – Academic Press, 1973.

[12] Jacques André et Marie-Anne Chabin (édité par). – *Les documents anciens, Document numérique*, volume 3. – Hermès, 1999.

[13] Éric Anquetil. – *Modélisation et reconnaissance par la logique floue : application à la lecture automatique en-ligne de l'écriture manuscrite omni-scripteur.* – Thèse de PhD, Université de Rennes 1, 1997.

[14] Éric Anquetil et Hélène Bouchereau. – Integration of an on-line handwriting recognition system in a smart phone device. – In *sixteenth IAPR International Conference on Pattern Recognition (ICPR 2002)*, volume 3, pp. 192–195, 2002.

[15] Eric Anquetil, Bertrand Coüasnon, et Frédéric Dambreville. – A symbol classifier able to reject wrong shapes for document recognition systems. – In *Graphics Recognition, Recent Advances*, *Lecture Notes in Computer Science*, volume 1941, pp. 209–218. Springer-Verlag, 2000.

[16] Éric Anquetil et Guy Lorette. – Automatic generation of hierarchical fuzzy classification systems based on explicit fuzzy rules deduced from possibilistic clustering: Application to on-line handwritten character recognition. – In *Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'96)*, pp. 259–264, 1996.

[17] Chidanand Apté et Sholom Weiss. – Data mining with decision trees and decision rules. – *Future Generation Computer Systems*, 13(2-3):197–210, 1997.

[18] Dennis Bahler et Laura Navarro. – Combining heterogeneous sets of classifiers: Theorical and experimental comparison of methods. – under review, http://citeseer.nj.nec.com/473779.html.

[19] Dennis Bahler et Laura Navarro. – Methods for combining heterogeneous sets of classifiers. – In *Proc. of the 7th National Conference on Artificial Intelligence (AAAI 2000), Workshop on New Research Problems for Machine Learning*, 2000. – http://citeseer.nj.nec.com/470241.html.

[20] A. Bellili, M. Gilloux, et P. Gallinari. – Reconnaissance de chiffres manuscrits par un système hybride mlp-svm. – In *Actes du 13e Congrès Francophone de Reconnaissance des Formes et d'Intelligence Artificielle (RFIA 2002)*, volume 3, pp. 761–769, 2002.

[21] Hugues Bersini, Giancula Bontempi, et Christine Decaestecker. – Comparing rbf and fuzzy inference systems on theorical and practical basis. – In *Internation Conference on Artificial Neural Networks (ICANN'95)*, volume 1, pp. 169–174, 1995.

[22] James C. Bezdek. – *Pattern recognition with fuzzy objective function algorithms.* – Plenum Press, 1981.

[23] Christopher M. Bishop. – *Neural Networks for Pattern Recognition.* – Oxford University Press, 1995.

[24] Isabelle Bloch. – Incertitude, imprécision et additivité en fusion de données : point de vue historique. – *Traitement du Signal*, 13(4):267–288, 1996.

[25] Isabelle Bloch. – Information combination operators for data fusion: A comparative review with classification. – *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 26(1):52–67, 1996.

[26] Bernadette Bouchon-Meunier. – *La logique floue et ses applications.* – Addison-Wesley, 1995.

[27] Bernadette Bouchon-Meunier et Christophe Marsala. – *Logique floue, principes, aide à la décision, Traité IC2 : Information - Commande - Communication.* – Traité IC2 : Information - Commande - Communication. Hermès-Lavoisier, 2003.

[28] Bernadette Bouchon-Meunier et Christophe Marsala. – *Traitements de données complexes et commande en logique floue*, *Traité IC2 : Information - Commande - Communication.* – Traité IC2 : Information - Commande - Communication. Hermès-Lavoisier, 2003.

[29] Leo Breiman. – Bagging predictors. – *Machine Learning*, 24(2):123–140, 1996.

[30] Leo Breiman, Jerome Friedman, Richard Olshen, et Charles Stone. – *Classification and Regression Trees.* – Wadsworth Int. Group, 1984.

[31] Carla E. Brodley et Paul E. Utgoff. – Multivariate decision trees. – *Machine Learning*, 19(1):45–77, 1995.

[32] D.S. Broomhead et D. Lowe. – Multivariable functional interpolation and adaptative networks. – *Complex systems 2*, pp. 321–355, 1988.

[33] Wray Buntine et Tim Niblett. – A further comparison of splitting rules for decision-tree induction. – *Machine Learning*, 8(1):75–85, 1992.

[34] Christopher J. C. Burges. – A tutorial on support vector machines for pattern recognition. – *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[35] Robin L.P. Chang et Theodosios Pavlidis. – Fuzzy decision tree algorithms. – *IEEE Transactions on Systems, Man and Cybernetics*, 7(1):28–35, 1977.

[36] Shyi-Ming Chen et Shih-Yirng Lin. – A new method for constructing fuzzy decision trees and generating fuzzy classification rules from training examples. – *Cybernetics and Systems*, 31(7):763–85, 2000.

[37] Kevin J. Cherkauer et Jude W. Shavlik. – Growing simpler decision trees to facilitate knowledge discovery. – In *KDD-96 Proceedings. Second International Conference on Knowledge Discovery and Data Mining*, pp. 315–318, 1996.

[38] Krzysztof Cios, Witold Pedrycz, et Roman Swiniarski. – *Data Mining Methods for Knowledge Discovery.* – Kluwer Academic Publishers, 1998.

[39] Krzysztof Cios, Witold Pedrycz, et Roman Swiniarski. – *Data Mining Methods for Knowledge Discovery*, chap. 8, p. 384. – Kluwer Academic Publishers, 1998.

[40] Bertrand Coüasnon et Jean Camillerapp. – Dmos, une méthode générique de reconnaissance de documents : évaluation sur 60 000 formulaires du 19e siècle. – In *ctes du Colloque International Francophone sur l'Écrit et le Document (CIFED'02)*, pp. 225–234, 2002.

[41] R. Collobert et S. Bengio. – Svmtorch: Support vector machines for large-scale regression problems. – *Journal of Machine Learning Research*, 1:143–160, 2001.

[42] Gusnard de Ventabert. – Représentation et exploitation électroniques de documents anciens (textes et images. – *Document numérique*, 3(1-2):57–73, 1999.

[43] Rameswar Debnath et Haruhisa Takakashi. – Learning capability: Classical rbf network vs. svm with gaussian kernel. – In *Developments in Applied Artificial Intelligence, 15th International Conference on Industrial and Engineering. Applications of Artificial Intelligence and Expert Systems, IEA/AIE 2002*, *LNAI 2358*, LNAI 2358, pp. 293–302. Springer-Verlag, 2002.

[44] Thomas G. Dietterich. – Ensemble methods in machine learning. – In J. Kittler et F. Roli (édité par), *MCS 2000, LNCS 1857*, pp. 1–15. Springer-Verlag, 2000.

[45] Chris Ding et Xiaofeng He. – Cluster merging and splitting in hierarchical clustering algorihtms. – In *Proceedings of the IEEE International Conference on Data Mining (ICDM'02)*, pp. 139–146, 2002.

[46] Didier Dubois et Henri Prade. – On data summarization with fuzzy sets. – In *Fifth IFSA Congress*, pp. 465–468, 1993.

[47] Didier Dubois et Henri Prade. – La problématique scientifique du traitement de l'information. – *Revue I3: Information - Interaction - Intelligence*, 1(2), 2001. – http://www.revue-i3.org/.

[48] Richard O. Duda et Peter E. Hart. – *Pattern Classification and Scene Analysis.* – Wiley, 1973.

[49] Richard O. Duda, Peter E. Hart, et David G. Stork. – *Pattern Classification.* – Wiley-Interscience, 2 édition, 2001.

[50] Richard O. Duda, Peter E. Hart, et David G. Stork. – *Pattern Classification*, chap. 10, pp. 542–548. – Wiley-Interscience, 2 édition, 2001.

[51] Robert P.W. Duin. – The combining classifier: to train or not to train? – In *Proc. of the 16th International Conference on Pattern Recognition (ICPR-2002)*, *Lecture Notes in Computer Sciences*, volume 2, pp. 765–770, 2002.

[52] Robert P.W. Duin et David M.J. Tax. – Experiments with classifier combining rules. – In J. Kittler et F. Roli (édité par), *MCS 2000*, *Lecture Notes in Computer Sciences*, volume 1857, pp. 16–29. Springer-Verlag, 2000.

[53] J. C. Dunn. – A fuzzy relative of the isodata process and its use in detecting compact, well separated clusters. – *Journal of Cybernetics*, 3:32–57, 1974.

[54] Theodoros Evgeniou et Massimiliano Pontil. – Support vector machines with clustering for training with very large datasets. – In *Methods and Applications of Artificial Intelligence, Second Hellenic Conference on AI (SETN 2002)*, *LNAI 2308*, LNAI 2308, pp. 346–354. Springer-Verlag, 2002.

[55] U. M. Fayyad et K. B. Irani. – On the handling of continuous-valued attributes in decision tree generation. – *Machine Learning*, 8:87–102, 1992.

[56] Usama Fayyad, Gregory Piatetsky-Shapiro, et Padhraic Smyth. – From data mining to knowledge discovery in databases. – *AI Magazine*, 17(3):37–54, 1996.

[57] Usama Fayyad, Gregory Piatetsky-Shapiro, et Padhraic Smyth. – Knowledge discovery and data mining: Towards a unifying framework. – In *Proc. of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pp. 82–88, 1996.

[58] W. Frawley, G. Piatetsky-Shapiro, et C. Matheus. – *Knowledge discovery in databases*, chap. Knowledge discovery in databases: An overview, pp. 1–27. – AAAI/MITPress, 1991.

[59] Yoav Freund et Llew Mason. – The alternating decision trees learning algorithm. – In *The Sixteenth International Conference on Machine Learning*, pp. 124–133, 1999.

[60] Yoav Freund et Robert E. Schapire. – Experiments with a new boosting algorithm. – In *International Conference on Machine Learning*, pp. 148–156, 1996.

[61] I. Gath et A. B. Geva. – Unsupervised optimal fuzzy clustering. – In *Proceedings of the IEEE Transaction on Pattern Analysis and Machine Intelligence*, volume 11, pp. 773–781, 1989.

[62] Stéphane Gentric. – *Architecture neuronale évolutive : application à la reconnaissance et à la segmentation automatique de l'écriture.* – Thèse de PhD, Université Pierre et Marie Curie, 1999.

[63] G. Giacinto et F. Roli. – Dynamic classifier selection based on multiple classifier behaviour. – *Pattern Recognition*, 34(9):179–181, 2001.

[64] Nicola Giusti, Francesco Masulli, et Alessandro Sperduti. – Theorical and experimental analysis of a two-stage system for classification. – *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):893–904, 2002.

[65] J.M. Gloger, A. Kaltenmeier, E. Mandler, et L. Andrews. – Reject management in a handwriting recognition system. – In *Proc. of the 4th International Conference on Document Analysis and Recognition (ICDAR'97)*, volume 2, pp. 556–559, 1997.

[66] Pierre Yves Glorennec. – *Algorithmes d'apprentissage pour systèmes d'inférence floue.* – Hermès, 1999.

[67] D.E. Goldberg. – *Genetic Algorithms in search, optimization, and machine learning.* – Addison-Wesley, 1989.

[68] Marina Guetova, Steffen Hölldobler, et Hans-Peter Störr. – Incremental fuzzy decision trees. – In *KI 2002, LNAI*, volume 2479, pp. 67–81. Springer-Verlag, 2002.

[69] H. Guo et S.B. Gelfand. – Classification trees with neural network feature extraction. – *IEEE Transactions on Neural Networks*, 3(6):923–33, 1992.

[70] E. E. Gustafson et W. C. Kessel. – Fuzzy clustering with a fuzzy covariance matrix. – In *Proceedings of the IEEE-CDC*, volume 2, pp. 761–766, 1979.

[71] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, et S. Janet. – UNIPEN project of on-line data exchange and recognizer benchmarks. – In *Proceedings of the 12th ICPR*, pp. 29–33, 1994.

[72] Isabelle Guyon, John Makhoul, Richard Schwartz, et Vladimir Vapnik. – What size test set gives good error rate estimates? – *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):52–64, 1998.

[73] H.C.Yau et M.T.Manry. – Automatic determination of reject thresholds in classifers employing discriminant functions. – *IEEE Transactions on Signal Processing*, 40(3):711–713, 1992.

[74] Tin Kam Ho. – *A Theory of Multiple Classifier Systems And Its Application to Visual Word Recognition.* – Thèse de PhD, Faculty of the Graduate School of State University of New York, 1992.

[75] Tin Kam Ho. – Data complexity analysis for classifier combination. – In *Proceedings of the 2nd International Workshop on Multiple Classifier Systems*, pp. 53–67, 2001.

[76] Tin Kam Ho. – Multiple classifier combination: Lessons and next steps. – In A. Kandel et H. Bunke (édité par), *Hybrid Methods in Pattern Recognition*, pp. 171–198. World Scientific, 2002.

[77] Tin Kam Ho et Mitra Basu. – Complexity measures of supervised classification problems. – *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):289–300, 2002.

[78] K. Hornik. – Approximation capabilities of multilayer feedforward networks. – *Neural Networks*, 4:251–257, 1991.

[79] C. Hsu et C. Lin. – *A comparison of methods for multi-class support vector machines*, 2001.

[80] Y. S. Huang et Ching Y. Suen. – A method of combining multiple experts for the recognition of unconstrained handwritten numerals. – *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(1):90–94, 1995.

[81] K. R. Ianakiev et V. Govindaraju. – Improvement of recognition accuracy using 2-stage classification. – In L.R.B. Schomaker et L.G. Vuurpijl (édité par), *Proc. of the Seventh International Workshop on Frontiers in Handwriting Recognition*, pp. 153–165, 2000.

[82] Andreas Ittner et Michael Schlosser. – Discovery of relevant new features by generating non-linear decision trees. – In *KDD-96 Proceedings. Second International Conference on Knowledge Discovery and Data Mining*, pp. 108–113, 1996.

[83] Ø ivind Due Trier, Anil K. Jain, et Torfinn Taxt. – Feature extraction methods for character recognition - a survey. – *Pattern Recognition*, 29(4):641–662, 1996.

[84] Robert A. Jacobs, Michael I.Jordan, Steven J. Nowlan, et Geoffrey E. Hinton. – Adaptative mixtures of local experts. – *Neural Computation*, 3(1):79–87, 1991.

[85] René Jager. – *Fuzzy Logic in Control.* – Thèse de PhD, Delft University of Technology, Department of Electrical Engineering, Control Laboratory, 1995.

[86] A. K. Jain et R. C.Dubes. – *Algorithms for clustering data.* – Prentice Hall, 1988.

[87] J.-S. Roger Jang et C.-T. Sun. – Functional equivalence between radial basis function networks and fuzzy inference systems. – *IEEE Transactions on Neural Networks*, 4(1):156–159, 1993.

[88] Cezary Z. Janikow. – A genetic algorithm method for optimizing fuzzy decision trees. – *Information Sciences*, 89(3-4):275–296, 1996.

[89] Cezary Z. Janikow. – Fuzzy decision trees: Issues and methods. – *IEEE Transactions on Systems, Man and Cybernetics*, 28:1–14, 1998.

[90] Cezary Z. Janikow et M. Fajfer. – Fuzzy partitioning with fid3.1. – In *18th International Conference of the North American Fuzzy Information Processing Society - NAFIPS*, pp. 476–71, 1999.

[91] Jane Yung jen Hsu et I-Jen Chiang. – Fuzzy classification trees. – In *Ninth International Symposium on Artificial Intelligence in Joint Cooperation with the Sixth International Conference on Industrial Fuzzy Control and Intelligent Systems*, pp. 431–8, 1996.

[92] M. Kearns et Y. Mansour. – A fast, bottom-up decision tree pruning algorithm with near-optimal generalization. – In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML'98)*, pp. 269–277, 1998.

[93] Eunjun Kim, Wooju Kim, et Yillbyung Lee. – Classifier fusion using local confidence. – In M.-S. Hacid et al. (édité par), *ISMIS 2002*, *Lecture Notes in Artificial Intelligence*, volume 2366, pp. 583–591. Springer-Verlag, 2002.

[94] Jin Ho Kim, Kye Kyung Kim, et Ching Y. Suen. – Hybrid schemes of homogeneous and heterogeneous classifiers for cursive word recognition. – In L.R.B.Schomaker et L.G.Vuurpijl (édité par), *Proc. of the Seventh International Workshop on Frontiers in Handwriting Recognition*, pp. 433–442, 2000.

[95] Josef Kittler, Mohamad Hatef, Robert P.W. Duin, et Jiri Matas. – On combining classifiers. – *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

[96] Ron Kohavi. – A study of cross-validation and bootstrap for accuracy estimation and model selection. – In *International Joint Conference on Artificial Intelligence (IJCAI-95)*, pp. 1137–1145, 1995.

[97] Ravi Kothari et Ming Dong. – Decision trees for classification: A review and some new results. – soumis à World Scientific, 2000.

[98] Raghu Krishnapuram. – Fuzzy clustering methods in computer vision. – In *Proceedings of the 1st European Congress on Fuzzy and Intelligent Technologies (EUFIT'93)*, pp. 720–730, 1993.

[99] Raghu Krishnapuram. – Generation of membership functions via possibilistic clustering. – In *IEEE World congress on computational intelligence*, pp. 902–908, 1994.

[100] Raghu Krishnapuram et James M. Keller. – A possibilistic approach to clustering. – *IEEE Transactions on Fuzzy Systems*, 1(2):98–110, 1993.

[101] Raghu Krishnapuram et James M. Keller. – The possibilistic c-means algorithm: Insights and recommendations. – *IEEE Transactions on Fuzzy Systems*, 4(3):385–393, 1996.

[102] L. I. Kuncheva, C. J. Whitaker, C. A. Shipp, et R. P.W. Duin. – Is independence good for combining classifiers? – In *Proc. of the 15th International Conference on Pattern Recognition (ICPR-2000)*, volume 2, pp. 168–171, 2000.

[103] Ludmila I. Kuncheva. – Switching between selection and fusion in combining classifiers: An experiment. – *IEEE Transactions on Systems, Man and Cybernetics–Part B: Cybernetics*, 32(2):146–156, 2002.

[104] Ludmila I. Kuncheva. – A theorical study on six classifier fusion strategies. – *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):281–286, 2002.

[105] Ludmila I. Kuncheva et Christopher J. Whitaker. – Measures of diversity in classifier ensembles. – *Machine Learning*, 51:181–207, 2003.

[106] Louisa Lam. – Classifier combinations: Implementations and theorical issues. – In J.Kittler et F.Roli (édité par), *Mulitple Classifier Systems*, *Lecture Notes in Computer Sciences*, volume 1857, pp. 77–86. Springer-Verlag, 2000.

[107] Louisa Lam et Ching Y. Suen. – Application of majority voting to pattern recognition: An analysis of its behavior and performance. – *IEEE Transactions*

on Systems, Man and Cybernetics–Part A: Systems and Humans, 27(5):553–568, 1997.

[108] Mark Last, Oded Maimon, et Einat Minkov. – Improving stability of decision trees. – *Int. Journal of Pattern Recognition and Artificial Intelligence*, 16(2):145–159, 2002.

[109] V. Di Lecce, G. Dimauro, A. Guerriero, S. Impedovo, G. Pirlo, et A. Salzo. – Classifier combination: The role of a-priori knowledge. – In *Proc. of the 7th International Workshop on Frontiers in Handwriting Recognition*, pp. 143–152, 2000.

[110] Régis Lengellé. – *Décision et reconnaissance des formes en signal*, *Traitement du Signal et de l'Image*, chap. 6. – Traitement du Signal et de l'Image. Hermès, Lavoisier, 2002.

[111] Israel-Cesar Lerman et Joaquim F.P. Da Costa. – Coefficients d'association et variables a tres grand nombre de categories dans les arbres de décision ; application a l'identification de la structure secondaire d'une protéine. – Rapport technique n° 2803, INRIA, 1996.

[112] Levenshtein. – Binary codes capable of correcting deletions, insertions and reversals. – *Soviet Physics Doklady*, 10:707–710, 1966.

[113] T. W. Liao, Aivars K. Celmins, et Robert J. Hammel II. – A fuzzy c-means variant for the generation of fuzzy term sets. – *Fuzzy Sets and Systems*, (135):241–257, 2003.

[114] Tjen-Sien Lim, Wei-Yin Loh, et Yu-Shan Shih. – A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. – *Machine Learning*, 40(3):203–228, 2000.

[115] W.-Y. Loh et Y.-S. Shih. – Split selection methods for classification trees. – *Statistica Sinica*, 7:815–840, 1997.

[116] J B. MacQueen. – Some methods for classification and analysis of multivariate observations. – In *Proceedings of the 5th Berkley Symposium on Mathematical Statistics and Probability*, pp. 281–297, 1967.

[117] Y. Mansour. – Pessimistic decision tree pruning based on tree size. – In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML'98)*, pp. 195–201, 1997.

[118] C. Marsala et B. Bouchon-Meunier. – Choice of a method for the construction of fuzzy decision trees. – In *Proc. of the Int. Conf. on Fuzzy Systems , FUZZ-IEEE'03*, pp. 584–589, 2003.

[119] C. Marsala et B. Bouchon-Meunier. – Measures of discrimination for the construction of fuzzy decision trees. – In *Proc. of Fuzzy Information Processing (FIP'03)*, pp. 709–714, 2003.

[120] C. Marsala, B. Bouchon-Meunier, et A. Ramer. – Hierarchical model for discrimination measures. – In *Proc. of the eight IFSA'99 World Congress*, pp. 339–343, 1999.

[121] Christophe Marsala. – *Apprentissage inductif en présence de données imprécises : construction et utilisation d'arbres de décision flous.* – Thèse de doctorat, Université de Paris 6, 1998.

[122] J. Kent Martin. – An exact probability metric for decision tree splitting and stopping. – *Machine Learning*, 28(2-3):257–91, 1997.

[123] Ujjwal Maulik et Sanghamitra Bandyopadhyay. – Performance evaluation of some clustering algorithms and validity indices. – *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1650–1654, 2002.

[124] Eddy Mayoraz et Ethem Alpaydin. – Support vector machines for multi-class classification. – In *Proceedings of the International Workshop on Artifical Neural Networks (IWANN99)*, pp. 833–842, 1999.

[125] W.S. McCulloch et W. Pitts. – A logical calculus of the ideas of immanent in nervous activity. – *Bulletin of Mathematical Biophysics 5*, pp. 115–133, 1943.

[126] John Mingers. – An empirical comparison of selection measures for decision tree induction. – *Machine Learning*, 3:319–342, 1989.

[127] Bahram Moobed. – *Combinaison de classifieurs, une nouvelle approche.* – Thèse de doctorat, Université de Paris XI Orsay, 1996.

[128] J. Moody et C.J. Darken. – Fast learning in networks of locally-tuned processing units. – *Neural Computation 1*, pp. 281–294, 1989.

[129] Sreerama K. Murthy. – Automatic construction of decision trees from data: a multi-disciplinary survey. – *Data Mining and Knowledge Discovery 2*, (4):345–389, 1998.

[130] Cristina Olaru et Louis Wehenkel. – A complete fuzzy decision tree technique. – *Fuzzy Sets and Systems*, (138):221–254, 2003.

[131] Edgar Osuna, Robert Freund, et Federico Girosi. – Support vector machines: Training and applications. – Rapport technique n° AIM-1602, 1997.

[132] Nikhil R. Pal, Kuhu Pal, et James C. Bezdk. – A mixed c-means clustering model. – In *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'97)*, pp. 11–21, 1997.

[133] Marc Parizeau, Alexandre Lemieux, et Christian Gagné. – Character recognition experiments using unipen data. – In *Proc. of the International Conference on Document Analysis and Recognition (ICDAR'01)*, pp. 481–485, 2001.

[134] J. Park et I.W. Sandberg. – Approximation and radial basis function networks. – *Neural Computation*, 5(2):305–316, 1993.

[135] Émilie Poisson, Christian Viard-Gaudin, et Pierre-Michel Lallican. – Combinaison de réseaux de neurones à convolution pour la reconnaissance de caractères manuscrits en-ligne. – In *Conférence Fédérative sur l'Écrit et le Document (CIFED'02)*, pp. 315–324, 2002.

[136] L. Prevost, S. Gentric, et M. Milgram. – Model generation and cooperation in on-line omni-writer handwriting recognition. – In *Proc. of the Third International Conference on Information Fusion*, volume 2, pp. 3–8, 2000.

[137] L. Prevost et M. Milgram. – Automatic allograph selection and multiple expert classification for totally unconstrained handwritten character recognition. – In *International Conference on Pattern Recognition*, pp. 381–383, 1998.

[138] L. Prevost et M. Milgram. – Modelizing character allographs in omni-scriptor frame: a new non-supervised clustering algorithm. – *Pattern Recognition Letters*, (21):295–302, 2000.

[139] L. Prevost, A. Moises, C. Michel-Sendis, L. Oudot, et M. Milgram. – Combining model-based and discriminative classifiers: application to handwritten character recognition. – In *Internation Conference on Document Analysis and Recognition (ICDAR'03)*, volume 1, pp. 31–35, 2003.

[140] Jan Puzicha, Yossi Rubner, Carlo Tomasi, et Joachim M. Buhmann. – Empirical evaluation of dissimilarity measures for color and texture. – In *Proceedings of the IEEE International Conference on Computer Vision (ICCV'99)*, pp. 1165–1173, 1999.

[141] John Ross Quinlan. – Induction of decision trees. – *Machine Learning*, 1:81–106, 1986.

[142] John Ross Quinlan. – *C4.5 Programs for machine learning.* – Morgan Kaufmann, 1993.

[143] John Ross Quinlan. – Improved use of continuous attributes in c4.5. – *Journal of Atificial Intelligence Research*, 4:77–90, 1996.

[144] Mohammed Ramdani. – *Système d'induction formelle à base de connaissances imprécises.* – Thèse de doctorat, Université de Paris 6, 1994.

[145] Eugene H. Ratzlaff. – A scanning n-tuple classifier for online recognition of handwritten digits. – In *Internation Conference on Document Analysis and Recognition (ICDAR'01)*, pp. 18–22, 2001.

[146] Eugene H. Ratzlaff. – Methods, report and survey for the comparison of divers isolated character recognition results on the unipen database. – In *Internation Conference on Document Analysis and Recognition (ICDAR'03)*, volume 1, pp. 623–628, 2003.

[147] Rifqi et S. Monties. – Fuzzy prototypes for fuzzy data mining. – In O. Pons, A. Vila, et J. Kacprzyk (édité par), *Knowledge Management in Fuzzy Databases.* Physica-Verlag, 2000.

[148] Fabio Roli et Giogio Giacinto. – *Hybrid Methods in Pattern Recognition*, chap. Design of Multiple Classifier Systems, pp. 199–226. – World Scientific, 2002.

[149] Enrique H. Ruspini. – A new approach to clustering. – *Information and Control*, 15:22–32, 1969.

[150] S. Rasoul Safavian et David Landgrebe. – A survey of decision tree classifier methodology. – *IEEE Transactions on Systems, Man, and Cybernetics*, 21(3):660–674, 1991.

[151] A. Sankar et R.J. Mammone. – Optimal pruning of neural tree networks for improved generalization. – In *IJCNN-91-Seattle: International Joint Conference on Neural Networks*, volume 2, pp. 219–24, 1991.

[152] W. S. Sarle. – Neural networks and statistical models. – In *Proceedings of the Nineteenth Annual SAS Users Groups International Conference*, pp. 1538–1550, 1994.

[153] B. Schölkopf, K.-K. Sung, C. J. C. Burges, F. Girosi, P. Niyogi, T. Poggio, et V. Vapnik. – Comparing support vector machines with Gaussian kernels to radial basis function classiers. – *IEEE Transactions on Signal Processing*, 45(11):2758–2765, 1997.

[154] Jürgen Schürmann. – *Pattern classification: a unified view of statistical and neural approaches*, chap. 10. – Wiley-Interscience, 1996.

[155] Glenn A Shafer. – *Mathematical Theory of Evidence.* – Princeton University Press, 1976.

[156] Claude E. Shannon. – *The mathematical theory of communication.* – University of Illinois Press, 1948.

[157] Yu-Shan Shih. – Families of splitting criteria for classification trees. – *Journal of Statistics and Computing*, 9(4):309–315, 1999.

[158] So Young Sohn. – Meta analysis of classification algorithms for pattern recognition. – *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11):1137–1144, 1999.

[159] William M. Spears. – *Evolutionary Algorithms, The Role of Mutation and Recombination.* – Springer, 2000.

[160] S.R.Waterhouse et A.J. Robinson. – Classification using hierarchical mixtures of experts. – In *Proc. of the IEEE Workshop on Neural Networks for Signal Processing IV*, pp. 177–186, 1994.

[161] Ching Y. Suen et Louisa Lam. – Multiple classifier combination methodologies for different output levels. – In *MCS 2000*, *Lecture Notes in Computer Sciences*, volume 1857, pp. 52–66. Springer-Verlag, 2000.

[162] H. Tanaka, T. Okuda, et K. Asai. – Fuzzy information and decision in statistical model. – In *Advances in Fuzzy Set Theory and Applications*, pp. 303–320. 1979.

[163] David M. J. Tax et Robert P. W. Duin. – Using two-class classifiers for multiclass classification. – In *Proc. of the 16th International Conference on Pattern Recognition*, volume 2, pp. 124–127, 2002.

[164] David M. J. Tax, Martijn van Breukelen, Robert P. W. Duin, et Josef Kittler. – Combining multiple classifiers by averaging or by multiplying? – *Pattern Recognition*, 33:1475–1485, 2000.

[165] G. Valentini et F. Masulli. – Ensembles of learning machines. – In *Neural Nets WIRN Vietri-2002*, *Lecture Notes in Computer Sciences*, volume 2486, pp. 3–19. Springer-Verlag, 2002.

[166] V. Vapnik. – *The Nature of Statistical Learning Theory.* – Springer-Verlag, 1995.

[167] W. N. Venables et B. D. Ripley. – *Modern Applied Statistics with S-plus.* – Springer, 2e édition, 1997.

[168] C. Viard-Gaudin, P. M. Lallican, S. Knerr, et P. Binter. – The IRESTE on/off (IRONOFF) dual handwriting database. – In *Proceedings of the Fifth International*

Conference on Document Analysis and Recognition (ICDAR'99), pp. 455–458, 1999.

[169] Louis G. Vuurpijl et Lambert R.B. Schomaker. – Two-stage character classification: A combined approach of clustering and support vector classifiers. – In L.R.B. Schomaker et L.G. Vuurpijl (édité par), *Proc. of the Seventh International Workshop on Frontiers in Handwriting Recognition*, pp. 423–432, 2000.

[170] R. Wagner et M.Fisher. – The string-to-string correction problem. – *Journal of the ACM*, 12(1):168–173, 1974.

[171] W. Wang, P. Jones, et D. Partridge. – Diversity between neural networks and decision trees for building multiple classifier systems. – In *Multiple Classifier Systems (MCS 2000)*, *Series Lecture Notes in Computer Sciences*, volume 1857, pp. 240–249. Springer-Verlag, 2000.

[172] X.-Z. Wang, D.S. Yeung, et E.C.C. Tsang. – A comparative study on heuristic algorithms for generating fuzzy decision trees. – *IEEE Transactions on Systems, Man and Cybernetics–Part B: Cybernetics*, 31(2):215–226, 2001.

[173] Donald K. Wedding et Krzystof J. Cios. – Certainty factors versus parzen windows as reliability measures in rbf networks. – *Neurocomputing*, 19(1-3):151–165, 1998.

[174] Michael P. Windham. – Cluster validity for the fuzzy c-means clustering algorithm. – *IEEE Trnasactions on Pattern Analysis and Machine Intelligence*, 4(4):1982, 1982.

[175] D. H. Wolpert. – Stacked generalization. – *Neural Networks*, 5:241–259, 1992.

[176] Xuanli Lisa Xie et Gerardo Beni. – A validity measure for fuzzy clustering. – *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8):841–847, 1991.

[177] Lei Xu, Adam Krzyżak, et Ching Y. Suen. – Methods of combining multiple classifiers and their applications to handwriting recognition. – *IEEE Transactions on Systems, Man and Cybernetics*, 22(3):418–435, 1992.

[178] Qizhi Xu, Louisa Lam, et Ching Y. Suen. – Automatic segmentation and recognition system for handwritten dates on canadian bank cheques. – In *Proc. of the 7th International Conference on Document Analysis and Recognition (ICDAR'03)*, volume 2, pp. 704–708, 2003.

[179] Yufei Yuan et Michael J. Shaw. – Induction of fuzzy decision trees. – *Fuzzy sets and systems*, (69):125–139, 1995.

[180] Lofti A. Zadeh. – Fuzzy sets. – *Information and Control 8*, pp. 338–353, 1965.

[181] Lofti A. Zadeh. – Probability measures of fuzzy events. – *Jour. Math. Analysis and Appl.*, 23:421–427, 1968.

[182] Lofti A. Zadeh. – Outline of a new approach to the analysis of complex systems and decision processes. – *IEEE Trans. on Systems, Man, and Cybernetics*, 3:28–44, 1973.

[183] Lofti A. Zadeh. – Fuzzy sets as a basis for a theory of possibility. – *Fuzzy Sets and Systems 1*, pp. 3–28, 1978.

[184] Lofti A. Zadeh. – Soft computing and fuzzy logic. – *IEEE Software*, pp. 48–56, 1994.

[185] N. Zahid, M. Limouri, et A. Essaid. – A new cluster-validity for fuzzy clustering. – *Pattern Recognition*, (32):1089–1097, 1999.

[186] J. Zeidler et M. Schlosser. – Fuzzy handling of continuous attributes in decision trees. – In *ECML-95 Mlnet Familiarization Workshop "Statistics, Machine Learning and Knowledge Discovery in Databases"*, pp. 41–46, 1995.

[187] J. Zeidler et M. Schlosser. – Continuous-valued attributes in fuzzy decision trees. – In *6 th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pp. 395–400, 1996.

[188] Abdelkader Zighed, Ricco Rakotomalala, et Yves Kodratoff. – *Graphes d'induction : Apprentissage et Data Mining.* – Hermès, 2000.

Résumé :
Pour faciliter la mise au point de systèmes de reconnaissance de formes, nous proposons une méthodologie de classification visant à réunir un ensemble de propriétés rarement satisfaites dans une même approche : *performances*, *généricité*, *fiabilité*, *robustesse*, *compacité* et *interprétabilité*. Ce dernier point permet au concepteur d'adapter, de maintenir et d'optimiser le système plus facilement. L'approche proposée, centrée sur la notion de connaissances dans un classifieur, est entièrement guidée par les données. L'originalité réside notamment dans l'exploitation conjointe de connaissances *intrinsèques* et *discriminantes* extraites automatiquement et organisées sur deux niveaux pour bénéficier au mieux de leur complémentarité : le premier modélise les classes par des *prototypes flous* et le second effectue une discrimination des formes *similaires* par des *arbres de décision flous*. L'ensemble est formalisé par des *systèmes d'inférence floue* qui sont combinés pour la classification. Cette approche a conduit à la réalisation du système *Mélidis* qui a été validé sur plusieurs benchmarks dont des problèmes de reconnaissance de caractères manuscrits en ligne.

Mots clés : Reconnaissance de formes, logique floue, systèmes d'inférence floue, classification non supervisée, arbres de décision flous, combinaison de classifieurs, reconnaissance de formes manuscrites.

Abstract:
To facilitate the integration of pattern recognition systems in real applications, we present a new approach which aim is to combine properties that are rarely fully satisfied in the same recognition system. These properties are: *performances*, *generic architecture and learning*, *reliability*, *robustness*, *compactness* and *transparency*. This last point is particularly important to make the system easier to adapt, maintain and optimize for a given application. Our approach is totally data driven and focuses on knowledge properties in a classifier. The main originality comes from the cooperation of *intrinsic* and *discriminant* knowledge that are extracted and modelized automatically. They are organized in two levels: the first one models classes with *fuzzy prototypes* and the second one operates discrimination on shapes that have *similar intrinsic properties* using *fuzzy decision trees*. The system is entirely formalized by *fuzzy inference systems* that are combined for final classification. This approach led up to the implementation of the *Mélidis* system that has been evaluated on several benchmarks including handwritten character recognition.

Keywords: Pattern recognition, fuzzy logic, fuzzy inference systems, unsupervised learning, fuzzy decision trees, classifier combination, handwritten character recognition.