THÈSE de DOCTORAT de l'UNIVERSITÉ PARIS 6

Spécialité :

Informatique

présentée par

Christophe MARSALA

Pour obtenir le grade de
DOCTEUR de l'UNIVERSITÉ PARIS 6

Sujet de la thèse :

Apprentissage inductif en présence de données imprécises : construction et utilisation d'arbres de décision flous.

soutenue le mardi 6 janvier 1998

devant le jury composé de :

Mme Bernadette BOUCHON-MEUNIER	Directrice de thèse
M. Janusz KACPRZYK	Rapporteur
M. Louis WEHENKEL	Rapporteur
M. Didier DUBOIS	
M. Jean-Gabriel GANASCIA	
M. Mohammed RAMDANI	

Remerciements

En premier lieu, je tiens à faire part ici de toute ma reconnaissance envers Mme *Bernadette Bouchon-Meunier* qui a accepté de me diriger durant cette thèse. Pendant ces trois années, je me suis efforcé de tirer profit de tous les conseils qu'elle m'a donnés et je garde le sentiment d'avoir eu le meilleur professeur possible pour me former au métier de chercheur. Ses compétences et sa grande disponibilité m'ont permis de travailler dans des conditions tellement idéales que le soulagement de soutenir cette thèse s'accompagne du regret de l'avoir terminée.

Je remercie M. *Janusz Kacprzyk* d'avoir accepté, malgré un emploi du temps très chargé, d'être le rapporteur de ma thèse. Sa double compétence en logique floue et en apprentissage inductif en fait pour moi un juge incontournable pour mon travail.

Je remercie également M. *Louis Wehenkel* d'avoir accepté d'être le rapporteur de ma thèse. J'ai beaucoup apprécié les discussions que nous avons eues et les nombreuses remarques qu'il m'a faites sur mon travail. C'est à l'issue de ces discussions que j'ai pu m'apercevoir de tout ce qu'il me restait encore à faire dans le domaine des arbres de décisions flous.

Je suis extrêmement honoré que M. *Didier Dubois* ait accepté de juger mon travail. C'est à l'occasion d'une courte discussion que j'ai eu avec lui que j'ai compris ce qu'était l'enthousiasme et la passion du chercheur.

Je remercie M. *Jean-Gabriel Ganascia* d'avoir accepté de faire partie du jury de cette thèse. Mon premier contact avec les arbres de décision a eu lieu avec son cours de DEA, il est donc juste qu'il juge ainsi mon travail dans ce domaine.

Ma thèse doit beaucoup à M. *Mohammed Ramdani* car il est à l'origine de son sujet, je l'en remercie vivement. Il m'a d'abord supervisé durant mon stage de DEA où il m'a permis de développer mon goût pour les arbres de décision flous. Durant ma thèse, les séances de travail et les nombreuses discussions que nous avons eues, m'ont souvent permis de progresser plus rapidement et d'abandonner les idées faciles.

Je voudrais remercier ici tous ceux qui ont contribué, directement ou indirectement, à l'aboutissement de cette thèse.

Tout d'abord, je remercie *Marie-Claire Masbou* qui gère le réseau de machines SUN du LAFORIA et qui résout tous nos problèmes d'utilisateurs.

Les nombreuses discussions que j'ai eues avec *Florence d'Alché-Buc* m'ont beaucoup apporté et j'ai énormément apprécié ses encouragements tout au cours de mon existence laforienne.

Les copains thésards ont aussi leur part dans ce travail. Une thèse dure trois ans, trois années de vie commune avec d'autres thésards avec ses heurts mais surtout ses moments de joie inoubliables. La minuscule salle des thésards de la tour 46-00 est une indispensable fontaine de jouvence qui suffit bien souvent à redonner de l'énergie dans les moments les plus critiques. Merci donc aux "filles" : *Amel*, la petite *Anne*, *Ilham*, *Sawsan* (plus connue sous le pseudo de So-so la maileuse), ainsi qu'aux garçons *Georges*, *Gerson*, *Gerthory* et le Pratchettien *Mourad*. Comme autres thésards, j'ai une pensée particulière pour mes copains galériens de la rédaction *Madeleine*, *Myni* et *Nathalie (G.)* qui m'ont permis de vérifier que la solidarité des thésards en phase de rédaction n'est pas un mythe.

Je remercie aussi *Zahia* pour la relecture des premiers chapitres de ce manuscrit et les améliorations qu'elle m'a permis de leur apporter, ainsi que pour tous les commentaires qu'elle a su me faire durant ma thèse.

Je dois plus particulièrement de grandes discussions scientifiques, littéraires, pkdickiennes et aussi, il faut bien l'avouer, absurdes à *Speedy* un des exilés de la tour 45.

Avant d'être de l'autre côté de l'Atlantique, *Isabelle Moulinier* m'a beaucoup apporté, sa gentillesse et sa disponibilité nous manquent beaucoup ici, même si maintenant, c'est toujours agréable de savoir que son essence emailienne et talkienne flotte encore parmi nous...

Toute l'équipe LOFTI est une grande famille dirigée par Bernadette Bouchon-Meunier dont les réunions sont autant de discussions animées et fructueuses. Merci donc aux animateurs de ces réunions : *Andréas*, *Christophe (B.)*, *Leïla*, *Jannick*, et *Virginie*.

Plus proche pour moi, je ne remercierais jamais assez *Nathalie Aladenise* à qui je dois ma conversion au Pennac-isme et la découverte de Perec, de l'OULIPO et de la fabuleuse loi de Roubaud du croissant au beurre.

Et ma petite “jumelle” *Maria Rifqi* qui est tellement vivante que le labo a paru si terne quand elle n'était pas là ! Ce n'est pas un simple “merci” qui pourra jamais exprimer la joie que j'ai eu à la côtoyer depuis trois ans, mais plutôt le “Qu'un(e) véritable ami(e) est une douce chose...” d'un La Fontaine autrement plus loquace que moi.

Finalement, ce manuscrit n'aurait peut-être pas vu le jour avant l'an 2000 si je n'avais pas eu à subir les persécutions intensives mais amicales de *Nara Martini Bigolin*. En plus de ses diligentes relectures des chapitres de cette thèse, Nara est l'amie qui m'a le plus encouragé et supporté pendant cette rédaction et qui a su me pousser à en arriver au bout, tout cela à contre-courant de ma mauvaise humeur chronique de thésard en phase terminale... C'est sans aucun doute à elle que devait penser Pennac en écrivant ces mots : “Si vous voyez un être humain dans la foule, suivez-le, suivez-le !”.

Enfin, j'ai la chance d'avoir trois amis : *Blandine* et *Gilles Moutiers* et *Frédéric Juanéda* (mon Obiwan Kenobi personnel), à qui je dois beaucoup et qui m'ont toujours soutenus depuis des temps immémoriaux. En ces temps là, l'informatique n'était qu'un mot pour moi et pourtant, eux, ils étaient déjà là...

À mes parents à qui je dois tout.

Table des matières

Introduction

HAL: "I'm sorry Dave, I'm afraid I can't do that."

A. C. Clarke – *2001: a Space Odyssey*

Le but premier de l'Intelligence artificielle a été de donner à l'ordinateur des compétences de raisonnement proches des compétences humaines. Certes, la difficulté d'une telle entreprise a tendance à moduler la finalité de ce domaine. Si l'ordinateur ne peut pas obligatoirement réfléchir comme un esprit vivant, il faut au moins qu'il apprenne à gérer son environnement de la meilleure façon possible pour atteindre le but, la tâche, qu'il s'est fixé (ou qui lui a été donné).

Comme pour tout être vivant, la faculté de mémoriser et de réutiliser ses expériences est un des éléments clés qui permettront à l'ordinateur d'atteindre un stade complet d'autonomie proche des modèles fondateurs de l'Intelligence artificielle.

À l'heure actuelle, en Intelligence artificielle, une des techniques utilisées pour améliorer les capacités de résolution de problèmes d'un système informatique, est de lui inclure des connaissances sur le domaine qu'il doit gérer. Ces connaissances peuvent prendre diverses formes : des heuristiques de résolution, une base de règles...

L'acquisition de ces connaissances spécifiques à un domaine d'expertise particulier est donc une étape essentielle pour atteindre un raisonnement efficace. Ces connaissances spécifiques se décomposent en deux catégories, soit ce sont des connaissances dites *simples* (par exemple des faits, des mesures, des observations), soit elles sont dites *complexes* (par exemple des règles, des lois, des relations).

Les connaissances simples portant sur un phénomène sont directement accessibles par l'observation, *à l'œil nu* ou à l'aide d'un instrument de mesure, du phénomène en question. Par contre, le moyen usuel d'acquérir des connaissances complexes est de les obtenir auprès d'experts du domaine en question. Cela reste délicat car même si un tel expert existe, ce qui peut ne pas être toujours le cas, les connaissances complexes qu'il possède ne sont pas forcément explicites.

C'est pourquoi, les systèmes d'apprentissage inductif sont utilisés pour inférer des connaissances complexes à partir de connaissances simples. C'est un moyen de déduire, d'une façon autonome, une loi générale à partir de plusieurs faits connus ou observés.

Cependant, pour les systèmes d'apprentissage actuels, il est souvent difficile de gérer les connaissances simples imprécises ou vagues. L'imprécision et le flou des observations sont pourtant des caractéristiques naturelles de notre monde réel, il faut donc obligatoirement apprendre à les intégrer. Nous, humains, savons les gérer et en tirer parti, et les connaissances complexes en tiennent compte. Il est donc important qu'un système informatique acquière cette capacité afin d'atteindre une certaine autonomie de fonctionnement.

La théorie des sous-ensembles flous de Lotfi Zadeh (Zadeh, 1965) permet la formalisation et le traitement de connaissances imprécises ou vagues. Cela nous amène donc à envisager l'intégration de cette théorie dans les systèmes d'apprentissage inductif pour prendre en compte l'imprécision des connaissances simples.

Apprentissage inductif et base de règles

Pour l'apprentissage inductif d'un phénomène, on se place dans un domaine où l'on possède un ensemble d'exemples de cas résolus par des experts du domaine en question. Chaque élément de cet ensemble est représenté par un couple [description, classe] dans lequel la description est composée d'un ensemble de couples [attribut, valeur]. Ces couples sont les connaissances simples disponibles sur le phénomène en question, obtenues par l'observation du phénomène. De tels exemples constituent une *base d'apprentissage*, c'est grâce à eux que le système informatique généralise les connaissances complexes, les lois qui régissent ces connaissances simples. L'objectif est alors de trouver des règles générales qui permettraient de classer toute description du phénomène, connue ou inconnue.

La méthode d'induction que nous avons choisi repose sur la construction d'un arbre de décision. C'est une méthode qui a l'avantage d'être simple à mettre en œuvre. Le but de nos travaux n'est pas d'exhiber une méthode inductive meilleure que toutes les autres, mais d'étudier la meilleure façon de prendre en compte l'imprécision et le flou des données.

Un arbre de décision est un outil explicite et aisément expliquable. Une décision prise en suivant le cheminement dans un arbre de décision est justifiable aux yeux d'un expert du domaine qui peut ne pas être, de surcroît, un expert informatique. Cette structure de connaissances complexes est, en outre, très proche des structures de connaissances que nous manipulons. C'est aussi une base de règles qui est immédiatement explicite et pour laquelle le mode d'inférence, la déduction, est un mode de raisonnement très proche du mode de pensée humain.

Prise en compte de l'imprécis

Mais les données de notre monde réel sont souvent entachées d'imprécision. L'esprit humain est rompu aux techniques de prise en compte de telles connaissances imprécises sur son environnement. C'est pour lui une adaptation à son environnement acquise tout au long de son évolution.

De plus, l'utilisation de telles données imprécises pour déduire des lois générales est un gain important. Une donnée imprécise, floue, est déjà une généralisation d'un ensemble de données précises, c'est donc un aspect de la connaissance avec lequel il faut apprendre à composer.

L'autonomie des ordinateurs est à ce prix, il lui faut être capable de considérer tous les types de connaissances qui l'entourent.

Mieux utiliser les connaissances existantes

Ainsi, notre souci d'amélioration des méthodes inductives pour la prise en considération des données imprécises a pour objectif majeur de pouvoir utiliser toutes les connaissances existantes sur un domaine considéré.

Les environnements informatiques ne sont pas essentiellement l'environnement humain. Une base de données, un réseau de machines interconnectées, un ensemble de programmes, sont aussi des environnements informatiques dans lesquels un ordinateur doit être capable de se *mouvoir*. Un ordinateur doit pouvoir être autonome dans tout environnement informatique, il doit être capable d'extraire de nouvelles connaissances à partir de toutes les connaissances ou les données qui l'entourent. C'est même une nécessité *vitale* pour tout ordinateur de construire sa connaissance à partir de ces environnements qui l'entourent.

L'apprentissage inductif par arbres de décision est depuis longtemps un moyen d'inférer de nouvelles connaissances en reformulant un ensemble de données et en le synthétisant sous la forme d'une base de règles. Cette technique occupe maintenant une place de choix dans le domaine actuel de la *fouille de données*[1].

En introduisant dans cette méthode des outils de la théorie des sous-ensembles flous pour construire des arbres de décision flous capables de prendre en compte des données imprécises, nous possédons alors un puissant outil d'inférence de nouvelles connaissances à partir de tout type de données. Tous les types de connaissances existantes peuvent être alors exploités.

Description de nos travaux

Afin de prendre en compte les données imprécises, nous nous proposons d'étudier les algorithmes existants de construction d'arbres de décision et d'en retirer leurs paramètres génériques. La meilleure façon d'introduire les éléments de la théorie des sous-ensembles flous s'effectuera alors au niveau des paramètres concernés. Une fois la méthode de construction d'arbres de décision flous introduite, il faudra présenter une méthode d'utilisation de ces arbres flous pour la classification de nouveaux exemples. C'est sur cette partie qu'une étude de la robustesse des arbres de décision flous en terme de stabilité de la décision sera faite.

Ensuite, les arbres de décision flous étant basés sur l'utilisation de modalités floues, nous proposerons une méthode de construction automatique d'une partition floue à partir d'un ensemble de valeurs numériques ou de données imprécises.

Une des difficultés des arbres de décision est de prendre en compte plusieurs classes simultanément, nous proposerons alors une méthode, basée sur la construction d'une forêt d'arbres de décision flous, pour gérer ce type de difficultés.

1. En anglais : *data mining*

Finalement, les algorithmes de construction d'arbres de décision flous existants se différencient généralement par le choix de la mesure de sélection d'attributs, ou *mesure de discrimination*, utilisée durant le développement de l'arbre. Nous étudierons donc les éléments qui permettent de comparer les mesures de discrimination entre elles, et de construire de nouvelles mesures adaptées à la prise en compte de l'imprécision.

De la vision de Kampé de Fériet

Nos travaux présentés ici, sont fortement imprégnés des idées issues de la vision de Joseph Kampé de Fériet sur la théorie de l'information.

Les apports de Kampé de Fériet pour la théorie de l'information ont été considérables. Ses travaux ont commencé sur la théorie mathématique dans les années 1910 et se sont prolongés pendant près de 80 ans ! Des mathématiques théoriques à la théorie des mesures d'information en passant par la théorie cinématique, la thermodynamique, l'aéronautique..., sa trajectoire scientifique en fait *indéniablement* un chercheur d'exception.

Ce sont ses travaux dans le domaine de la théorie de l'information qui nous ont permis d'élucider la plupart des points délicats de nos propres travaux :

- l'étude des mesures de discrimination que nous proposons, qui est fondée sur la vision *a posteriori* que Kampé de Fériet a développée pour sa propre étude des mesures d'information.
- sa formulation des axiomes des mesures d'information qui, sans rejeter la théorie sur l'entropie existante, autorise de plus l'introduction d'un nouveau type de mesure d'information.
- sa définition de l'indépendance d'un ensemble d'observateurs qui nous a permis de postuler l'indépendance de chaque arbre dans une forêt d'arbres de décision flous.

Sur chacun de ces points, c'est en adoptant la vision de Kampé de Fériet que des solutions adaptées à notre cadre spécifique nous sont apparues. Et pourtant, une grande partie de ses travaux reste encore à exploiter et nous sommes convaincus que les années à venir apporteront encore la confirmation de ses intuitions extraordinaires.

Plan de la thèse

Cette thèse est composée en deux parties.

La première partie introduit l'apprentissage en présence de données imprécises.

Le premier chapitre présente l'apprentissage par induction en général, et la méthode d'apprentissage par arbre de décision en particulier. La théorie des sous-ensembles flous [2] pour la prise en compte de l'imprécision des données est introduite et son adéquation à ce type de problème est relevée.

2. **Remarque :** Dans cette thèse, les bases de la théorie des sous-ensembles flous et de la logique floue sont supposées connues du lecteur. Dans le cas contraire, nous lui conseillons de se reporter, par exemple, aux ouvrages (Bouchon-Meunier, 1994) et/ou (Bouchon-Meunier, 1995).

Dans le chapitre 2, nous détaillons la méthode d'apprentissage par induction, basée sur les arbres de décision, que nous avons choisi d'étudier et pour laquelle nous proposons une nouvelle formalisation. À partir de cette formalisation, les paramètres essentiels d'un algorithme de construction d'arbres de décision sont relevés. Ce sont sur ces paramètres que les apports des outils de la théorie des sous-ensembles flous peuvent intervenir pour la prise en compte de l'imprécision des données.

Nous présentons ensuite notre méthode d'utilisation d'arbres de décision flous pour classer de nouveaux exemples et nous étudions la stabilité de cette méthode. Finalement, nous présentons l'application *Salammbô* qui a été implémentée pour construire et utiliser des arbres de décision flous par les méthodes introduites.

La deuxième partie de cette thèse présente des améliorations que nous suggérons d'apporter pour améliorer la prise en compte de l'imprécision dans un tel processus d'apprentissage inductif.

Le chapitre 3 présente les différentes composantes du système général de construction d'arbres de décision flous que nous avons élaboré. Les trois chapitres suivants présentent nos apports sur chacune de ces composantes : discrétisation des attributs numériques, prise en compte de classes nombreuses, mesures de discrimination.

Dans le chapitre 4, une nouvelle méthode de construction automatique de partitions floues sur les univers des valeurs numériques ou numériques-symboliques des attributs est présentée. Cette méthode permet à un système d'apprentissage d'être entièrement autonome et de construire, par lui-même, la partition floue la mieux adaptée au problème d'apprentissage qu'il a à résoudre.

Dans le chapitre 5, les forêts d'arbres de décision flous sont présentées. Une forêt permet d'utiliser l'apprentissage inductif par arbre de décision flou dans un cadre où il existe plus de deux classes à apprendre. Nous proposons une méthode de construction de forêt d'arbres de décision flous ainsi qu'une méthode pour utiliser une telle forêt pour classer de nouveaux exemples. Cette méthode a donné lieu au programme informatique *Tanit* qui est décrit dans ce chapitre, et qui est basé sur l'application *Salammbô*.

Dans le chapitre 6, une hiérarchie de fonctions permettant de classer et de construire des mesures de discrimination est introduite. Cette hiérarchie valide les mesures de discrimination existantes et autorise la construction de nouvelles mesures de discrimination.

Le chapitre 7 rapporte les résultats obtenus par *Salammbô* sur différentes bases d'apprentissage habituellement utilisées pour comparer les méthodes d'apprentissage entre elles.

Finalement, nous présentons les conclusions que nous pouvons déduire de cette thèse et nous proposons diverses perspectives d'approfondissement de nos travaux, pour améliorer encore le traitement des valeurs imprécises en apprentissage par induction. De nouvelles voies d'études sont aussi relevées et les nouvelles questions à élucider sont proposées pour des études futures.

Première partie

Apprendre en présence de données imprécises

Chapitre 1

Apprentissage inductif et théorie des sous-ensembles flous

Il avait toujours été convaincu qu'en réfléchissant suffisamment, on pouvait venir à bout de tous les problèmes. Le vent, par exemple. Ce phénomène l'avait toujours intrigué, jusqu'à ce qu'il comprenne que le déplacement de l'air était provoqué par l'agitation des arbres.

T. Pratchett – *Le grand livre des gnomes*

1.1 Introduction

L'apprentissage est le meilleur moyen de retirer quelque chose de positif de ses échecs, c'est aussi le meilleur moyen de se souvenir de ses succès. C'est en les dotant de capacités d'apprentissage que les ordinateurs seront alors capables de devenir entièrement autonomes. Pour leur permettre de devenir entièrement autonomes, il faut leur apporter la possibilité d'apprendre par eux-mêmes dans tout type d'environnement, en présence de tout type de données, et, en particulier, en présence de données imprécises.

1.1.1 Apprendre

Il y a quelques années, les concepteurs de logiciel de jeu d'échecs pouvaient remarquer que les joueurs d'échecs humains avaient une façon souvent très draconienne de battre les programmes informatiques à ce jeu. Il leur suffisait de gagner une partie contre la machine et, ensuite, de réitérer la même partie chaque fois qu'une victoire leur était nécessaire contre la même machine. Il y a là un exemple d'apprentissage, l'apprentissage humain à qui il suffit souvent d'une expérience réussie pour arriver à en trouver une stratégie gagnante. Voilà aussi un exemple de ce qui se produit quand on n'est pas à même de profiter de ses erreurs pour corriger ses faiblesses, la stratégie informatique par excellence : répéter sans se souvenir.

Des concepteurs d'un logiciel ont alors eu l'idée de donner à l'ordinateur la faculté d'utiliser cette expérience de l'échec pour ne pas retomber deux fois dans le même piège.

Ils décidèrent de faire mémoriser par la machine les positions dont l'évaluation s'était subitement transformée après un coup de l'adversaire, cette évaluation devant permettre à l'ordinateur d'évaluer correctement la même position qui surviendrait dans des parties futures. Le résultat fut, en soit, très concluant, la stratégie humaine de gain à tout coup ne put plus s'appliquer, il fallut trouver d'autres solutions. L'inconvénient résida dans la mémorisation très coûteuse de ces évaluations. Quoiqu'il en soit, se souvenir de ses erreurs pour éviter de les reproduire est souvent le meilleur moyen de progresser. C'est plus élégamment exprimé par la définition de l'apprentissage que donne Herbert Simon : "*Learning denotes changes in the system that are adaptative in the sense that they enable the system to do the same task or tasks drawn from the same population more efficiently and more effectively the next time*" (Simon, 1984)

L'apprentissage dont il est question dans cette thèse est l'apprentissage dit *inductif*, c'est-à-dire l'apprentissage qui se fonde sur l'exploitation de l'observation d'événements passés et présents pour en tirer une loi générale qui sera utilisée par la suite.

1.1.2 Manipuler des connaissances imprécises

La théorie des sous-ensemble flous, introduite par (Zadeh, 1965), est un bon moyen de formaliser les processus de raisonnement humain. Sans revenir sur tous les détails de cette théorie, rappelons que son concept de base est de graduer l'appartenance à un ensemble. C'est un moyen efficace de prendre en compte l'imprécision dans des connaissances.

À la différence de l'incertitude inhérente à une connaissance qui donne le degré avec lequel on peut être certain qu'un événement se produira, l'imprécision d'une connaissance est liée à la mesure qui en est faite. Ainsi, la théorie des sous-ensembles flous ne préjuge pas de l'occurrence d'événements mais plutôt de la manière dont des éléments sont perçus ou connus. Manipuler les imprécisions est une capacité humaine par excellence. Le mode de raisonnement humain est capable d'appréhender des données imprécises et d'arriver à en déduire des conclusions adéquates.

La théorie des sous-ensembles flous formalise et prend en compte cette capacité, elle permet alors de modéliser pour les systèmes informatiques des modes de raisonnement proches des modèles humains.

Dans ce chapitre, après un rapide aperçu de l'apprentissage inductif, nous présenterons l'intérêt d'introduire la théorie des sous-ensembles flous dans un tel processus.

1.2 Apprendre par induction

L'apprentissage par induction élève le particulier vers le général. Ainsi, c'est par induction que les théories des sciences physiques ont été formulées au cours des âges. L'exemple caractéristique est l'induction de Newton qui trouve la théorie de l'attraction des corps en observant la chute d'une pomme. Tout au long de l'histoire d'une science expérimentale comme la physique, les théories se sont succédées et ont été démontrées et infirmées au rythme des observations qu'il était possible de faire. C'est le grand principe de démonstration et de découverte basé sur une méthode inductive de raisonnement. À partir de faits ou d'expériences, des propositions théoriques sont inférées par induction.

Certes, il n'est pas sans inconvénient d'établir une théorie de telle façon. La théorie des sciences physiques en est encore le plus bel exemple. Tout au long de son histoire, elle a connu de perpétuels changements au gré des découvertes de nouveaux phénomènes. On peut citer plus particulièrement la théorie électrodynamique des corps en mouvements qui s'établira progressivement jusqu'au début de notre siècle. Au gré des expérimentations, cette théorie était augmentée ou légèrement modifiée pour tenir compte de nouvelles observations.

Mais, un tel procédé de construction de théorie reste encore entièrement valide et reste un moyen important d'intégrer des capacités d'apprentissage à un ordinateur.

Dans notre cadre, nous considérons une *classe* C qui représente un phénomène à apprendre. Ce phénomène est décrit par un ensemble d'attributs $\mathcal{A}$, chaque attribut A_j pouvant prendre une valeur v_{jl} parmi un ensemble $\{v_{j1}, ..., v_{jm_j}\}$ de valeurs possibles.

Une *description* est un N-uplet de couples attribut - valeur (A_j, v_{jl}). Chaque description est associée à une valeur particulière c_k de la classe C pour constituer un *exemple* e_i du phénomène.

L'apprentissage inductif est la généralisation à partir d'un ensemble $\mathcal{E} = \{e_1, ..., e_n\}$ d'exemples, d'une loi générale régissant l'occurrence des valeurs de la classe C. Cet ensemble $\mathcal{E}$ porte le nom de *base d'apprentissage*.

La connaissance de la valeur de la classe pour chaque description de la base d'apprentissage nous place dans le cadre de l'apprentissage supervisé, c'est-à-dire dans un cas où des valeurs de la classes ont été associées, par un *professeur* ou un *expert*, à chaque description.

Position	*Pièce*	*Nb Pions*	*Temps restant*	*Résultat*
p_1	Dame	1	25s	**Gagné**
p_2	Fou	2	80s	**Nulle**
p_3	Tour	2	65s	**Gagné**
p_4	Tour	2	55s	**Perdu**
p_5	Fou	5	80	**Gagné**

Table 1.1 – *Exemple de base d'apprentissage*

Par exemple, dans la Table 1.1, on se place dans le domaine du jeu d'échecs où l'on souhaite apprendre à reconnaître le résultat qui découlera d'une position donnée. Le phénomène à apprendre est donc le résultat de la position donnée pour un des deux joueurs. La classe est ce résultat et elle peut prendre trois valeurs symboliques : gagnante, perdante ou nulle. Une position p_i est ici simplement décrite par trois attributs et est associée à une valeur de la classe. On considère la vision du côté de l'un des deux joueurs, ce sont seulement les éléments de la position qui le concernent qui sont utilisés. Les attributs sont :

- la pièce principale que le joueur possède (on suppose qu'il n'en reste qu'une seule sur l'échiquier),

- le nombre de pions qui restent au joueur,

– le temps restant qui reste à la pendule du joueur (on se place dans le cadre d'une partie chronométrée).

À chaque position, une valeur de la classe est associée. La base d'apprentissage est donc le résultat d'une observation du jeu de deux joueurs : à un instant donné de la partie une "photographie" a été prise de la position, puis on a attendu la terminaison de la partie, et son résultat a été associé à la photographie. Une base d'apprentissage est un ensemble de photographies classées.

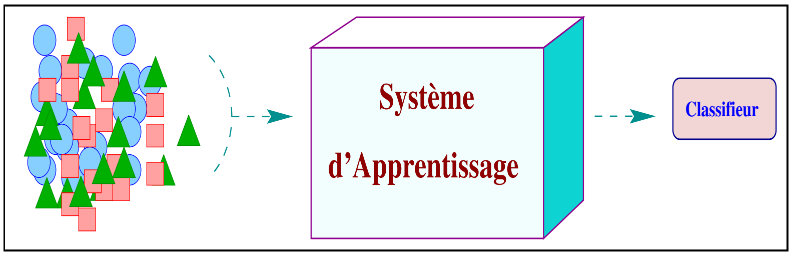


Figure 1.1 – *Schéma général de l'apprentissage inductif*

Le but de l'apprentissage inductif est de construire une fonction qui permette d'associer une valeur de la classe à toute nouvelle description. Cela revient à construire un *classifieur* (Figure 1.1) qui prenne pour entrée une description et donne en sortie la valeur de la classe associée à cette description (Figure 1.2).

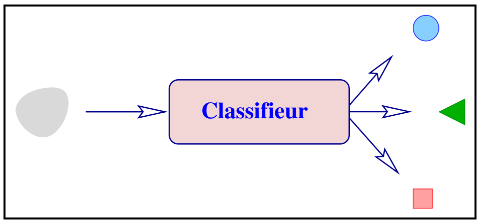


Figure 1.2 – *Classification*

Il existe de nombreux travaux sur la théorie de l'apprentissage en général et sur l'apprentissage inductif en particulier (entre autres (Carbonell et al., 1984), (Michalski, 1983; Michalski, 1986), (Ganascia, 1987), (Shavlik et Dietterich, 1990), (Kodratoff et Diday, 1991)).

Dans cette thèse, nous nous concentrons essentiellement sur une méthode d'apprentissage inductif particulière : la méthode de construction d'*arbres de décision*. C'est une méthode qui a l'avantage d'être simple à mettre en œuvre et de produire une structure facilement compréhensible. Le processus d'association d'une classe (dans le cas des arbres de décision on dit aussi la *décision*) à une description est aisément explicable. Les arbres de décision sont décrits en détail dans le chapitre 2 de cette thèse.

Construire une base de règles

Cette méthode d'apprentissage par induction permet de synthétiser un ensemble de connaissances sur un phénomène donné. La traduction d'une base d'apprentissage en un arbre de décision permet de ne garder que l'information pertinente pour la détermination de la classe, présente dans cette base. Toute redondance est supprimée et seuls les attributs fondamentaux pour déterminer la classe sont conservés. Le critère permettant de déterminer le sous-ensemble des attributs fondamentaux, la *mesure de discrimination*, caractérise l'algorithme de construction d'arbres de décision et sera examiné en détails dans le chapitre 6 de cette thèse.

L'apprentissage inductif par arbre de décision permet ainsi de construire une base de règles de production *si – alors* avec laquelle toute nouvelle description peut être classée par l'usage de la méthode d'inférence classique du raisonnement par déduction.

1.3 Apprentissage et sous-ensembles flous

Les méthodes d'apprentissage par induction connues et utilisées dans le domaine de l'Intelligence artificielle, sont souvent des méthodes performantes quand les attributs des descriptions sont des attributs *symboliques*, c'est-à-dire quand ils prennent leurs valeurs dans un ensemble discret et de cardinalité faible.

Par contre, ces méthodes rencontrent souvent des difficultés à traiter les attributs *numériques* ou *numériques-symboliques*. Un attribut numérique prend ses valeurs dans un ensemble continu. Un attribut numérique-symbolique est un attribut dont les valeurs peuvent être numériques ou symboliques. Dans ce cas, les valeurs symboliques sont alors attachées à une sémantique numérique. C'est cet aspect numérique-symbolique qui engendre une imprécision dans les données. Une modalité numérique-symbolique est souvent, par essence, imprécise.

Par exemple, dans le cadre de la base d'apprentissage donnée Table 1.1, l'attribut *Pièce* est un attribut symbolique, les attributs *Nb Pions* et *Temps restant* sont des attributs numériques. Cependant, le nombre de pions ou le temps restant peuvent être exprimés par une modalité symbolique : *peu, beaucoup, quelques...*, les attributs *Nb Pions* et *Temps restant* sont donc des attributs numériques-symboliques.

La théorie des sous-ensembles flous apporte des outils pour la prise en considération de telles données numériques-symboliques. Ces outils permettent ainsi d'introduire dans l'algorithme d'apprentissage inductif une meilleure gestion de l'imprécision, une robustesse accrue et une prise en compte du manque de connaissances.

Gérer l'imprécision

La théorie des sous-ensembles flous est un excellent moyen pour améliorer la prise en considération de ces types d'attributs particuliers. L'imprécision d'une donnée numérique-symbolique ne doit pas pour autant l'exclure du processus d'apprentissage. Bien au contraire, on peut souvent considérer qu'une valeur numérique-symbolique est hautement plus généralisable qu'une valeur précise. Cette valeur numérique-symbolique représente déjà une généralisation de valeurs strictement numériques. C'est donc une donnée privilégiée à exploiter et à rentabiliser dans un algorithme d'apprentissage inductif.

Par exemple, si l'on sait qu'avec *peu de pions* une partie est perdue, cela équivaut à savoir que cette partie est perdue avec 0, 1 ou 2 pions. Une valeur numérique-symbolique généralise alors à elle seule trois valeurs numériques précises.

Accroître la robustesse

La robustesse des systèmes flous est encore un moyen intéressant pour gérer la présence d'exemples contradictoires dans la base d'apprentissage (Kacprzyk et Iwanski, 1990; Kacprzyk et Iwański, 1991; Kacprzyk et Iwański, 1992). Deux exemples peuvent être considérés comme contradictoires s'ils possèdent exactement la même description mais qu'ils ne sont pas associés à la même classe.

Tenir compte du manque de connaissance

Finalement, la théorie du flou est un moyen de tenir compte d'un certain manque de connaissance qui peut apparaître lors de la constitution d'une base d'apprentissage.

La prise en compte d'un manque de connaissance peut être basée, par exemple, sur le modèle de la théorie des *rough sets* qui est une théorie dérivée de la théorie de sous-ensembles flous. L'appartenance des exemples aux classes n'est connue que par l'intermédiaire de deux approximations : un premier degré qui rend compte de la certitude avec laquelle un exemple appartient à une classe, et un deuxième pour la possibilité qu'il appartienne à cette classe (Pawlak, 1996), (Munakata et Pawlak, 1996).

Un autre manque de connaissance provient du fait que la valeur de certains attributs pour quelques exemples, peut ne pas être observable. Pourtant ces exemples sont intéressants pour enrichir la base d'apprentissage par la connaissance que donnent les valeurs de leurs autres attributs.

1.4 Conclusion

Dans ce chapitre, une vue d'ensemble de l'apprentissage par induction et une présentation de la théorie des sous-ensembles flous ont été données.

L'apprentissage inductif est le moyen pour généraliser des connaissances à partir d'un ensemble de cas mesurés d'un phénomène. La méthode d'apprentissage inductif que nous considérons est l'apprentissage par construction d'arbres de décision. Cette méthode est simple à mettre en œuvre et permet de construire une base de règles pour classifier de futures descriptions inconnues du phénomène.

Mais les techniques actuelles de construction d'arbres de décision, performantes en présence d'attributs symboliques, rencontrent des difficultés pour traiter les attributs numériques et numériques-symboliques, c'est-à-dire aussi l'imprécision des données observées, et pour traiter aussi les problèmes d'incohérence dans la base.

Ces difficultés sont naturellement gérées par la théorie des sous-ensembles flous. C'est pourquoi, il nous paraît intéressant d'étudier plus en détail tous les apports possibles de cette théorie pour améliorer le processus d'apprentissage par induction.

Dans le deuxième chapitre de cette partie, l'apprentissage inductif par arbres de décision est décrit en détail, de la phase de construction de l'arbre, l'apprentissage proprement dit, à la phase de classification avec un arbre ainsi construit. La théorie des sous-ensembles flous est introduite pour prendre en considération les données numériques-symboliques et imprécises dans ce processus d'apprentissage et permettre ainsi la construction d'un outil plus puissant que sont les *arbres de décision flous*.

Chapitre 2

Les arbres de décision flous

C'était à Mégara, faubourg de Carthage,
dans les jardins d'Hamilcar.

G. Flaubert – *Salammbô*

2.1 Introduction

La méthode d'induction que nous avons choisi d'utiliser repose sur la construction d'un arbre de décision. Un arbre de décision a l'avantage d'être simple à construire et à utiliser. De plus, c'est un outil explicite et aisément explicable. La structure arborescente d'un tel arbre est équivalente à une base de règles `si - alors`, ce qui rend justifiable une décision prise en suivant un de ses chemins. Cette structure de connaissances est, en outre, très proche des structures de connaissances manipulées naturellement par l'esprit humain.

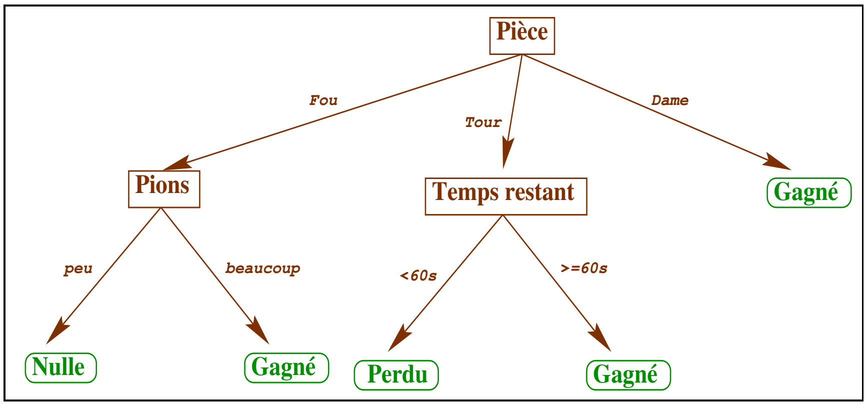


Figure 2.1 – *Exemple d'arbre de décision*

Un arbre de décision (Figure 2.1) est constitué de trois types d'éléments : les *nœuds*, les *arcs* et les *feuilles*. Chaque nœud est associé à un attribut et chaque arc issu de ce nœud est associé à l'une des caractérisations ou valeurs de cet attribut. Les feuilles, qui sont des nœuds sans arc sortant, donnent des modalités de la classe associées à la branche suivie pour l'atteindre.

Les arbres les plus simples correspondent au cas des attributs symboliques, possédant un nombre fini de valeurs (ou *modalités*). Mais d'autres types d'attributs peuvent apparaître, comme les attributs numériques qui prennent leurs valeurs dans un univers continu, ou les attributs numériques-symboliques qui peuvent prendre des valeurs à la fois numériques et symboliques. Par exemple, dans l'arbre donné dans la Figure 2.1, on se place dans le domaine du jeu d'échecs et l'on souhaite estimer le résultat d'une partie pour l'un des deux joueur, à partir de certains éléments de la position sur l'échiquier et de la consultation du temps qui lui reste pour réfléchir. L'attribut *Pièce* est un attribut symbolique qui prend ses valeurs dans l'ensemble des pièces du jeu d'échecs {Fou, Tour, Dame}. L'attribut *Temps restant* est un attribut numérique. L'attribut *Pions* est un attribut numérique-symbolique, ses valeurs sont numériques ($\{1, 2, ...\}$) mais l'interprétation qui en est faite est symbolique (*peu, beaucoup*).

Depuis de nombreuses années, il existe de nombreuses méthodes pour construire des arbres de décision avec des attributs symboliques ou numériques, mais les méthodes de construction d'arbres en présence d'attributs numériques-symboliques sont beaucoup plus récentes et beaucoup moins étudiées.

Ce chapitre présente les méthodes existantes de construction et d'utilisation d'arbres de décision flous ainsi que notre apport personnel dans ce domaine pour améliorer la prise en compte des données numériques-symboliques. Après un bref historique sur le domaine des arbres de décision et, en particulier, un rappel sur la théorie des questionnaires, nous présentons une nouvelle formalisation de la méthode générique de construction d'arbres de décision qui permet de faire ressortir les différents paramètres qui déterminent une méthode de construction. Cette formalisation nous permet ensuite de passer à la construction d'arbres de décision en présence de données imprécises en relevant les paramètres qui doivent prendre en compte l'imprécision des données. Les méthodes de construction d'arbres de décision flous sont alors détaillées pour faire ressortir leur similarité méthodologique.

Dans la seconde partie de ce chapitre, l'utilisation d'un arbre de décision pour le classement de nouveaux objets est présentée ; en particulier, l'analogie entre un arbre de décision et une base de règles est explicitée. Ensuite, nous développons plus en détails la méthode de classification avec un arbre de décision flou que nous proposons. Cette méthode est basée sur une mesure de la similarité entre les caractéristiques de l'objet à classer et les modalités présentes dans les nœuds de l'arbre.

Finalement, nous présentons l'implémentation de nos méthodes de construction et de classification avec un arbre de décision flou.

2.1.1 Un peu d'histoire

Les arbres de décision sont des descendants directs des *questionnaires*. Sans rentrer dans les détails trop formels, un questionnaire est un graphe quasi-fortement connexe inférieurement et sans cycle, où chaque nœud correspond à une question et chaque arc issu d'un nœud correspond à une réponse à la question (Picard, 1972). Les nœuds sans arc sortant, que nous appellerons des feuilles dans un arbre, sont des décisions à prendre en fonction du chemin choisi pour y arriver. Un questionnaire n'est pas forcément arborescent, il peut être latticiel, plusieurs arcs ayant la même extrémité terminale.

Les questionnaires ont été très étudiés durant les années 1960-70 (Picard, 1963; Picard, 1965; Picard, 1972; Picard, 1980), (Terrenoire, 1970). Au confluent de la théorie des graphes et de la théorie de l'information, la théorie des questionnaires est très riche et de nombreuses études y ont été menées (*cf.* par exemple, les actes de la conférence (Tours, 1983) pour un rapide aperçu de ce domaine). On peut citer, par exemple, les travaux de Lorigny à l'INSEE (Lorigny, 1980) qui sont très proches de ce que feront quelques années plus tard les *théoriciens* des arbres de décision.

Toujours dans les années 1960, les travaux de (Hunt et al., 1966) utilisent eux aussi le concept d'arbres de décision pour construire une structure de connaissance généralisant un ensemble de données.

Les travaux sur les questionnaires sont aussi la base des méthodes de construction de graphe d'induction comme la méthode SIPINA de (Zighed, 1985; Zighed et al., 1991).

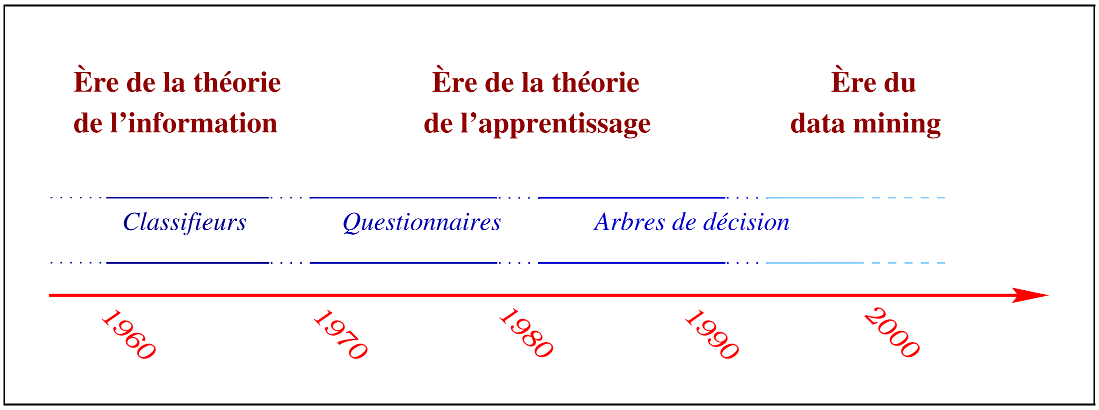


Figure 2.2 – *Terminologie à travers le temps du domaine des arbres de décision*

Dans le domaine statistique, les arbres de décision ont aussi été beaucoup étudiés. L'ouvrage sans doute le plus connu dans ce domaine est le livre présentant le système CART de construction d'arbres de régression (Breiman et al., 1984).

D'autre part, les travaux de (Christensen, 1980) de la fin des années 60 aux années 80 sont aussi très liés aux arbres de décision. Christensen utilise la mesure d'entropie de Shannon (Shannon, 1948) pour construire et améliorer les arbres de décision dans le domaine de la reconnaissance de formes. C'est un auteur très prolifique sur ce domaine.

Les méthodes de construction de questionnaires classiques étant très naturellement automatisables, il est apparu que, par ce type de méthodes, il était très aisé de construire un système de décision. Un tel système peut alors être utilisé pour déterminer la décision attachée à n'importe quel nouveau cas qui sera rencontré par la suite. Tous ces travaux de la théorie de l'information, surgissent au milieu des années 1980 dans le domaine de la théorie de l'apprentissage informatique. Les arbres de décision deviennent un des outils d'apprentissage inductif les plus populaires à partir de l'article de (Quinlan, 1984). L'*algorithme ID3*, qui est l'appellation donnée dans cet article à la méthode de construction d'arbres de décision, devint même alors l'algorithme éponyme des algorithmes de construction d'arbre de décision.

À l'heure actuelle, les arbres de décision poursuivent leur histoire dans les nouvelles disciplines informatiques liées à la découverte de connaissances, en particulier, dans la partie de la *fouille de données* (*data mining*) de ce domaine (Fayyad et al., 1996). On peut citer, entre autres, l'américain Fayyad, un des pionniers du domaine de la découverte de connaissances, qui a appliqué, avec succès, les arbres de décision dans le domaine de l'astronomie (Fayyad et Irani, 1992a; Fayyad et Irani, 1992b; Fayyad et Irani, 1993; Fayyad et al., 1993).

De plus en plus, au fil des temps, le besoin de traiter d'autres types d'attributs que des attributs symboliques, le besoin d'améliorer la prise en compte des attributs numériques, et la nécessité de gérer des données numériques-symboliques ont demandé l'apport de nouvelles techniques. La théorie des questionnaires a été la première à envisager l'utilisation des éléments de la théorie des sous-ensembles flous pour la prise en compte de données numériques ou numériques-symboliques.

2.1.2 Vers les questionnaires flous

Il apparut très vite la nécessité de traiter, d'abord les attributs numériques, puis les attributs numériques-symboliques, pour construire les questionnaires ou les arbres de décision.

Dans le cas des attributs numériques, leurs univers de valeurs ont, tout d'abord, été tout simplement discrétisés pour engendrer des seuils. Ces seuils déterminent des tests sur les valeurs dans les nœuds du questionnaire ou de l'arbre.

Les premiers travaux où apparaissent la théorie des sous-ensembles flous dans les arbres de décision sont ceux de (Chang et Pavlidis, 1977) et de (Adamo, 1980). Ces premiers travaux considèrent plutôt une utilisation floue d'arbres de décision classiques, notamment au niveau du respect des tests à chaque nœud de l'arbre.

Le concept de questionnaire flou apparaît un peu plus tard avec les travaux de (Bouchon, 1981; Bouchon, 1982b). Ici, les questionnaires peuvent être considérés comme véritablement flous : les étiquettes libellant les tests sont des valeurs floues.

Avec les arbres de décision post-ID3, le traitement des valeurs numériques a donné lieu à de nombreux travaux.

2.1.3 Les arbres de décision et les données numériques

Dans la première version de l'algorithme ID3, les attributs à valeurs numériques étaient conservées tels quels, les valeurs numériques étaient considérées comme les valeurs d'un attribut symbolique. Des méthodes plus sophistiquées existaient pourtant et étaient déjà connues dans la théorie des questionnaires.

La méthode traditionnelle de traitement des valeurs numériques utilise un seuil de discrétisation comme valeur test de comparaison. Généralement, comme dans la théorie des questionnaires, un seuil est donné pour chaque attribut numérique, mais certaines méthodes, comme celle de (Breiman et al., 1984), existent pour construire ce seuil pendant le processus de construction de l'arbre.

De même, les méthodes usuelles de prise en compte des attributs numériques discrétisent leur univers de valeurs en calculant des seuils qui serviront dans les tests des nœuds de l'arbre de décision (Catlett, 1991), (Wehenkel, 1997). Les techniques issues de la théorie des sous-ensembles flous sont alors un bon outil pour prendre en compte l'imprécision de tels seuils calculés lors de la phase de discrétisation : elles permettent d'assouplir les frontières créées par ces seuils. Des arbres de décision flous sont alors construits avec ces seuils qui seront considérés comme des valeurs floues lors de l'utilisation de l'arbre en généralisation (Bouchon-Meunier et al., 1995).

D'autres méthodes de construction d'arbres de décision prenant en compte les données numériques ont été proposées. Une des méthodes les plus naturelles est celle de (Maher et Saint-Clair, 1993). Ces auteurs considèrent les valeurs numériques comme des valeurs symboliques durant la construction de l'arbre de décision. Pendant la généralisation avec l'arbre ainsi construit, ils utilisent des degrés de possibilité pour la comparaison des valeurs de l'exemple à classer aux valeurs de l'arbre. L'inconvénient majeur d'une telle approche est la taille des arbres de décision ainsi construits. Ils sont très larges et peu profonds, rendant compte du fait qu'en règle générale, une valeur numérique est très discriminante mais peu informative. Les arbres ainsi construits possèdent presque autant de branches que de valeurs présentes dans la base d'apprentissage, chaque branche se terminant en une étape sur une feuille.

Une autre méthode est de complexifier la structure des nœuds présents dans l'arbre, par exemple en utilisant un trio de neurones (D'Alché-Buc, 1993).

En considérant que les valeurs symboliques des attributs numériques/symboliques sont des modalités floues sur l'univers numérique des valeurs de cet attribut, des méthodes de traitement particulièrement adaptées, issues de la théorie des sous-ensembles flous, peuvent aider à la prise en compte de ce type d'attributs, telles les méthodes de construction d'arbres de décision flous (Ramdani, 1992), (Weber, 1992), (Janikow, 1994), (Yuan et Shaw, 1995), (Boyen, 1995), (Bothorel, 1996).

C'est ce type de méthode de traitement des données numériques ou numériques-symboliques qui nous paraît le plus prometteur. Le flou dans les données, ou leur précision, sont des critères inhérents à notre monde réel. Doter un système de capacités d'apprentissage inductif automatique passe donc par cette étape obligatoire de traitement de ce type de données.

Dans la section suivante, nous allons présenter l'algorithme générique de construction d'arbres de décision. Cet algorithme nous permet de formaliser le processus de construction d'arbre de décision et d'en faire ressortir les éléments principaux sur lesquels il faut axer nos efforts pour la prise en compte de données numériques-symboliques. Ainsi, il est possible de proposer la construction d'arbres de décision flous, c'est-à-dire d'arbres de décision dont les modalités des attributs présentes dans les tests des nœuds sont numériques-symboliques ou bien, dont les modalités des classes libellant les feuilles sont des valeurs numériques-symboliques.

2.2 Construction des arbres de décision

Les algorithmes de construction d'arbres de décision à partir d'une base d'apprentissage procèdent tous d'une manière équivalente. Ils construisent un arbre d'une façon descendante, de la racine (qui est le faîte d'un arbre informatique) vers les feuilles (qui constituent la base d'un arbre informatique), selon une méthode générale de construction de système arborescent de classification. Ce qui les différencie réside dans le choix des critères utilisés.

Dans la partie qui suit, nous proposons une formalisation des algorithmes de construction des arbres de décision. Cette formalisation permet de faire apparaître les paramètres d'un tel algorithme sur lesquels il est alors possible d'intégrer, à leur niveau, des techniques de traitement des données numériques-symboliques.

2.2.1 Formalisation du principe de construction des arbres de décision

Un algorithme de construction d'arbres de décision est dit de type *Top Down Induction of Decision Tree* (TDIDT) quand il procède par divisions successives de la base d'apprentissage. Chaque division est réalisée par une question sur la valeur d'un attribut.

Étant donné une mesure de discrimination H, une stratégie de partitionnement P et un critère d'arrêt T, une base d'apprentissage $\mathcal{E}$, composés d'exemples e_i définis par un ensemble d'attribut $\mathcal{A}$ et une classe C, l'algorithme générique de construction d'arbres de décision est le suivant :

1. À l'aide de la mesure de discrimination H, *choisir* un attribut $A_j \in \mathcal{A}$, de modalités[1] $\{v_{j1}, ..., v_{jm_j}\}$, pour partitionner la base d'apprentissage $\mathcal{E}$. Un nœud $\eta(A_j)$ est créé dans l'arbre, portant un test sur la valeur de A_j.

2. Suivant la stratégie P, *partitionner* la base d'apprentissage courante ($\mathcal{E}$ au départ de l'algorithme) avec cet attribut en créant autant de sous-bases $\mathcal{E}_l$ que l'attribut possède de modalités v_{jl} :

$$\mathcal{E} = \bigcup_{l=1,\ldots,m_j} \mathcal{E}_l \quad \text{avec} \quad \mathcal{E}_l = \{e_i \in \mathcal{E} \mid e_i(A_j) = v_{jl}\}$$

   Chaque valeur v_{jl} de l'attribut libelle un arc issu de ce nœud $\eta(A_j)$, associé à la sous-base $\mathcal{E}_l$.

3. À l'aide du critère d'arrêt T, *vérifier la condition d'arrêt* sur chaque sous-base $\mathcal{E}_l$ créée. Une sous-base qui vérifie la condition d'arrêt donne naissance à une feuille dans l'arbre.

4. Recommencer en 1 avec les sous-bases qui ne vérifient pas le critère d'arrêt.

Comme il a été précisé, la différence entre les algorithmes existants se situe dans les choix à opérer pour *choisir* un attribut à l'étape 1, pour *partitionner* la base à

1. Une modalité est une valeur symbolique, une valeur numérique-symbolique ou, éventuellement, un ensemble de valeurs de l'attribut regroupées. Ce regroupement peut être le résultat d'une discrétisation (voir chapitre 4), ou le résultat du regroupement de plusieurs valeurs symboliques.

l'étape 2 et pour mesurer le *critère d'arrêt* à l'étape 3. De plus, un algorithme de construction d'arbres de décision traite une base d'apprentissage $\mathcal{E}$, qui est un ensemble d'exemples décrits par un ensemble d'attributs. Dans la structure d'une telle base $\mathcal{E}$, rien ne distingue l'attribut de classe des autres attributs, c'est sa définition *sémantique* qui le distingue des autres attributs. La définition sémantique rappelle que cet attribut doit être expliqué à partir de tous les autres [2].

Certains paramètres conditionnent la structure et le fonctionnement de l'algorithme :

a) Le choix d'un attribut s'effectue toujours à l'aide d'une *mesure de discrimination* H, aussi nommée *mesure d'impureté*, qui rend compte du pouvoir discriminant d'un attribut relativement aux classes, c'est-à-dire de la façon dont cet attribut permet de déterminer la classe des éléments de la base d'apprentissage. Cette mesure permet de classer les attributs les uns par rapport aux autres et d'en choisir un qui soit optimal pour partitionner la base d'apprentissage. Un attribut complètement discriminant possède des valeurs qui sont entièrement liées aux valeurs de la classe, de telle sorte que, connaissant la valeur de l'attribut, la classe s'en déduit immédiatement. Choisir un attribut à l'aide d'une mesure adéquate est une heuristique qui permet d'optimiser la taille de l'arbre construit. La cohérence sémantique d'un nœud correspond au fait qu'un expert du domaine le reconnaîtra en tant que question pertinente en vue de déterminer la classe.

b) La stratégie de partitionnement P de la base d'apprentissage relativement à une question posée sur les valeurs d'un attribut s'effectue généralement en décomposant la base en sous-bases, chacune induite par une modalité de la liste des modalités correspondant à l'attribut. Dans un cadre classique, il n'y a pas d'alternative de partitionnement. Par contre, dans le cas des arbres flous, il est possible d'envisager plusieurs manières différentes de partitionner la base après une question sur des modalités floues. L'appartenance des exemples aux modalités floues est graduelle, la partition de $\mathcal{E}$ engendrée par les modalités floues est une partition floue. Une stratégie de partitionnement est alors de diviser $\mathcal{E}$ en utilisant la partition floue, certains exemples pouvant alors appartenir à plusieurs sous-bases avec certains degrés d'appartenance. Une autre stratégie est d'utiliser des α-coupes pour retrouver des partitions plus classiques de $\mathcal{E}$.

c) Le critère d'arrêt T est souvent lié à la mesure de discrimination utilisée pour chercher le meilleur attribut à l'étape 1. Si tous les exemples de l'ensemble courant possèdent la même classe, le processus de développement de l'arbre peut s'arrêter. Mais, il est aussi possible d'arrêter la construction en fonction d'autres facteurs que la discrimination relative aux classes, comme la taille de la base.

Cet examen du principe des algorithmes de construction des arbres de décision fait apparaître une formalisation possible de ces processus.

2. Dans la théorie des statistiques, les termes de variable *endogène* et de variable *exogène* sont utilisés respectivement pour la classe et les attributs descriptifs. L'appellation générique *variable* rappelle alors mieux la similitude des deux types d'attributs que l'appellation de la théorie de l'apprentissage qui utilise deux termes distincts attribut et classe.

Formalisation des algorithmes de construction d'arbres de décision

On considère une liste d'attributs $\mathcal{A}$, et

- S^H : l'ensemble des mesures de discrimination
- S^P : l'ensemble des stratégies de partitionnement
- S^T : l'ensemble des critères d'arrêt
- $S^{\mathcal{A}}$: l'ensemble des bases d'exemples définies par les attributs de $\mathcal{A}$
- $S^{\mathcal{D}}$: l'ensemble des arbres de décision reconnaissant la classe C, qui est un attribut particulier de $\mathcal{A}$, en utilisant les attributs de $\mathcal{A} - \{C\}$

Un algorithme de construction d'arbres de décision se formalise à l'aide d'une fonction Φ telle que :

$$\begin{aligned} \Phi : S^H \times S^P \times S^T \times \mathcal{A} \times S^{\mathcal{A}} &\longrightarrow \quad S^{\mathcal{D}} \\ (H, P, T, C, \mathcal{E}) &\longmapsto \quad \mathcal{D} = \Phi(H, P, T, C, \mathcal{E}) \end{aligned}$$

Les paramètres de cette fonction sont donc :

- une mesure de discrimination H,
- une stratégie de partitionnement P,
- un critère d'arrêt T,
- un ensemble $\mathcal{E}$, constitué par des exemples e_i. Chaque exemple est décrit à l'aide d'attributs d'une liste $\mathcal{A} = \{A_1, ..., A_N, C\}$.
- un attribut particulier C de $\mathcal{A}$, appelé *classe*.

Les paramètres H, P et T sont dits *structurels* car ils conditionnent la structure même de l'algorithme, sa manière de fonctionner. Les paramètres $\mathcal{E}$ et C sont, eux, des paramètres *du domaine* d'expertise pour lequel un arbre de décision doit être construit.

Une vue synthétique de l'architecture générale d'un système de construction d'arbres de décision est donnée dans le schéma 2.3. Ce système comporte cinq entrées. Trois de ces entrées H, P et T, correspondent aux paramètres structurels ; elles influent sur le comportement de l'algorithme et conditionnent les trois étapes du processus de construction : le choix du meilleur attribut, la méthode de partition et le test d'arrêt. Ce sont ces paramètres qui différencient les algorithmes de construction entre eux. Les deux autres entrées $\mathcal{E}$ et C, sont des paramètres liés au domaine pour lequel l'arbre est construit.

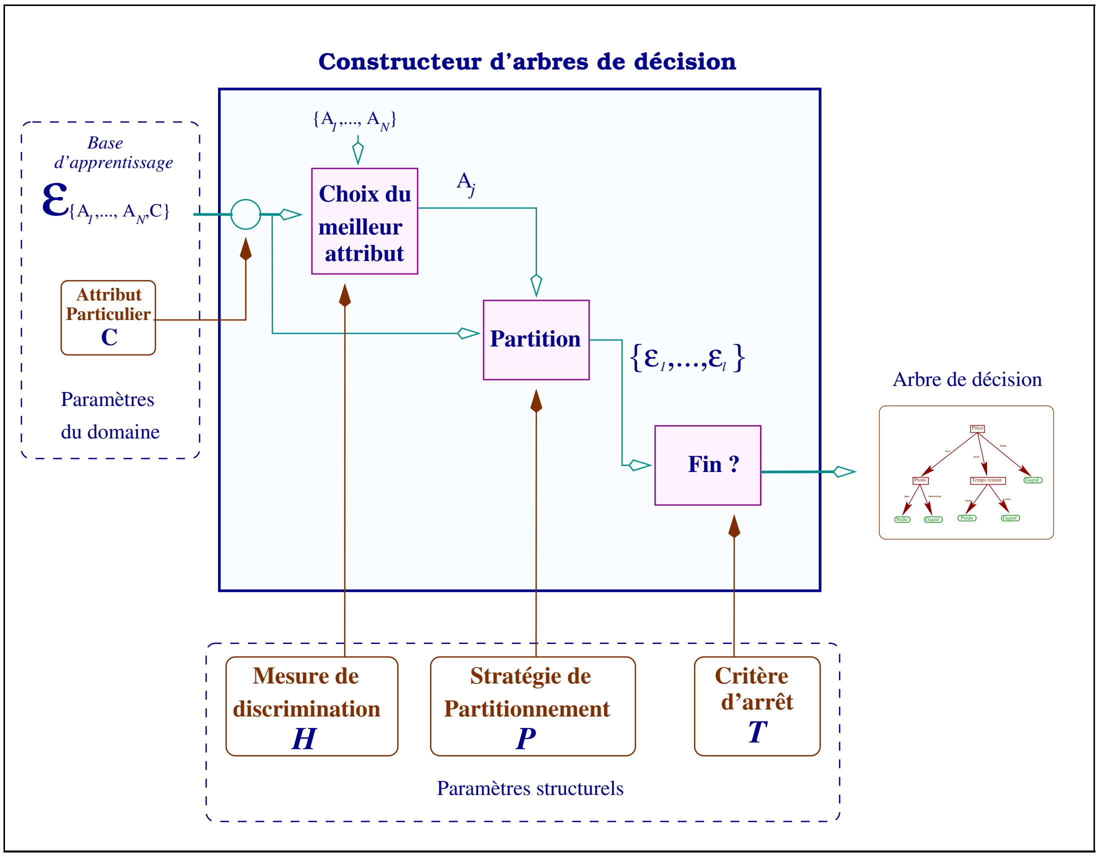


Figure 2.3 – *Architecture générale d'un système de construction d'arbres de décision*

Expression de l'algorithme ID3 à l'aide de ce formalisme

L'algorithme de construction d'arbres de décision ID3 de (Quinlan, 1984), décrit plus loin dans ce chapitre (section 2.2.5) s'exprime dans ce formalisme. Cet algorithme est adapté au traitement d'attributs symboliques ou numériques. Il utilise la mesure d'entropie de Shannon H_S comme mesure de discrimination.

La stratégie de partitionnement est le partitionnement classique induit par les valeurs v_{jl} de l'attribut A_j. Suivant les notations données précédemment dans l'algorithme générique, $\mathcal{E}$ est partitionné en m_j sous-bases induites par les valeurs de A_j. Cette stratégie de partitionnement est notée P_c (comme Partionnement Classique).

L'algorithme de construction arrête le développement d'un chemin quand tous les exemples de la base d'apprentissage $\mathcal{E}$ possèdent la même classe, c'est-à-dire quand $H_S(\mathcal{E}) = 0$. Cette stratégie d'arrêt est notée T_0.

Par conséquent, étant donné une base d'apprentissage $\mathcal{E}$ d'exemples décrits par un ensemble d'attributs $\{\mathcal{A}_1, ..., A_N, C\}$ et un attribut particulier C qui est la classe à reconnaître, l'algorithme ID3 est la fonction $\Phi(H_S, P_c, T_0, C, \mathcal{E})$ qui peut se noter plus simplement par :

$$\Phi_Q : \mathcal{A} \times S^{\mathcal{E}} \longrightarrow S^{\mathcal{D}}$$

$$\Phi_Q(C, \mathcal{E}) = \Phi(H_S, P_c, T_0, C, \mathcal{E})$$

La valeur des paramètres H, P et T étant fixée une fois le type d'algorithme choisi, l'algorithme ID3 peut alors être énoncé sous la forme de cette fonction de $\mathcal{A} \times S^{\mathcal{E}}$.

Formaliser pour étudier

Cette formalisation permet de se rendre compte des éléments d'étude de la théorie des arbres de décision. Elle permet aussi de se rendre compte à quel niveau l'apport des éléments de la théorie des sous-ensembles flous doit s'effectuer. Ces éléments sont spécifiques à chaque paramètre de la fonction Φ.

Dans ce qui suit, nous allons examiner un peu plus en détail ces différents paramètres de l'algorithme de construction en étudiant les choix faits dans les algorithmes existants. L'intérêt d'une telle étude est de pouvoir cataloguer ces choix afin de déterminer s'il en existe un meilleur que les autres, et de montrer l'intérêt des nouveaux critères que nous proposons. De plus, cette étude permettra de déterminer les différences entre les méthodes floues existantes et de mieux affiner les besoins, en termes de théorie de sous-ensembles flous, de tels algorithmes.

Comme il est possible de le remarquer dans l'algorithme de construction, il faut déterminer les deux mesures qui interviennent dans le processus. La première mesure à choisir, la mesure de discrimination, est celle qui permet de sélectionner les attributs entre eux en mesurant leur pouvoir discriminant vis-à-vis de la classe. La seconde mesure, relative au critère d'arrêt, est celle qui permet de savoir s'il est judicieux d'arrêter le processus de construction de l'arbre pour la branche en cours.

2.2.2 Mesure de discrimination

Une mesure de discrimination doit rendre compte du pouvoir discriminant d'un attribut vis-à-vis de la classe relativement à un ensemble d'exemples donné. Elle doit mesurer la façon dont cet attribut sépare cet ensemble en sous-ensembles dont les éléments possèdent tous la même classe. Mais afin de choisir une mesure de discrimination, encore il faut arriver à définir les propriétés que l'on doit en attendre.

Une *bonne* mesure est heuristiquement le meilleur moyen de limiter la taille de l'arbre et de donner une cohérence sémantique aux nœuds qui le composent. Un arbre est un *recodage* de la base d'apprentissage orienté vers la seule conservation de l'information utile inhérente à cette base. L'information est jugée utile quand elle suffit à déterminer la classe des éléments. Reformulée en ces termes, la justification de l'usage de la théorie de l'information apparaît clairement, puisque elle a été introduite dans la théorie du codage et, en particulier, la justification de l'usage de la mesure d'entropie de Shannon dans le domaine des arbres de décision.

Contrairement aux conclusions de Mingers, nous ne pensons pas que le pouvoir de prédiction d'un arbre de décision est indépendant du choix de la mesure de discrimination utilisée et qu'un choix aléatoire des attributs pour libeller les nœuds de l'arbre est aussi efficace qu'un choix fait à l'aide d'une mesure (Mingers, 1989b). Nous ne sommes pas les seuls à penser cela. Dans leur réponse à Mingers, Buntine et Niblett montrent que la mesure a un impact sur l'arbre construit et peut donc modifier le taux de classification (Buntine et Niblett, 1992). L'inconvénient des conclusions de Mingers repose dans le protocole expérimental qu'il met en œuvre pour comparer les mesures de sélection et sur les conclusions qu'il en retire. Ainsi, il précise que les arbres obtenus pour ces tests par les différentes méthodes ont été élagués pour obtenir une *mesure réaliste de la fiabilité des arbres*[3]. Même si la méthode utilisée pour élaguer (la méthode de Breiman) est identique pour chaque mesure testée, cela influence grandement l'efficacité de l'arbre. Dans leur article, Buntine et Niblett proposent, eux, un protocole expérimental qui permet, de façon plus réaliste, de vérifier l'importance d'une mesure adéquate pour sélectionner les attributs.

Les récents travaux de Lerman en analyse de données l'ont aussi amené à se pencher sur les choix d'une bonne mesure de discrimination (Lerman et da Costa, 1996). Dans (Lerman et da Costa, 1996), les auteurs recensent un grand nombre de mesures de discrimination utilisées et en recherchent une la mieux adaptée à un problème où le nombre de classe à reconnaître est très grand.

Dans ce qui suit, trois méthodes de validation de mesures de discrimination sont présentées. Chacunes de ces méthodes s'attache à la reconnaissance d'une bonne mesure de discrimination en fonction des propriétés qu'elle possède. La première méthode est très ancienne et révèle l'ancienneté de ce type de problème. La deuxième méthode est celle qui est la plus couramment utilisée. La dernière méthode présentée est celle que nous proposons d'introduire. Elle se situe dans une optique méthodologique différente de la précédente, car elle est initiée par les travaux de Joseph Kampé de Fériet. C'est une méthode *constructive* car elle permet, de par sa nature, de construire aisément une nouvelle mesure de discrimination.

Finalement, toute cette étude a aussi pour but de déterminer les critères déterminants qui permettent de distinguer une bonne mesure *floue* de discrimination, c'est-à-dire une mesure capable de déterminer le pouvoir discriminant d'un attribut numérique-symbolique, relativement à une classe floue.

À notre connaissance, si la question de la pertinence de la mesure est souvent étudiée dans le cadre symbolique (et numérique discrétisé, par extension), elle ne s'est pas encore posée dans le domaine du traitement des attributs numériques-symboliques. Pourtant, si des méthodes existent pour apprécier la pertinence d'une mesure pour rendre compte du pouvoir discriminant dans un cadre classique, il paraît aussi intéressant d'apprécier la pertinence d'une mesure dans un cadre flou. L'axiome de base qui sous-entend qu'il suffit d'étendre une mesure classique pour la prise en compte de données floues ne suffit pas puisqu'il existe diverses façons d'effectuer une telle extension, pour chaque mesure classique, et que, d'autre part, ces méthodes d'extension ne conservent pas obligatoirement les propriétés à exiger d'une mesure de discrimination.

Une première méthode de validation

Déjà dans le domaine de la sélection de caractéristiques dans les systèmes de reconnaissance statistique, (Lewis, 1962) définit les propriétés que doit posséder une mesure

3. "*In order to get a realistic measure of the accuracy of trees as they would be used in practice, it was necessary to prune them.*"(Mingers, 1989b), *p. 334*.

statistique qui rende compte de la *pertinence*[4] d'une caractéristique.

Pour être considérée comme pertinente, il demande à une mesure I de posséder les trois propriétés suivantes :

1. si $I(C|A_{j_1}) > I(C|A_{j_2})$ alors un système de reconnaissance qui n'utilise que l'attribut A_{j_1} possède un taux de reconnaissance plus élevé qu'un système qui n'utilise que l'attribut A_{j_2}.

2. si $I(C|A_{j_1}) > I(C|A_{j_2})$ alors un système de reconnaissance qui utilise l'attribut A_{j_1} et un ensemble d'autres attributs $\mathcal{A}$ possède un taux de reconnaissance plus élevé qu'un système qui utilise l'attribut A_{j_2} et $\mathcal{A}$.

3. étant donné un ensemble d'attributs $\mathcal{A}$, le taux de bonnes reconnaissances doit être une fonction linéaire de la somme $\sum_{A_j \in \mathcal{A}} I(C|A_j)$.

Si la dernière propriété paraît surprenante, il faut se rappeler que Lewis se place dans un cadre statistique. Pour lui, le taux de bonnes reconnaissances est la probabilité théorique de reconnaître la classe connaissant un ensemble de valeurs pour les attributs.

Après une étude plus approfondie de ces propriétés, il remarque qu'elles sont toutes les trois incompatibles car il n'existe pas de fonctions qui les satisfassent en même temps quelles que soient les conditions. Par contre, il existe des mesures qui satisfont ces propriétés dans un grand nombre de cas. En tout état de cause, la mesure qu'il se propose alors d'utiliser pour trouver les attributs pertinents est :

$$I_1(C|A_j) = \sum_{k=1}^{K} \sum_{l=1}^{m_j} P(c_k, v_{jl}) \log(\frac{P(v_{jl}|c_k)}{P(v_{jl})}) \tag{2.1}$$

Bien que Lewis ne fasse aucune référence à l'entropie classique de Shannon (Shannon, 1948), on montre aisément qu'elle apparaît à travers cette mesure. On a :

$$I_1(C|A_j) = \sum_{k=1}^{K} \sum_{l=1}^{m_j} P(c_k, v_{jl}) \log(\frac{P(c_k|v_{jl})}{P(c_k)})$$

Étant donné que, par définition :

$$\begin{aligned} P(v_{jl}|c_k) &= \frac{P(c_k, v_{jl})}{P(c_k)} \quad \text{et} \\ P(c_k|v_{jl}) &= \frac{P(c_k, v_{jl})}{P(v_{jl})} \end{aligned}$$

les probabilités conditionnelles vérifient donc :

$$\frac{P(v_{jl}|c_k)}{P(v_{jl})} = \frac{P(c_k|v_{jl})}{P(c_k)}$$

4. Lewis emploie le terme de *goodness*.

Par conséquent :

$$
\begin{aligned}
I_1(C|A_j) &= \sum_{k=1}^{K}\sum_{l=1}^{m_j} P(c_k, v_{jl}) \log(P(c_k|v_{jl}) - \sum_{k=1}^{K}\sum_{l=1}^{m_j} P(c_k, v_{jl}) \log(P(c_k)) \\
&= \sum_{k=1}^{K}\sum_{l=1}^{m_j} P(v_{jl})P(c_k|v_{jl}) \log(P(c_k|v_{jl}) - \sum_{k=1}^{K}\sum_{l=1}^{m_j} P(v_{jl}|c_k)P(c_k) \log(P(c_k)) \\
&= \sum_{l=1}^{m_j} P(v_{jl}) \sum_{k=1}^{K} P(c_k|v_{jl}) \log(P(c_k|v_{jl}) - \sum_{k=1}^{K} P(c_k) \log(P(c_k)) \sum_{l=1}^{m_j} P(v_{jl}|c_k)
\end{aligned}
$$

Et comme $\sum_{l=1}^{m_j} P(v_{jl}|c_k) = 1$ car les v_{jl} sont considérés comme constituant un système complet d'événements. Il vient :

$$
I_1(C|A_j) = \sum_{l=1}^{m_j} P(v_{jl}) \sum_{k=1}^{K} P(c_k|v_{jl}) \log(P(c_k|v_{jl}) - \sum_{k=1}^{K} P(c_k) \log(P(c_k))
$$

Soit

$$
I_1(C|A_j) = H_S(C) - H_S(C|A_j)
$$

en notant :

$$
H_S(C) = -\sum_{k=1}^{K} P(c_k) \log(P(c_k)) \quad \text{et} \tag{2.2}
$$

$$
H_S(C|A_j) = -\sum_{l=1}^{m_j} P(v_{jl}) \sum_{k=1}^{K} P(c_k|v_{jl}) \log(P(c_k|v_{jl}) \tag{2.3}
$$

$H_S(C|A_j)$ est la formule bien connue de l'entropie conditionnelle de Shannon, $H_S(C)$ est l'entropie de Shannon d'un ensemble relativement à la classe. Ainsi, $I_1(C|A_j)$ mesure le *gain d'information* apportée par une partition sur les valeurs de l'attribut A_j, relativement à la classe.

Il est intéressant de noter que Lewis complète la démonstration de l'adéquation de I_1 à ses propriétés en précisant : “*Another justification is that the information theory terms* $[I_1(C|A_j)]$ *is the average information about the* $[c_k]$ *given by* $[A_j]$”

La méthode de validation usuelle

Dans leur livre, (Breiman et al., 1984) proposent une étude des propriétés requises pour mesurer le pouvoir discriminant d'un attribut relativement à la classe. Dans le cas où il n'y a que deux valeurs de la classe c_1 et c_2 à reconnaître, ces propriétés sont :

1. *Minimalité* quand l'incertitude est nulle. $H(\mathcal{E}) = 0$ si tous les éléments de $\mathcal{E}$ ont la même classe.

2. *Maximalité* quand l'incertitude est totale. $H(\mathcal{E}) = 1$ si la moitié des exemples de $\mathcal{E}$ possède la classe c_1 et l'autre moitié possède la classe c_2.

3. *Symétrie.* Si $\mathcal{E}_1$ possède n_1 exemples de classe c_1 et n_2 exemples de classe c_2 et si $\mathcal{E}_2$ possède n_2 exemples de classe c_1 et n_1 exemples de classe c_2, alors leurs mesures ont la même valeur : $H(\mathcal{E}_1) = H(\mathcal{E}_2)$

Dans le cas où il existe plus de deux classes, ces trois propriétés sont reprises et une nouvelle propriété est ajoutée. Par exemple, à l'instar de (Zighed et al., 1991) qui recense un ensemble de six propriétés souhaitables pour une mesure d'incertitude, (Rabaséda-Loudcher, 1996) propose quatre propriétés fondamentales pour une mesure de discrimination. Dans ce qui suit, soit une partition $\mathcal{P}$ d'un ensemble $\mathcal{E}$ en m sous-ensembles : $\mathcal{P} = \{\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_m\}$. On note $H(\mathcal{P})$ l'information de cette partition.

1. *Minimalité* quand l'incertitude est nulle, c'est-à-dire, si, pour tout $j = 1, ..., m$, tous les $e_i \in \mathcal{E}_j$ possèdent la même classe, alors $H(\mathcal{P}) = 0$.

2. *Maximalité* quand l'incertitude est totale, c'est-à-dire, si, pour tout $j = 1, ..., m$, il existe autant d'exemples de chaque classe dans chaque $\mathcal{E}_j$, alors $H(\mathcal{P})$ est maximale.

3. *Symétrie*: pour toute permutation σ des indices $\{1, ..., m\}$ qui fait correspondre à la partition $\mathcal{P} = \{\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_m\}$, la partition $\mathcal{P}_\sigma = \{\mathcal{E}_{\sigma(1)}, \mathcal{E}_{\sigma(2)}, \ldots, \mathcal{E}_{\sigma(m)}\}$, $H(\mathcal{P}) = H(\mathcal{P}_\sigma)$

4. *Indépendance*: H est une mesure dite indépendante si pour tout couple de partitions $\mathcal{P} = \{\mathcal{E}_1, \mathcal{E}_2, ...\mathcal{E}_m\}$, $\mathcal{P}' = \{\mathcal{E}'_1, \mathcal{E}'_2, \ldots, \mathcal{E}'_{m-1}\}$ telles que $\exists i, j, k,\ \mathcal{E}_k = \mathcal{E}'_i \cup \mathcal{E}'_j$ et $\mathcal{P} \setminus \{\mathcal{E}_k\} = \mathcal{P}' \setminus \{\mathcal{E}'_i, \mathcal{E}'_j\}$ alors $H(\mathcal{P}) - H(\mathcal{P}')$ ne dépend que de $\mathcal{E}'_i, \mathcal{E}'_j$ et $\mathcal{E}_k$: $H(\mathcal{P}) - H(\mathcal{P}') = H(\{\mathcal{E}_k\}) - H(\{\mathcal{E}'_i, \mathcal{E}'_j\})$.

Ces propriétés sont celles qui sont le plus souvent exigées des mesures de discrimination. En fait, elles sont basées sur l'analogie qui existe entre ces mesures et les mesures d'information et, par conséquent, c'est en référence à ces mesures d'information que les propriétés sont examinées. Une telle analogie n'a rien d'extraordinaire si on considère que la théorie de l'information est l'une des bases de l'informatique. Pendant près de 40 ans, de nombreux travaux lui ont été consacrés ce qui ne peut pas avoir été sans conséquences pour les diverses branches modernes de l'informatique.

Une nouvelle méthode constructive de validation

Dans cette thèse, nous proposons une nouvelle méthode d'étude des fonctions de discrimination. Sans refuser les apports des méthodes classiques présentées, nous pensons qu'une méthode de classification des mesures de discrimination peut être faite avec une optique différente.

Cette partie sur l'étude des mesures sort du simple cadre de la construction d'arbres de décision car elle a aussi des apports intéressants pour le domaine plus général de l'apprentissage inductif, c'est pourquoi cette étude n'est pas développée dans ce chapitre ci, mais elle constitue un chapitre à part entière de notre thèse (le chapitre 6).

2.2.3 Critère d'arrêt

Un critère d'arrêt permet de jauger un ensemble donné et de savoir si cet ensemble est suffisamment ordonné relativement à la classe pour arrêter le développement de l'arbre et constituer une feuille. Ce critère n'a pas donné lieu à beaucoup de travaux dans le domaine de arbres de décision, on peut citer (Boyen et Wehenkel, 1996). En règle générale, c'est la mesure de discrimination sous sa forme brute (non conditionnelle) qui est utilisée pour mesurer le désordre ou l'impureté de l'ensemble courant. Soit un ensemble d'exemples $\mathcal{E}$, les différentes raisons d'arrêter le développement de l'arbre sur cet ensemble, et donc de se servir de cet ensemble comme une feuille, sont :

- tous les exemples de $\mathcal{E}$ ont la même classe, ou, au moins, un *grand nombre* d'exemples ont la même classe. C'est cette condition qui appelle l'utilisation d'une mesure d'impureté ou d'information telle l'entropie de Shannon.

- le nombre d'exemples dans $\mathcal{E}$ est trop faible pour continuer à le partitionner.

Le critère d'arrêt est souvent une combinaison de ces deux raisons. La mesure de discrimination est utilisée pour mesurer la pureté de l'ensemble en question. Un seuil peut être utilisé pour autoriser l'arrêt quand une *majorité* d'exemples a la même classe.

2.2.4 Stratégie de partitionnement

Comme il est dit dans l'introduction, dans le cadre classique il n'existe qu'une seule stratégie de partitionnement. Soit une base d'apprentissage $\mathcal{E}$ composée de n exemples $\{e_1, ..., e_n\}$ et un attribut A avec m modalités possibles $v_1, ..., v_m$. Partitionner $\mathcal{E}$ grâce à l'attribut A revient à créer autant de sous-bases qu'il existe de valeurs pour A, chaque sous-base étant composée par les exemples possédant une valeur identique pour A. En termes plus formels, la partition de $\mathcal{E}$ par A est telle que $\mathcal{E} = \mathcal{E}_1 \cup ... \cup \mathcal{E}_m$ avec $\forall l = 1, ...m,\ \mathcal{E}_l = \{e_i \in \mathcal{E} \mid e_i(A) = v_l\}$ et comme on se place dans un cadre classique, $\forall i, j = 1, ...m,\ \mathcal{E}_i \cap \mathcal{E}_j = \emptyset$.

Dans le cadre des arbres flous, cette stratégie ne s'applique plus de la même façon. En effet, la partition de $\mathcal{E}$ n'est plus obligatoirement classique, mais elle peut être floue, ce qui oblige à tenir compte de cet aspect dans la répartition des exemples. En fait, dans ce cas là, il faut remarquer qu'il est très difficile de trouver une telle partition car il est rare que $\mathcal{E}_i \cap \mathcal{E}_j = \emptyset$. Ce sont donc les exemples de ces intersections entre les $\mathcal{E}_i$ qu'il faut affecter à une des sous-bases induites par la partition sur les valeurs de l'attribut.

La façon la plus courante de procéder est de répartir les exemples dans toutes les sous-bases en les affectant d'un degré d'appartenance. Cette répartition est faite de façon explicite comme, par exemple, dans les méthodes de (Ramdani, 1994) ou de (Janikow, 1994). Elle peut aussi être faite de façon implicite, c'est-à-dire qu'elle est réalisée dans le calcul du pouvoir de discrimination d'un attribut sans que les exemples soit explicitement répartis en plusieurs sous-ensembles, par exemple dans la méthode de (Yuan et Shaw, 1995).

Une autre façon de procéder est de créer une partition classique à partir du cas flou en utilisant une α-coupe. Par exemple, avec $\alpha = \frac{1}{2}$, la partition créée est telle que :

$\forall l = 1, ...m,\ \mathcal{E}_l = \{e_i \in \mathcal{E} \mid \mu_{v_l}(e_i(A)) \geq \frac{1}{2}\}$. C'est cette méthode qui est utilisée dans la version actuelle de notre système Salammbô.

D'autres méthodes existent, par exemple, il est possible de dupliquer les exemples de l'intersection floue dans les deux sous-bases induite par les deux valeurs, en leur associant un degré d'appartenance de 1. Cette méthode a l'inconvénient de dupliquer la partie la plus imprécise et incertaine, voire bruitée, des données.

Une autre méthode est de créer une branche spécifique pour l'intersection floue où les exemples de cette intersection serviront à construire un sous-arbre pour les discriminer. Cette méthode est utilisée par (Henrichon et Fu, 1969) dans leur système de reconnaissance. Certes, ils ne se placent pas dans le domaine des arbres de décision, mais leur façon de procéder, par partitionnement successifs et combinés, est une méthode équivalente aux méthodes par arbres. Ils utilisent des modules de discrimination qui séparent l'univers des valeurs. Certaines régions ne sont discriminées que par la combinaison de modules, ce qui reproduit le processus des arbres de décision.

2.2.5 Des exemples de méthodes existantes

Comme il a été dit, les méthodes se différencient entre elles par la mesure de discrimination qu'elles utilisent.

Il existe les méthodes classiques qui ne peuvent pas prendre en compte les données numériques-symboliques pour construire un arbre de décision. Ces méthodes restent intéressantes à connaître car elles ont donné naissance, par extension, aux méthodes floues. Ainsi, la plupart des méthodes floues actuelles utilisent une mesure entropique d'ensembles flous dérivée de la mesure classique de Shannon. Peu de méthodes floues se distinguent en utilisant une mesure d'origine différente. L'attrait mystique de la notion d'entropie dispense alors les auteurs de ces méthodes d'une étude approfondie des propriétés de la mesure ainsi étendue.

Les méthodes classiques

Parmi les méthodes classiques, c'est-à-dire qui ne traitent pas de données floues, les deux méthodes les plus utilisées sont l'algorithme ID3 et la méthode CART.

La méthode la plus connue de construction d'arbres de décision est l'algorithme ID3 de (Quinlan, 1984; Quinlan, 1986; Quinlan, 1993). C'est à partir de l'article introductif de cette méthode que les arbres de décision ont refait leur apparition dans l'ère moderne de la théorie informatique de l'apprentissage.

Comme dans la théorie des questionnaires, la mesure de discrimination utilisée dans cet algorithme est l'entropie de Shannon (équation 2.3). Mais, Quinlan s'est rendu compte du problème d'utiliser une telle mesure en présence d'attributs numériques dans la base d'apprentissage. Étant donné que la mesure d'entropie a tendance à favoriser les attributs possédant un grand nombre de modalités, il apparaît que les attributs numériques sont plus souvent choisis à cause de leur grand nombre de valeurs. La mesure d'entropie de Shannon d'un ensemble d'événements n'est dépendante que de la probabilité individuelle de chaque événement. Si chaque événement de l'ensemble ne possède qu'une classe, l'entropie conditionnelle à chaque événement de cet ensemble relativement à la classe est nulle, quel que soit le nombre d'éléments dans l'ensemble.

Quinlan a alors proposé l'introduction dans la mesure de discrimination d'un facteur lié au nombre d'éléments dans l'ensemble considéré. C'est l'idée du *gain ratio*. Pour cela, il utilise une seconde mesure d'entropie, la mesure de l'information nécessaire pour répondre à la question: "Quelle est la valeur de l'attribut A_j ?", qui est donnée par[5] :

$$H_S(A_j) = -\sum_{l=1}^{m_j} P(v_{jl}) \log(P(v_{jl}))$$

Le *gain ratio* est alors le rapport de la variation d'information apportée par l'attribut A_j et la valeur de $H_S(A_j)$:

$$H_Q(C|A_j) = \frac{H_S(C|A_j)}{H_S(A_j)} \tag{2.4}$$

La valeur de la fonction $H_S(A_j)$ est d'autant plus grande que l'attribut possède un grand nombre de modalités. Par conséquent, la mesure de son pouvoir discriminant s'en trouve réduite d'autant plus.

La seconde méthode la plus connue de construction d'arbres de décision est issue du domaine de l'analyse de données et des statistiques : l'algorithme CART de (Breiman et al., 1984). Les auteurs proposent des critères de choix pour une bonne mesure d'impureté et ils peuvent proposer ainsi différentes mesures pour construire un arbre. Outre l'entropie de Shannon, ils proposent une mesure d'impureté basé sur le test de Gini[6] :

$$H_G(C|A_j) = \sum_{l=1}^{m_j} P(v_{jl}) \sum_{k_1 \neq k_2} P(c_{k_1}|v_{jl}) P(c_{k_2}|v_{jl}) \tag{2.5}$$

Ce test vérifie toutes les propriétés requises pour être une bonne mesure d'impureté et c'est ce test que les auteurs utilisent pour construire leur arbre. De plus, pour traiter les données numériques, ils introduisent la notion de discrétisation dynamique des univers des valeurs des attributs numériques. La discrétisation est dynamique dans le sens où elle est effectuée à chaque nœud de l'arbre, durant son développement et non pas en une seule et unique fois avant le lancement de l'algorithme. L'univers numérique est découpé en plusieurs parties (généralement deux dans le cas de CART) par le calcul de seuils de coupure. Ces seuils sont choisis afin de minimiser une mesure d'impureté.

Il existe d'autres méthodes de construction d'arbres de décision utilisant des mesures différentes. Ces méthodes ne sont pas développées ici. On peut quand même citer la

5. (Quinlan, 1986) nomme cette mesure IV et utilise la notation $IV(A_j) = -\sum_{l=1}^{m_j} P(v_{jl}) \log(P(v_{jl}))$. Pour homogénéiser nos notations, nous préférons utiliser la notation sous la forme de la fonction de Shannon.

6. Le *test de Gini* $i(v_{jl})$, ou *index de diversité de Gini*, de la valeur v_{jl} relativement à l'ensembles des modalités c_k de la classe C est donné (Breiman et al., 1984) par :

$$i(v_{jl}) = \sum_{k_1 \neq k_2} P(c_{k_1}|v_{jl}) P(c_{k_2}|v_{jl})$$

méthode de (Utgoff et Clouse, 1996) qui utilise la distance de Kolmogorov-Smirnov (équation 2.14) comme mesure de discrimination. Il faut aussi citer la mesure proposée par (López de Mántaras, 1991). Dans son article, (López de Mántaras, 1991) revient sur le problème de la mesure utilisée dans l'algorithme ID3 quand les attributs possèdent un grand nombre de modalités. Il propose alors une mesure basée sur un calcul de distance entre les partitions :

$$H_L(C|A_j) = \frac{H_S(C|A_j) + H_S(A_j|C)}{H_S(A_j \wedge C)} \tag{2.6}$$

L'auteur démontre alors que cette mesure est tout aussi efficace que le *gain ratio* de Quinlan, tout en étant formellement prouvée de façon plus rigoureuse.

Une méthode originale de construction d'arbres de décision en présence d'attributs numériques est celle proposée par (Van de Merckt, 1993). Il se propose d'incorporer une mesure de similarité dans la mesure de discrimination pour favoriser les partitions les moins similaires. La mesure de similarité qu'il utilise est une mesure de contraste entre les partitions. L'hypothèse pour calculer une mesure de contraste est que l'ensemble des valeurs numériques de l'attribut A_j soit décomposé en deux partitions engendrées par les classes. Dans le cas où la classe C peut prendre deux valeurs c_1 et c_2, elles induisent une partition en deux ensembles : l'ensemble des valeurs ayant la classe c_1 et l'ensemble des valeurs ayant la classe c_2. La mesure de contraste retourne alors le contraste qui sépare les deux partitions :

$$H_{V_1}(C|A_j) = \frac{|c_1|.|c_2|}{|c_1 \cup c_2|}(\bar{v}_{j1} - \bar{v}_{j2})^2 \tag{2.7}$$

avec $\bar{v}_{j1}$ (resp. $\bar{v}_{j2}$) la moyenne des valeurs de A_j possédant la classe c_1 (resp. c_2) et $|c_1|$ leur nombre (resp. $|c_2|$).

Van de Merckt propose de combiner cette mesure de contraste avec une mesure d'entropie pour obtenir une mesure de discrimination :

$$H_{V_2}(C|A_j) = \frac{H_{V_1}(C|A_j)}{H_S(C|A_j)} \tag{2.8}$$

Finalement, dans des premiers travaux, (Cios et Liu, 1992) introduisent un algorithme de construction d'arbres de décision en présence de données numériques. Cet algorithme, baptisé *Continuous ID3*, est basé sur l'utilisation de l'algorithme ID3 associé à des éléments de la théorie des réseaux de neurones. Il introduit une discrétisation des attributs numériques à l'aide d'une surface de séparation qui n'est pas forcément linéaire. Le résultat de la séparation est mesuré par l'utilisation de la mesure d'entropie, comme le font (Breiman et al., 1984).

Après ce rapide survol des méthodes classiques de construction d'arbres de décision, nous allons examiner les différentes méthodes existantes pour construire des arbres en présence de données floues.

Les méthodes floues utilisant l'entropie étoile

En présence de données numériques-symboliques, la mesure la plus généralement utilisée, à quelques rares exceptions près, est une extension de l'entropie de Shannon, l'*entropie d'événements flous*, aussi nommée *entropie étoile* ou *entropie star*.

$$H_S^*(C) \quad = \quad - \sum_{k=1}^{K} P^*(c_k) \log(P^*(c_k)) \tag{2.9}$$

Cette mesure est une extension de l'entropie de Shannon obtenue en substituant à la mesure de probabilité classique une mesure de probabilité d'événements flous. La mesure de probabilité d'événements flous généralement utilisée est celle proposée par (Zadeh, 1968) :

Définition 1 *Étant donné un ensemble flou $\mathcal{V} = \{e_1, ..., e_n\}$ de fonction d'appartenance μ, chaque élément e_i de $\mathcal{V}$ est associé à une probabilité (classique) d'occurrence $P(e_i)$. $\mathcal{V}$ est assimilé à un événement flou. La* probabilité de l'événement flou *$\mathcal{V}$ est définie par :*

$$P^*(\mathcal{V}) = \sum_{i=1}^{n} \mu(e_i) P(e_i) \tag{2.10}$$

Comme il l'a été montré, la mesure P^* est bien une mesure de probabilité (Zadeh, 1968), (Kaufmann, 1977), (Smets, 1982).

Il est à noter que la mesure d'entropie d'un événement flou relativement à une distribution de probabilité proposée par Zadeh dans son article initial (Zadeh, 1968) est de la forme :

$$H_Z(C) \quad = \quad - \sum_{k=1}^{K} \mu(e_i) P(c_k) \log(P(c_k)) \tag{2.11}$$

Comme le précise Zadeh, cette entropie ne généralise pas l'entropie classique de Shannon. Cela explique peut être le fait que cette mesure n'a jamais, à notre connaissance, été utilisée comme mesure de discrimination dans la construction d'arbres de décision.

La mesure d'entropie d'événements flous $H_S^*(C)$ a été introduite par (Tanaka et al., 1979) comme mesure d'information floue. Elle a ensuite été utilisée dans plusieurs systèmes de construction d'arbres de décision flous (Ramdani, 1994), (Umano et al., 1994), (Weber, 1992), (Janikow, 1994).

La méthode SAFI de (Ramdani, 1992; Ramdani, 1993; Ramdani, 1994) a été une des premières méthodes de construction d'arbres de décision flous dans le domaine de l'apprentissage en présence de données numériques-symboliques. L'arbre est construit en utilisant l'entropie étoile comme mesure de discrimination pour trouver les meilleurs attributs pour partitionner la base d'apprentissage. Dans (Ramdani, 1994), une étude de certaines propriétés de l'entropie étoile est faite afin de la valider en tant que mesure de discrimination. Il est alors montré que cette mesure est valable en terme de théorie de l'information, au sens de (Aczél et Daróczy, 1975). C'est une des rares références où une validation formelle du choix de ce type d'extension de l'entropie de Shannon est faite pour valider son utilisation dans la construction d'un arbre de décision flou. La méthode SAFI est de plus basée sur l'utilisation de partitions floues données sur les univers de chaque attribut numérique. Ces modalités floues sont censées être fournies par un expert du domaine étudié. L'inconvénient dans ce cas là est que les modalités de l'expert ne soient pas optimalement choisies pour résoudre le problème posé. (Ramdani, 1994) propose une méthode d'optimisation de ces partitions basée sur l'utilisation de la mesure d'entropie étoile. Dans le même esprit que (Breiman et al., 1984), l'entropie étoile sert à comparer différentes partitions floues possibles entre elles. La meilleure partition est celle qui minimise la mesure d'entropie étoile. L'inconvénient de cette méthode d'optimisation de partitions floues est qu'elle favorise les partitions non floues sur l'univers des valeurs[7]. Ramdani propose alors l'utilisation d'un *degré d'étalement* minimal entre les noyaux des partitions floues étudiées.

À la même époque, (Weber, 1992) propose la même méthode de construction d'arbres de décision flous en utilisant l'entropie étoile pour discriminer les attributs entre eux.

De leur côté, (Umano et al., 1994) proposent eux aussi une méthode de construction d'arbres de décision flous. Leur algorithme de construction est très proche de celui du système SAFI. Il est aussi basé sur l'utilisation de modalités floues attachées à l'univers des valeurs d'un attribut numérique. Ces modalités sont aussi données par l'utilisateur. Leur système utilise un cas particulier de l'entropie étoile comme mesure de discrimination des attributs. Les probabilités des événements sont calculées de façon fréquentielle en fonctions des valeurs de la base d'apprentissage. Ensuite, ils utilisent l'arbre de décision flou pour classer en se servant des valeurs floues qui libellent les nœuds de l'arbre. Les inférences sont calquées sur la méthode classique de classification avec un arbre de décision. Ils utilisent le produit pour réaliser le “et” sur une même branche, mais, un point étonnant pour un cadre flou est qu'ils se servent de l'addition, qui n'est pas une t-conorme, pour réaliser le “ou” entre des branches de l'arbre.

Leur méthode de construction d'arbres de décision flous est reprise par (Shibata et al., 1995; Shibata, 1997). Dans leur domaine de la robotique, ils utilisent des arbres de décisions flous pour l'extraction de caractéristiques pertinentes à partir d'un ensemble de valeurs numériques. Les ensembles flous sur les univers des valeurs numériques sont déterminés dans une étape préalable par une méthode automatique.

Le système FIDMV de (Janikow, 1994; Janikow, 1996) se propose lui aussi de construire des arbres de décision flous à partir de données numériques sur lesquelles sont définies des modalités floues. L'entropie étoile est toujours la mesure de discrimination utilisée pour trouver le meilleur attribut. Pour optimiser les partitions floues des attributs, une méthode basée sur les algorithmes génétiques est proposée (Janikow, 1995).

(Ichihashi et al., 1996) reprennent l'algorithme de construction d'arbres de décision flous de (Umano et al., 1994) en y introduisant une méthode pour construire des modalités floues quand elles ne sont pas disponibles. Cette méthode est basée sur la théorie de l'évidence et sur le calcul d'une fonction de répartition de masse de croyance, pour chaque valeur d'un attribut numérique, dans une des classes. Les fonctions de croyance sont construites à l'aide d'un système d'équation linéaire à résoudre avec

7. Cet inconvénient sera aussi soulevé par (Boyen et Wehenkel, 1996) qui démontrera, d'une façon plus générale, que c'est le problème de toute mesure convexe d'ensemble flou qui atteint son optimum quand l'ensemble est non flou.

comme contrainte la minimisation de l'entropie du système.

Toutes ces méthodes ont donc comme point commun d'utiliser la même mesure de discrimination, à savoir l'entropie étoile. Certes, ces méthodes diffèrent encore sur quelques points comme l'optimisation des partitions floues ou l'utilisation de l'arbre pendant la classification de nouveaux cas. Mais globalement, les mêmes arbres seront construits. Il est intéressant de remarquer l'explosion des méthodes de construction d'arbres de décision flous qui a lieu depuis 1992. Mais intrinsèquement, ces méthodes ne diffèrent que sur un aspect extérieur à la construction : la définition et l'optimisation des modalités floues utilisées.

Les méthodes suivantes sont différentes car elles utilisent des mesures de discrimination qui ne sont pas nécessairement basées sur une généralisation de l'entropie de Shannon aux ensembles flous.

Les méthodes floues utilisant une mesure différente de l'entropie étoile

Les travaux de (Cios et Liu, 1992) relatifs à la construction d'arbres de décision en présence de données numériques ont déjà été présentés dans la partie concernant les méthodes classiques. Dans des travaux ultérieurs, (Cios et Sztandera, 1992) proposent de remplacer la mesure d'entropie classique utilisée par des mesures floues comme la mesure de floue de (De Luca et Termini, 1972) ou l'entropie floue de Kosko (Kosko, 1997) couplée aux opérateurs généralisés de Dombi pour réaliser l'union et l'intersection floues.

Étant donné un ensemble flou $\mathcal{V} = \{e_1, ..., e_n\}$ de fonction d'appartenance μ. L'entropie de l'ensemble flou $\mathcal{V}$ définie par De Luca et Termini est :

$$H_D(\mathcal{V}) \quad = \quad -\sum_{i=1}^{n} \left(\mu(e_i)\log(\mu(e_i)) + (1-\mu(e_i))\log(1-\mu(e_i))\right) \tag{2.12}$$

Cette mesure n'est pas construite pour être une mesure de discrimination, mais pour mesurer le degré de flou de l'ensemble $\mathcal{V}$.

L'entropie proposée par Kosko est définie par :

$$H_K(\mathcal{V}) \quad = \quad \frac{\Sigma_{\text{count}}(\mathcal{V} \cap \bar{\mathcal{V}})}{\Sigma_{\text{count}}(\mathcal{V} \cup \bar{\mathcal{V}})} \tag{2.13}$$

où Σ_{count} est la cardinalité d'ensembles flous de Zadeh, et $\bar{\mathcal{V}}$ est le sous-ensemble flou complémentaire de $\mathcal{V}$. Comme l'entropie de (De Luca et Termini, 1972), l'entropie de Kosko est surtout une mesure du degré de flou d'un ensemble flou plutôt que sa mesure d'entropie.

Dans un autre registre de mesures, (Araya, 1994) propose la construction d'arbres de décision en présence de données et de classes floues en utilisant une extension de la mesure de Kolmogorov-Smirnov. Dans un cadre où il n'existe que deux classes à reconnaître, la mesure de Kolmogorov-Smirnov retourne la distance maximale qui sépare les distributions de probabilité de chacune des classes :

$$\mathcal{K}(c_1, c_2) = \max_x(|P_{c_1}(x) - P_{c_2}(x)|) \tag{2.14}$$

avec P_{c_1} (resp. P_{c_2}) la distribution de probabilité de la classe c_1 (resp. c_2). Araya propose d'étendre cette mesure pour la prise en compte de classes floues en se servant, comme pour l'extension de l'entropie de Shannon, de la cardinalité d'ensembles flous de Zadeh.

Dans leur domaine d'application des réseaux électriques, après s'être rendu compte des limites des mesures classiques pour la prise en compte des données floues, Boyen et Wehenkel ont proposé deux méthodes pour la prise en compte des données numériques-symboliques (Boyen, 1995; Boyen et Wehenkel, 1995; Boyen et Wehenkel, 1996). La première méthode, baptisée *Outbound Fuzzy Decision Trees* (OFDT), se propose d'étendre l'espace des discriminateurs pour les adapter aux mesures existantes. La seconde méthode, *Fussy Fuzzy Decision Trees* (FFDT), se propose de définir de nouvelles mesures de discrimination. C'est la mesure de Kolmogorov-Smirnov qui sert de base à leurs deux types de méthodes.

Concernant un autre type de mesures, (Yuan et Shaw, 1995) proposent une méthode pour construire un arbre de décision flou. Leur algorithme utilise comme mesure de discrimination, une mesure d'ambiguïté de classification[8] qui est fondée sur la mesure de satisfiabilité (équation 2.18) que nous utilisons au paragraphe 2.5.1, ainsi que sur une mesure de non-spécificité.

Soit une base d'apprentissage $\mathcal{E}$, comportant n exemples e_i, décrits par un attribut A_j à m_j modalités v_{jl}. Chaque exemple est associé à une modalité c_k de la classe C. Yuan et Shaw définissent la mesure d'ambiguïté de classification par :

$$H_Y(C|A_j) = g(\Pi(C|A_j)) \tag{2.15}$$

Où $\Pi(C|A_j) = \{\pi(c_k|A_j),\ k = 1, .., K\}$ est la distribution de possibilité de la modalité c_k relativement à l'attribut A_j, et g est une mesure de non-spécificité. Chaque $\pi(c_k|A_j)$ est défini par :

$$\pi(c_k|A_j) = \frac{S(A_j, c_k)}{\max\limits_{i=1,..,K} S(A_j, c_i)}$$

La mesure de satisfiabilité[9] S d'une modalité w relativement à la modalité v est définie par :

$$S(w, v) = \frac{\sum\limits_{x \in X} \mu_{w \cap v}(x)}{\sum\limits_{x \in X} \mu_w(x)}$$

La mesure de non-spécificité g, Yuan et Shaw disent aussi *mesure possibiliste d'ambiguïté*, d'une distribution de possibilité $\Pi = \{\pi_1, ..., \pi_M\}$ est définie par :

$$g(\Pi) = \sum_{i=1}^{M} (\pi_i^* - \pi_{i+1}^*) \log i$$

8. En anglais : *classification ambiguity.*

9. On utilise ici l'appellation introduite dans la section 2.5.1, Yuan et Shaw utilisent le terme de *fuzzy subsethood.*

Où la distribution $\Pi^* = \{\pi_1^*, ..., \pi_{M+1}^*\}$ est une permutation des π_i telle que pour tout $i = 1, .., M$, $\pi_i^* \geq \pi_{i+1}^*$ et $\pi_{M+1}^* = 0$.

Dans leur article, Yuan et Shaw proposent aussi une méthode pour estomper[10] un ensemble de valeurs numériques dont nous reparlerons dans notre chapitre sur la construction de partitions floues (chapitre 4). Pour classifier, l'arbre de décision flou est transformé en base de règles.

Pour terminer, il faut aussi citer la méthode de construction d'arbres de décision flou de (Bothorel, 1996). Dans son système, le choix de l'attribut le plus discriminant s'effectue par l'intermédiaire d'une mesure de contraste entre deux sous-ensembles flous:

$$\mathcal{C}(\mathcal{V}_1, \mathcal{V}_2) = \frac{\int \min(\mu_{V_1}, \mu_{V_2})}{\int \max(\mu_{V_1}, \mu_{V_2})} \tag{2.16}$$

Dans son système, Bothorel choisit l'attribut qui minimise $\mathcal{C}(\mathcal{V}_1, \mathcal{V}_2)$ où $\mathcal{V}_k$ est le sous-ensemble flou correspondant à la classe c_k sur l'univers des valeurs de l'attribut en question.

2.2.6 Choisir une méthode

Comme il a été vu dans les parties précédentes, les méthodes de construction d'arbres de décision se différencient essentiellement par la mesure qu'elles utilisent. Choisir sa méthode revient donc à sélectionner une mesure de discrimination parmi les autres. Cette sélection n'est pas facilitée par les méthodes de validation de mesures actuelles. S'il est possible de déterminer si une mesure est une bonne mesure de discrimination, il est plus délicat d'arriver à savoir si une mesure est meilleure qu'une autre pour un problème de sélection d'attribut.

C'est pourquoi, il paraît intéressant de proposer une nouvelle façon d'étudier les mesures de discrimination. Il faut arriver à discerner les structures intrinsèques d'une mesure de comparaison, son mode de fonctionnement et de variations.

Toutes ces études et réflexions ont amené l'architecture de construction de mesures de discrimination présentée au chapitre 6 de cette thèse.

2.3 L'élagage des arbres de décision

L'élagage, ou *pruning* en anglais, des arbres de décision permet de simplifier un arbre déjà construit. En effet, souvent l'arbre résultant d'un algorithme de construction est très complexe, autant en nombre de branches qu'en profondeur.

Il existe deux types d'élagage, l'élagage en tant que tel s'effectue après la construction de l'arbre. Un second type d'élagage d'arbres de décision existe, c'est l'élagage mis en œuvre durant la construction de l'arbre lorsque le critère d'arrêt autorise explicitement l'arrêt du développement d'une branche quand l'ensemble courant n'est pourtant pas

10. `estomper`: *v. tr.* Étendre avec une estompe le crayon sur le papier, *Par ext.* Couvrir d'une ombre légèrement dégradée. *Fig.* Adoucir, voiler. *(Larousse de Poche)*

parfaitement classé. Afin de simplifier l'arbre, c'est donc une étape très utile. Par contre, les meilleurs moyens d'élagage ne sont pas facilement définissables.

Élaguer c'est d'abord soit couper les branches trop peu représentatives, celles qui n'ont que peu de chances d'être utiles. C'est aussi, regrouper des branches entre elles en fonction de la ressemblance qu'elles possèdent. Ainsi, déjà au niveau des arbres de décision flous, (Ramdani, 1994) étudiait la meilleure façon de mesurer la similarité entre des valeurs similaires afin de les regrouper en une seule valeur unique.

Dans cette thèse, les aspects concernant l'élagage d'un arbre de décision après sa construction ne sont pas traités. Le seul type d'élagage que nous considérons concerne l'élagage en profondeur de l'arbre de décision, durant sa construction. C'est le choix du critère d'arrêt qui est déterminant dans ce type d'élagage. Un premier choix est d'arrêter le développement d'une branche même quand tous les exemples de la base courante ne possèdent pas la même classe. Que ce soit en fixant un seuil pour la mesure d'impureté au lieu de ne s'arrêter que quand la mesure de la base courante est nulle, ou bien en fixant un nombre minimal d'exemples dans la base pour la partitionner encore. Cette technique d'élagage est plus performante en présence d'arbres de décision flous parce que, lors de la classification, tous les exemples compris dans une feuille sont utilisés pour la classification.

Comme techniques plus élaborées d'élagage, on peut citer la méthode utilisée par (Breiman et al., 1984) dans leur système CART ou la méthode d'élagage de (Wehenkel, 1993), basée sur l'utilisation d'une mesure de qualité pour rendre compte de l'apport en terme d'information des sous-arbres issus d'un nœud. Si ces sous-arbres sont peu informatifs relativement à la classe, ils sont coupés et le nœud devient un nœud terminal. De même, (Mehta et al., 1995) utilise une méthode basée sur le principe de longueur de description minimum (MDL).

L'article de (Mingers, 1989a) peut lui être consulté pour un survol de différentes méthodes d'élagage d'arbres de décision. Finalement, pour les arbres de décision flous, (Barone, 1992) utilise l'analogie existante entre les arbres de décision et un ensemble de règles pour élaguer les arbres.

Dans le cadre des graphes d'induction, (Rabaséda-Loudcher, 1996) utilise aussi cette analogie entre un arbre et un ensemble de règles pour élaguer les graphes obtenus par le système SIPINA. Sa méthode est basée sur un système de transformation de prémisses de règles, construit sur le calcul propositionnel.

2.4 Utilisation d'un arbre de décision pour le classement d'objets

2.4.1 La méthode classique

Classiquement, un arbre de décision est un outil très pratique et très facile à utiliser pour classer de nouveaux cas. Un nouveau cas à classer est une description qui n'est pas associée à une classe, c'est cette classe que l'arbre de décision permet de trouver. Du moins, c'est le processus inductif, qui, après avoir restructuré la connaissance en termes de liens d'association entre les descriptions et les classes des exemples de la base d'apprentissage, permet de généraliser ces liens et de déduire la classe associée à n'importe quelle description.

Le nouveau cas à classer est présenté séquentiellement aux nœuds de l'arbre de décision. Il est d'abord présenté à la racine où un test est réalisé sur la valeur qu'il possède pour l'attribut qui libelle le test associé à ce nœud. Selon le résultat de ce test, le cas suit alors l'arc libellé par ce résultat pour atteindre soit un nouveau nœud pour lequel le processus de comparaison est réitéré, soit une feuille. Dans cette dernière situation, la classification est terminée : le cas est associé à la classe libellant cette feuille.

Par exemple, soit la description `[Pièce = Tour, Temps Restant = 56s, Pions = peu]` pour laquelle il serait intéressant de trouver la classe, et ceci grâce à l'arbre de décision donné dans la figure 2.1. La racine est un test sur l'attribut `Pièce`, la valeur de la description pour cet attribut est `Tour`, le chemin suivi est donc celui dont l'arc est libellé par cette valeur. Le nœud atteint est celui réalisant un test sur l'attribut `Temps_restant`. La valeur de la description pour cet attribut est `56s`, par conséquent, l'arc libellé par `Temps_restant <60s` est donc suivi, pour atteindre la feuille libellée `Perdu`. Ainsi donc, la description `[Pièce = Tour, Temps_restant = 56s, Pions = peu]` est associée à la classe `Perdu`. La figure 2.4 résume cette suite de tests et montre le chemin suivi pour classer cette description.

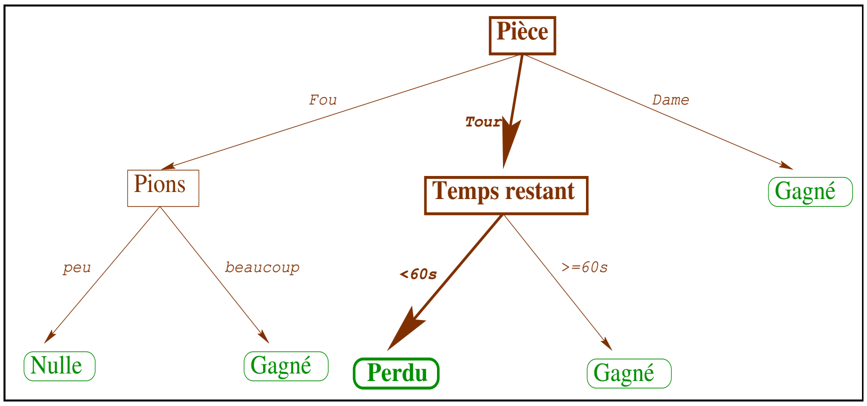


Figure 2.4 – *Exemple de classification avec un arbre de décision*

Cette manière d'utiliser un arbre de décision pour classer de nouveaux cas est la plus classique et la plus naturelle des méthodes. Il faut se rappeler que les arbres de décision sont un outil connu et utilisé dans des domaines qui ne sont pas exclusivement liés à l'informatique, par exemple en biologie, en médecine... C'est de la facilité de leur utilisation que les arbres de décision doivent certainement leur succès auprès de leurs utilisateurs.

2.4.2 Un arbre de décision comme une base de règles

Chaque chemin dans un arbre de décision réalise une série de tests sur les valeurs des attributs pour déduire la classe à associer aux valeurs testées. Un chemin est donc équivalent[11] à une règle de production `si [prémisses] alors [conclusion]`. La partie `prémisses` de cette règle contient une conjonction des tests à réaliser sur le chemin en question, et la partie `conclusion` contient la classe libellant la feuille du chemin. Ainsi, un arbre de décision est équivalent à un système à base de règles. Il est clair que les règles de cette base sont toutes différentes car elles sont produites par un arbre de décision dans lequel tous les chemins de la racine vers les feuilles sont uniques.

Chaque chemin dans l'arbre produit donc une règle r ayant la forme suivante :

$$\texttt{si } A_{l_1} = v_{l_1} \texttt{ et } A_{l_2} = v_{l_2} \texttt{ et } \ldots\texttt{et } A_{l_p} = v_{l_p} \texttt{ alors } C = c_k.$$

Dans cette règle, les $A_{l_1}, ..., A_{l_p}$ sont les attributs utilisés dans les nœuds du chemin, choisis dans la liste $\{A_1, ..., A_N\}$ de tous les attributs possibles, et les $v_{l_1}, ..., v_{l_p}$ sont les caractérisations respectives associées aux arcs du chemin pour les attributs $A_{l_1}, ..., A_{l_p}$.

L'exemple e à classer est caractérisé par une série de valeurs pour les attributs $A_1, ..., A_N$:

$$A_1 = w_1 \texttt{ et } A_2 = w_2 \texttt{ et } \ldots\texttt{et } A_N = w_N.$$

C'est à partir de cette caractérisation et de l'ensemble des règles produites par l'arbre de décision, que la classe c_e de e doit être déduite.

2.4.3 La méthode classique est un modus ponens

La méthode classique d'utilisation de l'arbre de décision en classification se réduit en fait à l'application du modus ponens à partir de l'ensemble de règles produit par l'arbre.

Le *modus ponens* est la méthode de déduction naturelle, issue de la logique classique, souvent utilisée pour formaliser le mode de pensée humain :

$$\begin{array}{lccc} \text{Règle :} & P & \Longrightarrow & C \\ \text{Observation :} & P & & \\ \hline \text{Déduction :} & & & C \end{array}$$

Connaissant la règle de causalité entre une prémisse et une conclusion ($P \Longrightarrow C$) et sachant que la prémisse P est observée, la règle du modus ponens statue que la conclusion C peut être déduite.

La notation sous la forme $P \Longrightarrow C$ est équivalente à la notation usuelle des règles de production `si P alors C` bien connue en Intelligence artificielle.

La classification à l'aide un arbre de décision reproduit fidèlement le processus de déduction du modus ponens. L'arbre de décision est un ensemble de règles `si-alors`, c'est-à-dire de règles $P_i \Longrightarrow C_i$, chaque règle correspondant à un chemin dans l'arbre de la racine vers une feuille. Chaque prémisse P_i est composée par une conjonction de tests sur des valeurs d'attributs rencontrés sur le chemin de l'arbre, et la conclusion C_i correspond à la classe libellant la feuille du chemin en question.

11. En fait, il faut plutôt dire qu'un chemin *implique* une règle de production, car il n'y a pas symétrie entre un chemin et une règle. En effet, l'ordre des prémisses est indifférent dans une règle alors que les tests dans un arbres sont hiérarchiquement disposés.

Lors de la classification d'un nouveau cas, la description de ce cas est comparée avec toutes les prémisses P_i des règles et est associée avec la classe C_i liée à la prémisse égale à la description. Pour une prémisse, la comparaison s'effectue en vérifiant les tests sur les valeurs d'attributs qui la composent avec les valeurs correspondantes dans la description.

Par exemple, l'arbre de la figure 2.1 définit 5 règles :

R1 : `si Pièce = Fou et Pions = Peu alors Nulle`
R2 : `si Pièce = Fou et Pions = Beaucoup alors Gagné`
R3 : `si Pièce = Tour et Temps Restant <60s alors Perdu`
R4 : `si Pièce = Tour et Temps Restant ≥60s alors Gagné`
R5 : `si Pièce = Dame alors Gagné`

La description `[Pièce = Tour, Temps Restant = 56s, Pions = peu]` est comparée aux prémisses des 5 règles et il apparaît qu'elle coïncide avec la prémisse de la règle `R3`. En utilisant l'autre notation, il apparaît que la règle du modus ponens qui se déclenche est donc :

Règle :	`Pièce = Tour et Temps Restant` $< 60s$	$\Longrightarrow$	`Perdu`
Observation :	`Pièce = Tour et Temps Restant = 56s`		
Déduction :			`Perdu`

2.5 Notre méthode de classification avec un arbre de décision flou

En présence de données floues, autant au niveau de l'arbre qu'au niveau des observations, il faut prendre en compte un nouveau type de problème. Dans le cadre classique, un test sur une valeur d'attribut est respecté ou n'est pas respecté. Dans le domaine des données floues, cela n'est plus vérifié : le résultat d'un test peut être associé à une gradualité. Il faut donc être capable de mesurer et de prendre en compte le degré de satisfaction, la similarité, d'une nouvelle valeur vis-à-vis des tests de l'arbre de décision.

C'est une méthode basée sur une telle mesure qui est proposée dans cette section (Marsala et Ramdani, 1995; Bouchon-Meunier et al., 1997b). Pour un nouvel objet à classer, une observation, la similarité de chacune de ses valeurs vis-à-vis des modalités présentes dans les tests de l'arbre de décision est mesurée. La similarité de l'observation à chaque prémisse des règles issues de l'arbre de décision flou peut être alors calculée et permettre d'inférer des conclusions.

Dans cette section, après avoir présenté le moyen de mesurer la similarité entre une valeur de l'observation et une modalité, nous présentons la méthode d'inférence que nous utilisons pour inférer des conclusions à partir des règles issues d'un arbre de décision flou. Ensuite, nous terminons sur une étude des apports d'une telle méthode de classification floue vis-à-vis de la méthode classique en terme de stabilité dans la décision obtenue.

2.5.1 Mesurer la similarité entre deux valeurs

La *similarité* entre deux objets est le degré avec lequel ils se ressemblent selon un certain critère de ressemblance donné. Dans un cadre classique, deux objets se ressemblent, ou ne se ressemblent pas, il n'y a pas d'équivoque. Dans un cadre flou, il est admis que deux objets peuvent se ressembler plus ou moins, avec une certaine gradualité dans la ressemblance. Différentes méthodes existent pour mesurer la ressemblance entre deux valeurs, on peut citer par exemple les travaux sur le filtrage flou dans les bases de connaissances de (Dubois et Prade, 1982; Dubois et al., 1988; Dubois et al., 1990; Dubois et al., 1991), (Mo, 1990).

Il existe différents types de critères de ressemblance, chacun possédant des propriétés particulières. La hiérarchie proposée par (Bouchon-Meunier et al., 1996; Rifqi, 1996) classe les différentes mesures de ressemblance en fonction des propriétés qu'elles possèdent. Ainsi, en fonction des propriétés requises pour évaluer la ressemblance entre deux objets, il est possible de déterminer la forme la plus adéquate de mesure.

Afin de mesurer la similarité $s(w, v)$ entre la modalité w d'un attribut pour une observation et la modalité v du même attribut présente dans le test d'un nœud de l'arbre, les propriétés suivantes sont requises :

- $s(w, v) = 1$ quand $w \subseteq v$
- $s(w, v) = 0$ quand $w \neq v$
- $s(w, v)$ est croissante avec $w \cap v$

La première propriété demande que la similarité entre les deux valeurs soit égale à 1 au moins dans le cas où la valeur observée est égale ou complètement incluse dans la valeur test. La deuxième propriété demande que la similarité prenne la valeur zéro quand la valeur observée et la valeur test sont disjointes, c'est-à-dire qu'elles n'ont rien en commun. La dernière propriété paraît aussi naturelle : la similarité entre la valeur observée et la valeur test est d'autant plus importante que ces deux valeurs possèdent des parties communes.

Dans notre domaine, il faut préciser les définitions sous-jacentes sur lesquelles sont fondées ces propriétés.

Soit un univers X, la définition d'inclusion entre les classes de X utilisée est la suivante :

$$\forall v, w, \quad w \subseteq v \quad \Leftrightarrow \quad w \cap v = w$$

Ce qui se traduit pour des sous-ensembles flous de la façon suivante : pour toutes modalités floues v et w de fonctions d'appartenance respectives μ_v et μ_w, si $\mu_{w\cap v}$ est la fonction d'appartenance de l'intersection de ces deux sous-ensembles flous, on considère que $w \subseteq v \Leftrightarrow \mu_{w\cap v} = \mu_w$.

L'inégalité $w \neq v$ se définit à partir de l'intersection par :

$$\forall v, w, \quad w \neq v \quad \Leftrightarrow \quad w \cap v = v \cap w = \emptyset$$

En termes de fonctions d'appartenance : $w \neq v \Leftrightarrow \forall x \in X, \;\; \mu_{w\cap v}(x) = \mu_{v\cap w}(x) = 0$.

La croissance en $w \cap v$ se définit par le fait que

$$\forall v, w_1, w_2, \quad (w_1 \cap v) \subseteq (w_2 \cap v) \quad \Leftrightarrow \quad s(w_1, v) \leq s(w_1, v)$$

Dans la hiérarchie citée plus haut, l'intersection et l'inclusion de sous-ensembles flous sont définies classiquement à partir de l'opérateur minimum. Une mesure de ce type appartient, dans cette hiérarchie, à la famille des mesures de similitude, qui se décompose en trois sous-familles : les mesures de ressemblance, les mesures d'inclusion et les mesures de satisfiabilité. La mesure s se distingue des mesures de ressemblance par le fait qu'elle ne doit pas être nécessairement symétrique (*ie.* on ne demande pas que $\forall v, w,\ s(w, v) = s(v, w)$). Elle se distingue aussi des mesures d'inclusion par le fait que l'on souhaite que $s(w, v)$ dépende de $w - v$ (*ie.* les éléments qui appartiennent à w et n'appartiennent pas à v).

Par contre, la mesure s possède des critères suffisants pour être une mesure de satisfiabilité. En plus de satisfaire les conditions pour les cas limites, s est indépendante de $v - w$ (*ie.* les éléments qui appartiennent à v et n'appartiennent pas à w). Ainsi, elle est apte à rendre compte du degré avec lequel une valeur satisfait une valeur de référence. Pour la suite, le degré de satisfiabilité $s(w, v)$ entre la valeur observée pour un attribut w et la valeur de référence v est noté $\texttt{Deg}(w, v)$.

Un exemple d'une telle mesure de satisfiabilité est (Bouchon-Meunier et al., 1996; Rifqi, 1996) :

$$\texttt{Deg}(w, v) = \frac{\mathcal{M}(w \cap v)}{\mathcal{M}(w)} \tag{2.17}$$

où $\mathcal{M}$ est une mesure d'ensembles flous au sens de (Rifqi, 1996) :

Définition 2 (Mesure d'ensemble flou) *Étant donné Ω un ensemble d'éléments, $F(\Omega)$ l'ensemble des sous-ensembles flous de Ω, et étant donné $\subseteq$ une relation d'ordre sur les éléments de $F(\Omega)$. Une* mesure d'ensemble flou *$\mathcal{M}$ est une fonction définie sur $F(\Omega)$, à valeurs dans $I\!R^+$, telle que $\mathcal{M}(\emptyset) = 0$ et $\forall A, B \in F(\Omega)$, $B \subseteq A \Rightarrow \mathcal{M}(B) \leq \mathcal{M}(A)$.*

Cette mesure est une version non normalisée de l'évaluateur de sous-ensembles flous introduit par (Dubois et Prade, 1982).

Dans notre cas, à l'instar de (Ramdani, 1994), la mesure $\mathcal{M}(A)$ que nous choisissons dans le système Salammbô est la suivante. Pour tout $A \in F(\Omega)$, de fonction d'appartenance μ_A à valeurs dans $[0, 1]$ et définie sur $\Omega_A \subseteq \Omega$:

- si Ω_A est un univers continu de valeurs : $\mathcal{M}(A) = \displaystyle\int_{\Omega_A} \mu_A(x)\, dx$
- si Ω_A est un ensemble discret de valeurs : $\mathcal{M}(A) = \displaystyle\sum_{x \in \Omega_A} \mu_A(x)$

Dans ce cas, si Ω_A est un ensemble discret de valeurs, on retrouve la *relative sigma-count* de (Zadeh, 1983).

Pour simplifier les écritures, dans tout le reste de cette thèse, la notation $\displaystyle\int_{\Omega_A} \mu_A(x) dx$ sera utilisée à la fois pour le cas où Ω_A est continu mais aussi pour le cas où Ω_A est discret.

La mesure de satisfiabilité définie par la mesure $\mathcal{M}$ que nous choisissons est :

$$\mathtt{Deg}(w,v) = \frac{\displaystyle\int_X \mu_{w\cap v} dx}{\displaystyle\int_X \mu_w \, dx} \quad \text{si} \quad \int_X \mu_w \, dx \neq 0 \tag{2.18}$$

où μ_w est la fonction d'appartenance associée à la valeur w, $\mu_{w\cap v}$ est la fonction d'appartenance de l'intersection $w \cap v$ entre la valeur observée w et la valeur de référence v, X est l'univers, supposé fini, des valeurs sur lequel sont définies μ_w et $\mu_{w\cap v}$.

Dans le cas où $\int_X \mu_w dx = 0$, on pose[12] $\mathtt{Deg}(w,v) = 0$.

Cas d'une valeur précise de l'attribut

Dans tout système numérique-symbolique, apparaissent des valeurs floues et des valeurs précises d'une même attribut, selon les exemples. Toute modalité v associée à une branche de l'arbre de décision est floue mais la valeur w qui lui est comparée peut être précise : $w = \{x_0\}$. La seule mesure de satisfiabilité qui ait encore un sens est alors $\mu_v(x_0)$. Il est alors intéressant d'obtenir cette mesure en appliquant la définition générale de la mesure de satisfiabilité (équation 2.18).

Dans le cas où w se réduit à une seule valeur précise x_0 pour laquelle $\mu_w(x_0) \neq 0$, en notant $\top$ la norme triangulaire[13] qui réalise l'intersection des sous-ensembles flous, on a :

$$\mu_{w\cap v}(x_0) = \top(\mu_w(x_0), \mu_v(x_0))$$

et

$$\forall x \in X,\ x \neq x_0,\ \mu_w(x) = 0 \quad \text{soit} \quad \mu_{w\cap v}(x) = 0$$

car toute t-norme $\top$ vérifie la propriété suivante : $\forall a \in [0,1],\ \top(0,a) = 0$.

Le degré de satisfiabilité de la valeur précise w relativement à v est donc :

$$\mathtt{Deg}(w,v) = \frac{\displaystyle\sum_{x\in X} \mu_{w\cap v}(x)}{\displaystyle\sum_{x\in X} \mu_w(x)} = \frac{\top(\mu_w(x_0), \mu_v(x_0))}{\mu_w(x_0)}$$

Si de plus, $\mu_w(x_0) = 1$, comme il est souvent admis quand on parle de valeurs précises dans notre cadre, on obtient alors :

$$\mathtt{Deg}(w,v) = \top(1, \mu_v(x_0)) = \mu_v(x_0)$$

C'est vrai, par exemple, si la t-norme choisie pour réaliser l'intersection de sous-ensembles flous est la t-norme produit.

12. On rappelle que μ_w étant la fonction d'appartenance d'un sous-ensemble flou, c'est-à-dire que $\forall x \in X,\ \mu_w(x) \in [0,1]$, on a $\int_X \mu_w \, dx = 0 \Leftrightarrow \forall x \in X,\ \mu_w(x) = 0$ et donc, par conséquent, $w = \emptyset$. Le fait que X soit continu ou discret n'intervient pas ici.

13. Comme on le verra par la suite, le choix de la t-norme est très dépendant du choix de la mesure du degré de satisfiabilité.

Finalement, il faut rappeler que conformément à la façon de définir l'inclusion que nous avons choisie, la valeur du degré de satisfiabilité $\texttt{Deg}(v, v)$ n'est égale à 1 que dans le cas où $\forall x \in X,\ \mu_{v\cap v}(x) = \mu_v(x)$.

Par exemple, avec la t-norme produit, $\forall x \in X,\ \mu_{v\cap v}(x) = \mu_v^2(x)$ qui n'est égal à $\mu_v(x)$ que quand v est un sous-ensemble ordinaire de X ($\mu_v(x) = 0$ ou $\mu_v(x) = 1$ pour tout x).

Notons que, dans le cas de la t-norme *produit*, l'inclusion $w \subseteq v$ correspond à l'égalité pour tout $x \in X : \mu_{v\cap w}(x) = \mu_w(x)$, soit $\mu_v(x)\mu_w(x) = \mu_w(x)$, c'est-à-dire que le support de w doit être inclus dans le noyau de v.

La mesure de satisfiabilité de l'équation 2.18 a été introduite sous le nom de relation floue d'inclusion [14] par (Sanchez, 1979). En analyse fonctionnelle, elle est aussi citée par (Goffman et Pedrick, 1965) comme un cas particulier de mesure. Leur définition de *mesure relative* est très proche de celle déjà présentée :

Définition 3 (Mesure relative) *Soit $\gamma \subseteq [0, 1]$ un espace mesurable, Γ, l'ensemble de tous les sous-ensembles de γ et $\mathcal{M}$, une mesure sur Γ. Pour tout intervalle $I \subseteq [0, 1]$, la* mesure relative *de γ dans I est définie par :*

$$\frac{\mathcal{M}(\gamma \cap I)}{\mathcal{M}(I)}$$

D'autres mesures de satisfiabilité

Il existe de nombreuses autres mesures de satisfiabilité, chacune possédant des propriétés la différenciant des autres, mais toutes possèdent les trois propriétés basiques que l'on requiert.

Toute l'étude qui est faite dans cette thèse est basée sur la mesure de satisfiabilité 2.18. Nous pourrons remarquer que le choix de la forme intégrale implique un choix judicieux pour les autres opérateurs implicites de la théorie des sous-ensembles flous (intersection, produit cartésien...).

Dans le cas où une autre mesure serait choisie pour mesurer le degré de satisfiabilité, il faudrait alors reproduire la même étude pour vérifier le même ensemble de propriétés que celles que nous montrons dans le cas de la mesure intégrale (continuité, stabilité...).

2.5.2 Mesurer la similarité entre une observation et une prémisse

Mesurer la similarité entre une observation W et une prémisse V d'une règle revient à mesurer la similarité entre chaque composant de la prémisse, représentant des tests sur des valeurs d'attributs, et les valeurs correspondantes de l'observation. Ainsi, la ressemblance pour chaque caractérisation w_{l_i} de l'observation à la prémisse v_{l_i} de la règle r est mesurée : $\texttt{Deg}(w_{l_i}, v_{l_i})$. En utilisant les notations introduites dans la partie 2.4.2, on peut dire que ce degré rend compte de la force avec laquelle $A_{l_i} = w_{l_i}$ coïncide avec la partie de la prémisse $A_{l_i} = v_{l_i}$. Par la suite, pour obtenir un degré de satisfaction

14. La terminologie entre (Sanchez, 1979) et (Rifqi, 1996) est antinomique mais on démontre que la relation floue d'inclusion est bien une mesure de satisfiabilité.

global à la prémisse de la règle, il faut agréger tous les degrés de satisfiabilité intervenant dans la prémisse.

Les degrés de toutes les prémisses d'une règle sont alors combinés pour obtenir un *degré global* suivant cette règle. Ce degré global $\mathtt{Deg}(W, V)$ est calculé en agrégeant tous les degrés $\mathtt{Deg}(w_{l_i}, v_{l_i})$ calculés :

$$\mathtt{Deg}(W, V) = \top_{i\,=\,1\ldots p}\,\mathtt{Deg}(w_{l_i}, v_{l_i}) \tag{2.19}$$

Cette agrégation s'effectue à l'aide d'une norme triangulaire $\top$ afin de transcrire le lien conjonctif existant entre les prémisses.

Selon la mesure de satisfiabilité choisie pour définir le degré de satisfiabilité, un choix judicieux des opérateurs d'intersection et de produit cartésien de sous-ensembles flous dans la définition du degré global peut permettre de conserver l'égalité :

$$\top_{i\,=\,1\ldots p}\,\mathtt{Deg}(w_{l_i}, v_{l_i}) = \mathtt{Deg}((w_{l_1}, \ldots, w_{l_p}), (v_{l_1}, \ldots, v_{l_p}))$$

La conservation de cette égalité est un moyen de garantir la correspondance entre le processus de ressemblance ainsi formalisé et le processus cognitif réellement mis en œuvre : la similarité entre deux objets complexes est fonction des similarités entre chacune de leurs parties. On pourrait cependant utiliser un opérateur d'agrégation différent d'une t-norme correspondant à une représentation différente de cette fonction.

Par exemple, pour la définition du degré de satisfiabilité présenté plus haut, si l'intersection et le produit cartésien de sous-ensembles flous sont définis grâce à la t-norme produit, on vérifie que l'on a bien [15] :

$$\prod_{i=1\ldots p} \mathtt{Deg}(w_i, v_i) = \mathtt{Deg}((w_1, \ldots, w_p), (v_1, \ldots, v_p)) \tag{2.20}$$

En effet, avec le degré de satisfiabilité choisi (équation 2.18), on a :

$$\mathtt{Deg}((w_1, \ldots, w_p), (v_1, \ldots, v_p)) = \frac{\displaystyle\int_X \mu_{((w_1,\ldots,w_p)\cap(v_1,\ldots,v_p))}dx}{\displaystyle\int_X \mu_{(w_1,\ldots,w_p)}dx} \tag{2.21}$$

avec $X = X_1 \times \ldots \times X_p$, où X_i est l'univers de valeurs de l'attribut A_i et en supposant que $\int_X \mu_{(w_1,\ldots,w_p)}dx \neq \emptyset$.

La fonction d'appartenance $\mu_{(w_1,\ldots,w_p)}$, définie sur X, produit cartésien des X_i, se déduit des fonctions d'appartenance μ_{w_i} à l'aide d'une norme triangulaire :

$$\forall x = (x_1, \ldots, x_p) \in X,\ \ \mu_{(w_1,\ldots,w_p)}(x) = \mathop{\top}_{i\,=\,1\ldots p} \mu_{w_i}(x_i)$$

Ici, il est à remarquer que, dans le cas où $\int_X \mu_{(w_1,\ldots,w_p)}dx = 0$, cela revient à avoir $w_i(x_i) = 0$ pour au moins une valeur d'indice i, et donc, pour ce i, on a $\mathtt{Deg}(w_i, v_i) = 0$. Ainsi, dans ce cas, l'égalité 2.20 est immédiate :

$$\prod_{i=1\ldots p} \mathtt{Deg}(w_i, v_i) = \mathtt{Deg}((w_1, \ldots, w_p), (v_1, \ldots, v_p)) = 0$$

15. Afin d'alléger les notations de la partie qui suit, la notation indicée l_i est abrégée en indiçant simplement par i. Ainsi, par exemple, on notera w_{l_i} plus simplement par w_i.

La fonction d'appartenance $\mu_{((w_1,...,w_p)\cap(v_1,...,v_p))}$ se définit à partir des fonctions d'appartenance $\mu_{(w_1,...,w_p)}$ et $\mu_{(v_1,...,v_p)}$ à l'aide d'une norme triangulaire utilisée pour réaliser l'intersection de deux sous-ensembles flous, généralement identique à celle utilisées pour construire ce produit cartésien ci-dessus :

$$\mu_{((w_1,...,w_p)\cap(v_1,...,v_p))} = \top(\mu_{(w_1,...,w_p)}, \mu_{(v_1,...,v_p)})$$

Dans le cas où le produit classique est la t-norme utilisée pour réaliser l'intersection de sous ensembles flous et le produit cartésien on a donc :

$$\mu_{((w_1,...,w_p)\cap(v_1,...,v_p))} = \mu_{(w_1,...,w_p)} \cdot \mu_{(v_1,...,v_p)}$$

$$\forall x = (x_1, ..., x_p) \in X, \;\; \mu_{(w_1,...,w_p)}(x) = \mu_{w_1}(x_1)...\mu_{w_p}(x_p)$$

et par conséquent, $\forall x = (x_1, ..., x_p) \in X$,

$$\mu_{((w_1,...,w_p)\cap(v_1,...,v_p))}(x) = \mu_{w_1}(x_1)...\mu_{w_p}(x_p) \cdot \mu_{v_1}(x_1)...\mu_{v_p}(x_p)$$

soit encore, $\forall x = (x_1, ..., x_p) \in X$,

$$\mu_{((w_1,...,w_p)\cap(v_1,...,v_p))}(x) = \prod_{i=1...p} \mu_{w_i}(x_i) \cdot \mu_{v_i}(x_i)$$

Ainsi, sur l'univers $X = X_1 \times ... \times X_p$, l'équation 2.21 s'écrit :

$$\int_X \mu_{((w_1,...,w_p)\cap(v_1,...,v_p))} dx \;\; = \;\; \int_X (\prod_{i=1...p} \mu_{w_i}(x_i) \cdot \mu_{v_i}(x_i)) dx$$

En supposant que les univers de valeurs X_i sont indépendants les uns des autres, c'est-à-dire que dans ce cas $dx = d(x_1, ..., x_p) = dx_1...dx_p$, on obtient :

$$\int_X \mu_{((w_1,...,w_p)\cap(v_1,...,v_p))} dx \;\; = \;\; \prod_{i=1...p} \int_{X_i} \mu_{w_i}(x_i) \cdot \mu_{v_i}(x_i) dx_i$$

Avec la même hypothèse d'indépendance des univers de valeurs des attributs, on a :

$$\int_X \mu_{(w_1,...,w_p)} dx = \prod_{i=1...p} \int_{X_i} \mu_{w_i}(x_i)$$

Par conséquent, l'égalité 2.20 se retrouve donc bien à partir de l'équation 2.21 par le calcul suivant :

$$\begin{aligned}
\mathtt{Deg}((w_1,...,w_p),(v_1,...,v_p)) &= \frac{\prod_{i=1...p}\int_{X_i}\mu_{w_i}(x_i)\,.\,\mu_{v_i}(x_i)dx_i}{\prod_{i=1...p}\int_{X_i}\mu_{w_i}(x_i)dx_i} \\
&= \prod_{i=1...p}\frac{\int_{X_i}\mu_{w_i}(x_i)\,.\,\mu_{v_i}(x_i)dx_i}{\int_{X_i}\mu_{w_i}(x_i)dx_i} \\
&= \prod_{i=1...p}\frac{\int_{X_i}\mu_{w_i\cap v_i}(x_i)dx_i}{\int_{X_i}\mu_{w_i}(x_i)dx_i} \\
&= \prod_{i=1...p}\mathtt{Deg}(w_i,v_i)
\end{aligned}$$

2.5.3 Classification floue d'une observation

Comme[16] il a été précisé dans la partie 2.4.2, chaque chemin dans l'arbre produit une règle r ayant la forme suivante :

$$\mathtt{si}\ A_{l_1}=v_{l_1}\ \mathtt{et}\ A_{l_2}=v_{l_2}\ \mathtt{et}\ \ldots\mathtt{et}\ A_{l_p}=v_{l_p}\ \mathtt{alors}\ C=c_k$$

Mais dans le cadre flou, il est possible que la conclusion attachée à cette règle soit tout l'ensemble des classes $\{c_1,...,c_K\}$, chaque classe c_j étant pondérée par un degré déduit des fonctions d'appartenance. La traduction d'un chemin de l'arbre en règle s'écrit donc alors :

$$\begin{gathered}
\mathtt{si}\ A_{l_1}=v_{l_1}\ \mathtt{et}\ A_{l_2}=v_{l_2}\ \mathtt{et}\ \ldots\mathtt{et}\ A_{l_p}=v_{l_p}\ \mathtt{alors}\ C=c_1\ \mathtt{avec\ le\ degr\acute{e}} \\
P^*(c_1|(v_{l_1},v_{l_2},...v_{l_p}))\ \mathtt{et}\ C=c_2\ \mathtt{avec\ le\ degr\acute{e}}\ P^*(c_2|(v_{l_1},v_{l_2},...v_{l_p}))\ldots \\
\mathtt{et}\ C=c_K\ \mathtt{avec\ le\ degr\acute{e}}\ P^*(c_K|(v_{l_1},v_{l_2},...v_{l_p}))
\end{gathered}$$

La *probabilité conditionnelle* (Zadeh, 1968) $P^*(c_j|(v_{l_1},v_{l_2},...v_{l_p}))$ de la classe c_j, pour la règle r, pondère alors la classe pour la règle. Cela représente donc la probabilité conditionnelle pour un exemple de la base d'apprentissage de posséder cette classe sachant qu'il vérifie tous les tests sur les attributs de la prémisse. Cette pondération ne joue aucun rôle dans la classification classique par les méthode usuelles. Dans ce cas, chaque feuille n'est libellée que par une seule classe c_k, celle qui est majoritaire. Par contre, elle est prise en compte pour la classification floue. Elle est aussi prise en compte dans le cas particulier des arbres probabilistes (Quinlan, 1990). Dans cette amélioration des arbres de décision classiques, chaque feuille est associée à une probabilité conditionnelle issue de l'arrêt de l'algorithme sur des feuilles imparfaites, c'est-à-dire dont les exemples appartenant à l'ensemble qui a servi à la construire, ne possédaient pas la même classe.

16. À partir d'ici, les notations utilisées, notamment pour les indices, sont à nouveau celles présentées dans la partie 2.4.2.

Finalement, l'observation est associée à la classe c_j, suivant la règle r, avec un *degré final* de satisfiabilité. Ce degré final correspond au degré global donné par l'équation 2.19, pondéré par la probabilité conditionnelle de la classe c_j associée à la règle r :

$$\mathtt{Fdeg}_r(c_j) = \top_{i=1\ldots p}\mathtt{Deg}(w_{l_i}, v_{l_i}).P^*(c_j|(v_{l_1}, v_{l_2}, ...v_{l_p}))$$

Ce degré mesure à quel point "c_j doit être la classe de e d'après la règle r".

Les composants d'un cas à classer peuvent avoir des degrés de satisfiabilité non nuls pour plusieurs modalités d'apprentissage d'un même attribut, par conséquent un exemple peut déclencher plusieurs règles et produire ainsi plusieurs degrés finals pour différentes classes. Tous ces degrés sont agrégés par l'intermédiaire d'une t-conorme $\perp$ (par exemple le *maximum*) pour obtenir un unique degré pour chaque classe. Si n_ρ est le nombre de règles issues de l'arbre de décision, on a :

$$\mathtt{Fdeg}(c_j) = \perp_{r=1\ldots n_\rho}\mathtt{Fdeg}_r(c_j)$$

Cela représente le degré avec lequel l'exemple e peut être associé à la classe c_j conformément à l'arbre de décision. Le cas à classer e est donc associé à la classe c_e correspondant à la classe obtenant le plus haut de tous les degrés ainsi calculés :

$$\mathtt{FDeg}(c_e) = \max_{j=1\ldots K}\mathtt{FDeg}(c_j)$$

Comme résultat de la classification du cas e, il est aussi possible de garder toutes les classes associées aux degrés calculés afin de posséder une nuance dans la prise de décision, ou de pouvoir utiliser ces degrés dans un système multi-classifieurs.

Comme pour la méthode classique, il est maintenant intéressant de positionner cette méthode de classification floue, vis-à-vis de la méthode du modus ponens généralisé qui est le système d'inférence classique d'un système à base de règles floues.

Le modus ponens généralisé en présence de données floues

Le *modus ponens généralisé* (Zadeh, 1976) est une extension du modus ponens au raisonnement avec des données floues :

$$\begin{array}{lccc} \text{Règle :} & P & \Longrightarrow & C \\ \text{Observation :} & P' & & \\ \hline \text{Déduction :} & & & C' \end{array}$$

Ainsi, connaissant une règle et observant une donnée *proche* de la prémisse de la règle, le modus ponens généralisé permet d'inférer une conclusion *proche* de la conclusion de la règle. La différence apparaît dans le sens où le modus ponens généralisé autorise la déduction d'une conclusion même dans le cas où l'observation P' est légèrement différente de la prémisse P de la règle, ce qui n'est pas le cas dans le modus ponens classique où la prémisse et l'observation doivent être rigoureusement identiques. Avec le modus ponens généralisé, la conclusion C' peut-être légèrement différente de la conclusion C de la règle. Il est à noter que ce mode de déduction, plus souple que le modus ponens classique, semble mieux refléter le mode de réflexion de l'esprit humain. En effet, il est souvent rare que la même cause se reproduise sans que pour autant notre mode de pensée soit pris en défaut de ne pouvoir conclure.

La conclusion C' se déduit de la conclusion C et de la forme de l'observation P' par la formule du modus ponens généralisé :

$$\forall y \in Y, \quad \mu_{C'}(y) = \sup_{x \in X} \top_m(\mu_{P'}(x), f_R(x,y)) \tag{2.22}$$

dans laquelle $\mu_{C'}$ (resp. $\mu_{P'}$) est la fonction d'appartenance associée à C' (resp. P'), Y est l'univers des valeurs sur lequel est définie la fonction d'appartenance de la conclusion, et X est l'univers des valeurs sur lequel est définie la fonction d'appartenance de la prémisse. f_R est l'implication floue qui décrit le lien entre la prémisse et la conclusion. La norme triangulaire $\top_m$ est l'opérateur de modus ponens généralisé (Bouchon-Meunier, 1995).

La différence première avec la méthode précédente, décrite pour la classification floue, réside dans le fait qu'il n'est pas nécessaire, dans le modus ponens généralisé, d'introduire une mesure de satisfiabilité pour mesurer la ressemblance entre l'observation et la prémisse de la règle. Cette ressemblance sera implicitement prise en compte par les opérateurs qui serviront pour réaliser l'inférence : l'opérateur de modus ponens généralisé et l'implication floue.

Par exemple, si la t-norme probabiliste (le produit) est choisie pour réaliser l'implication, dans un cadre de raisonnement de type Mamdani, on a alors :

$$f_R(x,y) = \mu_P(x) \,.\, \mu_C(y)$$

et si cette t-norme est aussi choisie comme opérateur de modus ponens généralisé :

$$\top_m(\mu_{P'}(x), f_R(x,y)) = \mu_{P'}(x) \,.\, f_R(x,y)$$

il apparaît alors que, dans ce cas, l'équation 2.22 s'écrit :

$$\forall y \in Y, \quad \mu_{C'}(y) = \sup_{x \in X} (\mu_{P'}(x) \,.\, \mu_P(x) \,.\, \mu_C(y))$$

ce qui revient donc à avoir :

$$\forall y \in Y, \quad \mu_{C'}(y) = \mu_C(y) \,.\, \sup_{x \in X} (\mu_{P'}(x) \,.\, \mu_P(x))$$

Il est alors intéressant de remarquer que, si $\mu_{P'}$ et μ_P sont normalisées sur leur univers de définition X et si le *produit* est aussi la t-norme utilisée pour l'intersection, la mesure $\sigma(P', P)$ définie par

$$\sigma(P', P) = \sup_{x \in X} (\mu_{P'}(x) \,.\, \mu_P(x))$$

est une mesure de satisfiabilité. En effet, on montre alors que :

– si $P' \subseteq P$ alors $\exists x \in X, \quad \mu_{P'}(x) = \mu_P(x) = 1$ soit $\sigma(P', P) = 1$

– si $P' \cap P = \emptyset$ alors $\forall x \in X,\ \ \mu_{P'}(x) \,.\, \mu_P(x) = 0$ soit $\sigma(P', P) = 0$

– si $P' \cap P$ croît, alors cela signifie que $\sup_{x\in X} \top(\mu_{P'}(x), \mu_P(x))$ augmente, pour un opérateur d'intersection $\top$. Dans notre cas, $\top$ est le produit et l'expression de σ est retrouvée, ainsi $\sigma(P', P)$ augmente donc avec $P' \cap P$.

Par conséquent, $\sigma(P', P)$ peut être assimilé au degré global $\texttt{Fdeg}_r(C)$ obtenu par la règle r pour la classe C et ainsi, avec cet exemple, il est possible de faire une analogie entre le calcul du modus ponens généralisé et la classification floue présentée précédemment. Il est aussi possible d'utiliser beaucoup d'autres formes de modus ponens généralisé pour effectuer la classification floue, comme par exemple la méthode de filtrage flou introduite par (Mo, 1990), (Dubois et al., 1988; Dubois et al., 1990; Dubois et al., 1991).

2.5.4 Stabilité comparée des méthodes classique et floue de classification

L'apport principal de la méthode de classification floue proposée est de pouvoir traiter des données floues, autant en termes de nœud dans l'arbre de décision qu'en termes de valeurs floues dans l'observation à classer. C'est un aspect important de la phase de généralisation car, dans notre monde réel, les données et mesures sont souvent imprécises et les apports de la théorie des sous-ensembles flous sont donc une aide très précieuse pour les traiter efficacement.

De plus, la classification floue permet de conserver une certaine continuité dans la décision quand la donnée observée varie légèrement. La classification floue est plus stable que la méthode classique. Les valeurs critiques pour ce type de stabilité apparaissent pour les valeurs numériques ou floues.

Altération d'une observation dans le cas classique

Dans le cas numérique et pour la classification classique, près des bornes des tests apparaissant dans les nœuds de l'arbre, une sensible variation dans la valeur d'entrée peut modifier, du tout au tout, la classe obtenue en sortie. En effet, comme il a été précisé dans la partie 2.4.3, l'inférence de la classe est réalisée par l'application du modus ponens, c'est-à-dire que la classe inférée est celle qui libelle la règle dont la prémisse concorde exactement avec l'observation. Si une des valeurs de l'observation change, même légèrement, la prémisse peut ne plus s'accorder avec elle, ce qui provoque l'activation d'une règle complètement différente. Ainsi, pour une variation légère de l'entrée, deux règles différentes s'activent sans qu'il ne puisse y avoir de rapport cohérent dans leur conclusion.

Par exemple, avec l'arbre de la figure 2.1 et les règles qu'il induit, pour l'observation `[Pièce = Tour, Temps Restant = 59s, Pions = peu]`, la règle `si Pièce = Tour et Temps Restant <60s alors Perdu` s'active, ce qui donne la classe `Perdu` pour cette observation. Maintenant, si la valeur du `Temps Restant` varie, la nouvelle observation à classer, par exemple `[Pièce = Tour, Temps Restant = 61s, Pions = peu]`, n'active plus la même règle mais la règle `si Pièce = Tour et Temps Restant`

$\geq$`60s alors Gagné`, ce qui donne la classe `Gagné` pour une différence qui peut paraître légère d'une des valeurs de l'observation.

Certes, dans certains cas, il existe un grand pouvoir de discrimination pour une valeur d'un attribut. Cette valeur peut agir comme un seuil inéluctable, prépondérant dans la détermination de la classe. Cependant, dans la majorité des cas, en apprentissage inductif, les valeurs des seuils présents dans les nœuds de l'arbre apparaissent à la suite d'une discrétisation des attributs numériques et sont, par conséquent, notoirement imprécis et dépendants de l'ensemble d'apprentissage.

Altération d'une observation dans le cas flou

Par contre, avec la classification floue, toujours dans le seul cas numérique, mais aussi en présence de données floues, à la fois dans les tests et dans l'observation, il se produit une continuité dans la décision. Cette continuité résulte du degré de satisfiabilité qui est utilisé pour rendre compte de la façon dont l'observation satisfait la prémisse. À la différence du cas classique, le degré de satisfiabilité n'est plus binaire à valeurs dans $\{0, 1\}$ mais il est graduel à valeurs dans $[0, 1]$. Par conséquent, les changements d'une règle à l'autre s'effectuent d'une manière plus progressive quand l'observation se modifie légèrement. Cette progression dépend bien entendu du degré de satisfiabilité et de l'opérateur d'intersection de sous-ensembles flous choisis.

Le moyen pour se rendre compte de la façon dont évolue le degré de satisfiabilité est d'étudier la continuité de la fonction `Deg`. Une valeur observée $\tilde{w}$, proche de la valeur w, possède comme degré de satisfaction $\mathtt{Deg}(\tilde{w}, v)$. Considérer la valeur $\tilde{w}$ comme légèrement différente de la valeur w, se transcrit en utilisant une fonction d'erreur ϵ : $X \longrightarrow [-1, 1]$: $\forall x \in X,\ \ \mu_{\tilde{w}}(x) = \mu_w(x) + \epsilon(x)$.

La figure 2.5 montre graphiquement la forme que peut posséder une telle fonction d'erreur.

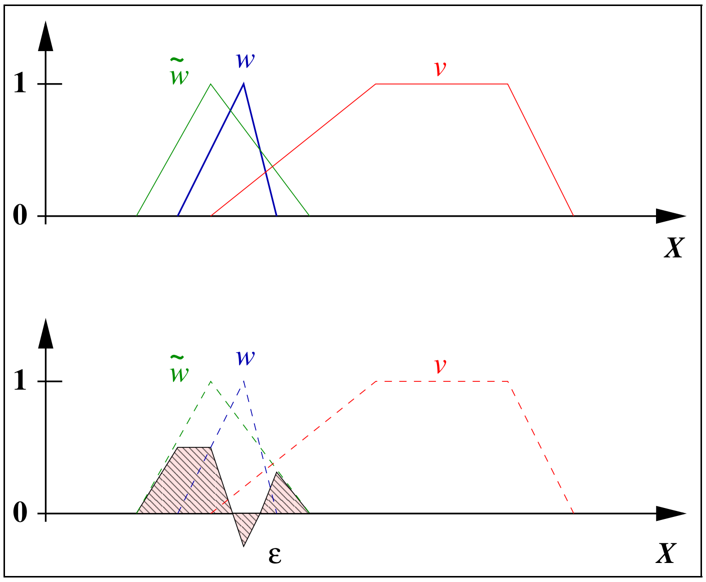


Figure 2.5 – *Exemple de fonction modificatrice*

Continuité de la mesure de satisfiabilité

Il faut donc s'intéresser à la continuité de $\mathtt{Deg}$ en w, soit regarder comment évolue $|\mathtt{Deg}(\tilde{w}, v) - \mathtt{Deg}(w, v)|$ en fonction de $|\tilde{w} - w|$. Par exemple, considérons le cas où le degré de satisfiabilité est celui donné précédemment, où X est l'univers de définition de w et v :

$$\mathtt{Deg}(w, v) = \frac{\int_X \mu_{w \cap v}\, dx}{\int_X \mu_w\, dx} \quad \text{si} \quad \int_X \mu_w\, dx \neq 0$$

Avec ce degré de satisfaction, le produit doit être utilisé comme norme triangulaire pour réaliser l'intersection des sous-ensembles flous. Il est possible d'exprimer $\mathtt{Deg}(w, v)$ en fonction de $\mathtt{Deg}(\tilde{w}, v)$:

$$\begin{aligned}
\mathtt{Deg}(w, v) &= \frac{\int_X \mu_{w \cap v}\, dx}{\int_X \mu_w\, dx} \\
&= \frac{\int_X \mu_w \mu_v\, dx}{\int_X \mu_w\, dx} \\
&= \frac{\int_X (\mu_{\tilde{w}} - \epsilon)\mu_v\, dx}{\int_X \mu_w\, dx} \\
&= \frac{\int_X \mu_{\tilde{w}} \mu_v\, dx}{\int_X \mu_w\, dx} - \frac{\int_X \epsilon \mu_v\, dx}{\int_X \mu_w\, dx} \\
&= \left(\frac{\int_X \mu_{\tilde{w}} \mu_v\, dx}{\int_X \tilde{w}\, dx} \frac{\int_X \mu_{\tilde{w}}\, dx}{\int_X \mu_w\, dx} \right) - \frac{\int_X \epsilon \mu_v\, dx}{\int_X \mu_w\, dx} \\
&= \left(\mathtt{Deg}(\tilde{w}, v) \frac{\int_X \mu_{\tilde{w}}\, dx}{\int_X \mu_w\, dx} \right) - \frac{\int_X \epsilon \mu_v\, dx}{\int_X \mu_w\, dx}
\end{aligned}$$

Ce qui permet de reformuler $\mathtt{Deg}(\tilde{w}, v) - \mathtt{Deg}(w, v)$ ainsi :

$$\begin{aligned}
\mathtt{Deg}(\tilde{w}, v) - \mathtt{Deg}(w, v) &= \mathtt{Deg}(\tilde{w}, v) - \left(\mathtt{Deg}(\tilde{w}, v) \frac{\int_X \mu_{\tilde{w}}\, dx}{\int_X \mu_w\, dx} \right) + \frac{\int_X \epsilon \mu_v\, dx}{\int_X \mu_w\, dx} \\
&= \mathtt{Deg}(\tilde{w}, v) \left(1 - \frac{\int_X \mu_{\tilde{w}}\, dx}{\int_X \mu_w\, dx} \right) + \frac{\int_X \epsilon \mu_v\, dx}{\int_X \mu_w\, dx}
\end{aligned}$$

$$
\begin{aligned}
&= \mathtt{Deg}(\tilde{w}, v) \left(\frac{\displaystyle\int_X \mu_w - \mu_{\tilde{w}}\, dx}{\displaystyle\int_X \mu_w\, dx} \right) + \frac{\displaystyle\int_X \epsilon \mu_v\, dx}{\displaystyle\int_X \mu_w\, dx} \\
&= -\mathtt{Deg}(\tilde{w}, v) \frac{\displaystyle\int_X \epsilon\, dx}{\displaystyle\int_X \mu_w\, dx} + \frac{\displaystyle\int_X \epsilon \mu_v\, dx}{\displaystyle\int_X \mu_w\, dx}
\end{aligned}
$$

soit en valeur absolue :

$$
|\mathtt{Deg}(\tilde{w}, v) - \mathtt{Deg}(w, v)| = \left| -\mathtt{Deg}(\tilde{w}, v) \frac{\displaystyle\int_X \epsilon\, dx}{\displaystyle\int_X \mu_w\, dx} + \frac{\displaystyle\int_X \epsilon \mu_v\, dx}{\displaystyle\int_X \mu_w\, dx} \right|
$$

Par conséquent, $|\mathtt{Deg}(\tilde{w}, v) - \mathtt{Deg}(w, v)|$ peut se majorer en utilisant l'inégalité triangulaire :

$$
\begin{aligned}
|\mathtt{Deg}(\tilde{w}, v) - \mathtt{Deg}(w, v)| &\leq \mathtt{Deg}(\tilde{w}, v) \frac{\left|\displaystyle\int_X \epsilon\, dx\right|}{\displaystyle\int_X \mu_w\, dx} + \frac{\left|\displaystyle\int_X \epsilon \mu_v\, dx\right|}{\displaystyle\int_X \mu_w\, dx} \\
&\leq \frac{\displaystyle\int_X |\epsilon|\, dx}{\displaystyle\int_X \mu_w\, dx} + \frac{\displaystyle\int_X |\epsilon| \mu_v\, dx}{\displaystyle\int_X \mu_w\, dx} \qquad (2.23)
\end{aligned}
$$

S'il existe un réel positif α tel que $\forall x \in X,\ |\epsilon(x)| \leq \alpha$ alors :

$$
\int_X |\epsilon(x)|\, dx \leq \alpha \int_X dx
$$

Ainsi donc, en notant les intégrales, supposées finies,

$$
M(w) = \int_X \mu_w\, dx, \qquad M(v) = \int_X \mu_v\, dx \qquad \text{et} \quad M = \int_X dx,
$$

l'inégalité 2.23 permet d'écrire :

$$
|\mathtt{Deg}(\tilde{w}, v) - \mathtt{Deg}(w, v)| \leq \frac{\alpha M + \alpha M(v)}{M(w)}
$$

Par définition, $\forall x \in X$, $\mu_v(x) \leq 1$ et donc $M(v) \leq M$, on obtient finalement :

$$
|\mathtt{Deg}(\tilde{w}, v) - \mathtt{Deg}(w, v)| \leq \frac{2\alpha M}{M(w)}
$$

On peut donc en déduire la continuité de Deg en w :

$$
\forall \sigma > 0,\ \exists \alpha = \frac{\sigma M(w)}{2M} \ \text{ tel que}
$$

$$\forall x \in X,\ |\epsilon(x)| \leq \alpha \Rightarrow |\mathtt{Deg}(\tilde{w}, v) - \mathtt{Deg}(w, v)| \leq \sigma$$

soit encore :

$$\forall \sigma > 0,\ \exists \alpha = \frac{\sigma M(w)}{2M} \text{ tel que}$$

$$\forall x \in X,\ |\mu_{\tilde{w}}(x) - \mu_w(x)| \leq \alpha \Rightarrow |\mathtt{Deg}(\tilde{w}, v) - \mathtt{Deg}(w, v)| \leq \sigma$$

On démontre ainsi la continuité du degré de satisfiabilité. La variation du degré de satisfiabilité est continue quand une observation floue est légèrement altérée.

Altération floue d'une valeur précise

Intéressons nous maintenant au cas où la valeur initiale observée w est une valeur précise et qu'une valeur $\tilde{w}$ très proche de w est ensuite observée. Étant donné que v est une valeur floue, le degré de satisfiabilité de w relatif à v est donné par $\mathtt{Deg}(w, v) = \mu_v(w)$ car w est une valeur précise.

Si $\tilde{w}$ est aussi une valeur précise, il est clair que la variation de degré sera continue, car dans ce cas $\mathtt{Deg}(\tilde{w}, v) = \mu_v(\tilde{w})$ et μ_v est supposée continue.

Si $\tilde{w}$ est une valeur floue, il faut vérifier alors que $\mathtt{Deg}(\tilde{w}, v)$ qui est défini par l'équation 2.18 converge bien vers $\mathtt{Deg}(w, v) = \mu_v(\tilde{w})$.

Pour les calculs, on considère que $\tilde{w}$ est un nombre flou de valeur modale w et de support $[w - \beta, w + \beta]$, et on considère que l'on se situe sur la pente de v comme il l'est représenté dans la figure 2.6. Pour $x \in [w - \beta, w]$, on a $\mu_{\tilde{w}}(x) = \frac{x+\beta-w}{\beta}$ et pour $x \in [w, w + \beta]$, on a $\mu_{\tilde{w}}(x) = \frac{-x+\beta+w}{\beta}$. On pose aussi que $\forall x \in [w - \beta, w + \beta]$, μ_v est une droite d'équation $\mu_v(x) = ax + b$.

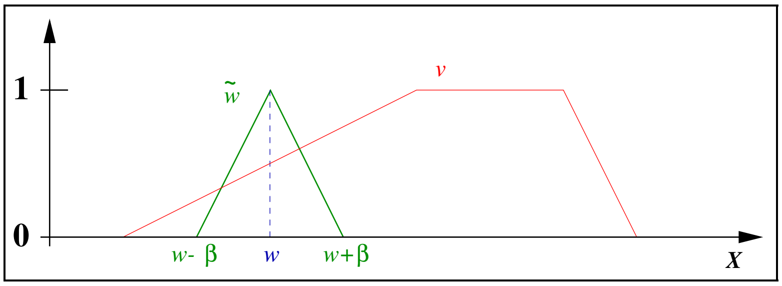


Figure 2.6 – *Exemple de variation (cas $(w - \beta) > -\frac{b}{a}$)*

Pour assurer la continuité dans ce cas là, il faut vérifier que

$$\lim_{\beta \to 0} \mathtt{Deg}(\tilde{w}, v) = \mathtt{Deg}(w, v)$$

$\diamond$ Considérons d'abord le cas où $(w - \beta) > -\frac{b}{a}$. On a :

$$\mathtt{Deg}(\tilde{w}, v) = \frac{\displaystyle\int_X \mu_{\tilde{w}\cap v}\, dx}{\displaystyle\int_X \mu_{\tilde{w}}\, dx}$$

avec :

$$\int_X \mu_{\tilde{w}}\, dx = \beta$$

en calculant cette intégrale comme l'aire du nombre flou triangulaire $\tilde{w}$.

Ainsi, en utilisant la t-norme produit pour réaliser l'intersection, on obtient :

$$\begin{aligned} \forall x \in [w-\beta, w],\ \mu_{\tilde{w}\cap v}(x) = \frac{x+\beta-w}{\beta}(ax+b) \\ \forall x \in [w, w+\beta], \\ \mu_{\tilde{w}\cap v}(x) = \frac{-x+\beta+w}{\beta}(ax+b) \end{aligned} \tag{2.24}$$

ce qui donne alors :

$$\begin{aligned}
\int_X \mu_{\tilde{w}\cap v}\, dx &= \int_{w-\beta}^{w} \mu_{\tilde{w}\cap v}(x)\, dx + \int_{w}^{w+\beta} \mu_{\tilde{w}\cap v}(x)\, dx \\
&= \int_{w-\beta}^{w} \frac{x+\beta-w}{\beta}(ax+b)\, dx + \int_{w}^{w+\beta} \frac{-x+\beta+w}{\beta}(ax+b)\, dx \\
&= \int_{w}^{w+\beta} \frac{x-w}{\beta}(a(x-\beta)+b)\, dx + \int_{w}^{w+\beta} \frac{-x+\beta+w}{\beta}(ax+b)\, dx \\
&= \int_{w}^{w+\beta} (\frac{x-w}{\beta}(a(x-\beta)+b) + \frac{-x+\beta+w}{\beta}(ax+b))\, dx \\
&= \int_{w}^{w+\beta} ((ax+b)(\frac{x-w}{\beta} + \frac{-x+\beta+w}{\beta}) - a\beta\frac{x-w}{\beta})\, dx \\
&= \int_{w}^{w+\beta} (ax+b-ax+aw)\, dx \\
&= \int_{w}^{w+\beta} (aw+b)\, dx \\
&= [(aw+b)x]_{w}^{w+\beta} \\
&= (aw+b)(w+\beta) - (aw+b)w \\
&= (aw+b)\beta
\end{aligned}$$

soit donc :

$$\int_X \mu_{\tilde{w}\cap v}\, dx \quad = \quad \mu_v(w)\beta$$

et on obtient alors :

$$\mathtt{Deg}(\tilde{w}, v) \quad = \quad \frac{\mu_v(w)\beta}{\beta} = \mu_v(w) = \mathtt{Deg}(w, v)$$

$\diamond$ Considérons maintenant le cas où $w = -\frac{b}{a}$.

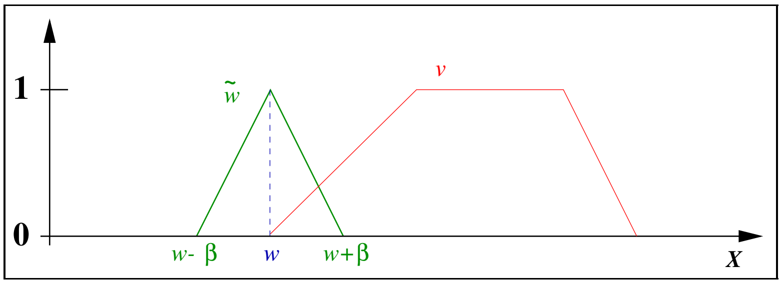


Figure 2.7 – *Exemple de variation (cas $w = -\frac{b}{a}$)*

Dans ce cas, $\texttt{Deg}(w, v) = \mu_v(w) = \mu_v(-\frac{b}{a}) = 0$. Toujours en utilisant la t-norme produit pour réaliser l'intersection, on a :

$$\forall x \in [-\frac{b}{a} - \beta, -\frac{b}{a}],\ \mu_{\tilde{w}\cap v}(x) = 0$$

$$\text{et}\quad \forall x \in [-\frac{b}{a}, -\frac{b}{a} + \beta],\ \mu_{\tilde{w}\cap v}(x) = \frac{-x + \beta + -\frac{b}{a}}{\beta}(ax + b)$$

ce qui donne alors :

$$\begin{aligned}
\int_X \mu_{\tilde{w}\cap v}\,dx &= \int_{-\frac{b}{a}}^{-\frac{b}{a}+\beta} \mu_{\tilde{w}\cap v}(x)\,dx \\
&= \int_0^{\beta} \mu_{\tilde{w}\cap v}(x - \frac{b}{a})\,dx \\
&= \int_0^{\beta} \frac{-(x - \frac{b}{a}) + \beta - \frac{b}{a}}{\beta}(a(x - \frac{b}{a}) + b)\,dx \\
&= \int_0^{\beta} \frac{-x + \beta}{\beta} ax\,dx \\
&= \frac{1}{\beta}\int_0^{\beta} a\beta x - ax^2\,dx \\
&= \frac{1}{\beta}\left[\frac{a\beta x^2}{2} - \frac{ax^3}{3}\right]_0^{\beta} \\
&= \frac{a\beta^2}{2} - \frac{a\beta^2}{3} \\
&= \frac{a\beta^2}{6}
\end{aligned}$$

Par conséquent, on a finalement :

$$\texttt{Deg}(\tilde{w}, v) = \frac{\frac{a\beta^2}{6}}{\beta} = \frac{a\beta}{6}$$

Et donc

$$\lim_{\beta\to 0} \texttt{Deg}(\tilde{w}, v) = \lim_{\beta\to 0} \frac{a\beta}{6} = 0$$

Pour une modalité floue de référence v donnée, le degré de satisfiabilité de $\tilde{w}$ tend donc bien vers le degré de satisfiabilité de w quand β tend vers 0.
$\diamond$ Dans le cas où $w < -\frac{b}{a}$, le degré de satisfiabilité de w est nul relativement à la modalité v. Une altération de w telle que $w + \beta \leq -\frac{b}{a}$ conservera la même valeur nulle de satisfiabilité. Par contre, si on choisit β suffisamment grand pour que $w + \beta > -\frac{b}{a}$, il faut réitérer les calculs pour trouver que le degré converge bien vers 0. Ceux-ci se font sur les mêmes techniques que les calculs déjà donnés et ne seront donc pas répétés ici.
$\diamond$ De même, dans le cas où $w = -\frac{1-b}{a}$, les calculs des limites se font sur les mêmes techniques et permettent de vérifier que le degré de $\mathtt{Deg}(\tilde{w}, v)$ converge bien vers $\mathtt{Deg}(w, v)$.

Tous les calculs donnés ici concernent le cas où l'on se place sur une des pentes du sous-ensemble flou v. Il serait fastidieux de répéter ici la totalité des calculs pour chaque type de pente possible et ils ne seront donc pas donnés.

Finalement, on montre donc que la fonction F ainsi définie, c'est-à-dire le degré de satisfiabilité relatif à une même valeur de référence, est continue.

Ainsi, avec le degré de satisfiabilité étudié, il paraît donc clair qu'une légère modification de la valeur observée entraînera simplement une modification continue du degré de satisfiabilité correspondant. Par voie de conséquence, la répercussion de cette modification sur les degrés calculés pour cette règle est elle aussi continue, de même que le degré final d'obtenir la classe par l'ensemble des règles de la base[17].

Influence sur la conclusion de la variation de la valeur degré

Après avoir vérifié la continuité de la variation du degré de satisfiabilité quand l'observation varie légèrement, il faut maintenant regarder le changement qu'une variation produit sur la conclusion inférée.

Comme il a été précisé précédemment, les degrés de toutes les prémisses d'une règle sont agrégés pour obtenir un degré global qui a la forme suivante (*cf.* équation 2.19 en utilisant toujours la t-norme produit pour réaliser l'intersection) :

$$\prod_{i=1\ldots,p} \mathtt{Deg}(w_{l_i}, v_{l_i})$$

On note $D(w)$ ce produit. Si l'une des observations varie légèrement, on a vu que dans ce cas, le degré de cette variation varie lui aussi légèrement. Soit w_{l_k} l'observation initiale qui se trouve altérée en $\tilde{w_{l_k}} + \epsilon$. Dans ce cas, grâce à la continuité du degré de satisfiabilité, le degré $\mathtt{Deg}(w_{l_k}, v_{l_k})$ se trouve modifié en $\mathtt{Deg}(w_{l_k}, v_{l_k}) + \varepsilon$, avec $\varepsilon \in \mathbb{R}$ et ε petit. Le degré global $D(\tilde{w})$ est :

$$D(\tilde{w}) = (\mathtt{Deg}(w_{l_k}, v_{l_k}) + \varepsilon) \prod_{i=1\ldots k,k+1,\ldots,p} \mathtt{Deg}(w_{l_i}, v_{l_i})$$

soit

$$D(\tilde{w}) = D(w) + \varepsilon \left(\prod_{i=1\ldots k,k+1,\ldots,p} \mathtt{Deg}(w_{l_i}, v_{l_i}) \right)$$

17. Pour vérifier cela, il suffit de propager la valeur du degré $\mathtt{Deg}(\tilde{w}, v)$ en fonction de $\mathtt{Deg}(w, v)$ dans toutes les équations des degrés obtenus par agrégation, données dans la partie 2.5.3.

C'est-à-dire que :

$$|D(\tilde{w}) - D(w)| = \left| \varepsilon \left(\prod_{i=1\ldots k, k+1, \ldots, p} \mathtt{Deg}(w_{l_i}, v_{l_i}) \right) \right|$$

Et par conséquent :

$$|D(\tilde{w}) - D(w)| \leq |\varepsilon| \quad \text{car } \forall w_{l_i}, v_{l_i},\ \mathtt{Deg}(w_{l_i}, v_{l_i}) \leq 1$$

On peut donc en déduire la continuité de $D(w)$ en w :

$$\forall \sigma > 0,\ \exists \alpha = \sigma \ \text{ tel que } \forall x \in X,\ |\varepsilon| \leq \alpha \Rightarrow |D(\tilde{w}) - D(w)| \leq \sigma$$

Par conséquent, en utilisant la méthode de classification floue, il est possible d'en déduire qu'il n'apparaîtra plus de changement brutal de conclusion comme il en apparaît avec la méthode classique.

2.6 Implémentation : l'application *Salammbô*

La méthode générique de construction d'arbres de décision flous présentée dans la section 2.2 et la méthode de classification floue présentée dans la section 2.5 ont été implémentées pour donner naissance à l'application *Salammbô*[18].

Salammbô est notre système de construction d'arbres de décision flous en présence de données imparfaites. L'arbre construit peut être utilisé pour classer de nouveaux exemples en utilisant les principes de la classification floue présentée.

Salammbô a été implémentée en langage C et en langage Lex et elle représente environ 7500 lignes de code source. La structure de données utilisée ainsi que les fonctions pour implémenter les mesures de discrimination sont données en annexe D.

Salammbô est adaptée à plusieurs contraintes que nous nous sommes fixés :

- Ne posséder aucune connaissance sur les données à traiter afin d'être entièrement autonome et générique. C'est pourquoi, elle utilise un analyseur lexical, implémenté en langage Lex, pour analyser la base d'apprentissage et la transformer dynamiquement dans la structure de données qu'elle peut utiliser.
- Pouvoir être paramétrable :
  - au niveau de la mesure de discrimination utilisée.
  - au niveau de la mesure d'arrêt.
  - au niveau du système d'inférence et en particulier des opérateurs à utiliser pour réaliser le classement de nouveaux exemples.
- Être capable de pouvoir gérer, avec un temps d'exécution raisonnable, de grandes bases d'exemples.

18. Ce nom n'est pas un acronyme.

La partie du programme relative à la mesure de discrimination a été écrite de façon à faciliter l'incorporation de nouvelles mesures (*cf.* Annexe D). Le but de Salammbô est alors de pouvoir tester et comparer les différentes mesures entre elles.

La classification de nouveaux objets s'effectue selon la méthode proposée dans la section 2.5.3, en utilisant différents opérateurs possibles (t-norme, t-conorme, opérateur d'implication). Comme pour les mesures de discrimination, la partie du programme concernant les opérateurs a été conçue afin de faciliter l'introduction de nouveaux opérateurs.

Salammbô est une application générique de construction d'arbres de décision. Elle est conçue avec le souci de pouvoir lui incorporer un grand nombre de mesures de discrimination et un large choix de systèmes d'inférence. Dans un premier temps, cela permet de comparer les différentes options de construction et de classement entres elles.

Les résultats obtenus avec les différentes mesures et différents opérateurs sur diverses bases d'apprentissage sont donnés dans le chapitre 7.

2.7 Conclusion

Dans ce chapitre, les arbres de décision flous ont été présentés.

Différentes méthodes de construction d'arbres de décision flous existent. Pourtant, ces méthodes sont souvent similaires, à la notation utilisée près.

Une formalisation des méthodes de construction d'arbres de décision a été proposée. Cette formalisation permet de mieux discerner les différents niveaux d'études de ces algorithmes. Ensuite, elle autorise l'introduction des éléments de la théorie des sous-ensembles flous de façon spécifique au niveau en question. Il reste encore à formaliser les différents paramètres de l'algorithme de construction d'arbres.

Le processus de construction d'un arbre est unique, la seule chose qui différencie les méthodes réside dans :

- le choix de la mesure de discrimination utilisée pour sélectionner l'attribut qui donnera naissance à un nœud,
- le choix du critère d'arrêt pour stopper le processus de construction est créer une feuille,
- le choix de la stratégie de partitionnement mise en œuvre quand l'attribut est un attribut numérique-symbolique,
- la méthode d'élagage utilisée pour simplifier l'arbre construit.

Nous nous sommes essentiellement focalisés, dans nos travaux, sur le choix de la mesure de discrimination.

Après avoir rappelé les différentes techniques de validation d'une mesure de discrimination, nous avons décidé de proposer une nouvelle technique *constructive* de validation. Cette technique doit permettre de déterminer si les propriétés requises pour une mesure de discrimination pour le cas des attributs numériques-symboliques sont identiques à celles requises dans un cas classique.

Une méthode d'utilisation des arbres de décision flous pour le classement de nouveaux objets a ensuite été présentée. La méthode floue peut être considérée comme une méthode hautement plus robuste que la méthode d'inférence classique. La robustesse des systèmes flous est largement reconnue. Tel est encore le cas dans le domaine des arbres de décision flous, dont la variation de la décision concernant un objet sera continue quand la description de l'objet changera.

Les arbres de décision flous apparaissent alors comme très utiles pour l'apprentissage inductif ou la construction de bases de règles. Les outils de traitement de données numériques-symboliques ou imprécises de la théorie des sous-ensembles flous sont parfaitement adaptés à leur introduction dans ce domaine.

Finalement, l'application Salammbô a été présentée. Salammbô implémente les techniques de construction d'arbres de décision flous en utilisant différentes mesures de discrimination pour sélectionner les attributs. Elle utilise ensuite l'arbre de décision flou construit pour classer de nouveaux objets. Le classement s'effectue en utilisant différentes t-normes pour agréger les résultats. Chaque nouvel objet se voit alors associé à chaque classe à reconnaître, avec un degré de satisfiabilité, ce qui autorise une gradualité dans la décision qui lui sera associée.

Deuxième partie

Améliorer la prise en compte de l'imperfection

Chapitre 3

Présentation

Instruit par la mésaventure des miens, j'avais appris à me méfier des évidences. Lorsque tout le monde s'agglutine autour d'une même opinion, je m'enfuis : la vérité est sûrement ailleurs.

A. Maalouf – *Léon l'africain*

3.1 Introduction

Dans la première partie de cette thèse, une méthode d'apprentissage inductif prenant en considération les données imprécises a été présentée. L'algorithme générique de construction d'arbres de décision flous et la méthode de classification par arbre de décision flous proposés ont donné lieu à l'implémentation de l'application générique Salammbô.

Dans cette seconde partie de notre thèse, nous proposons des améliorations à apporter à cet algorithme. Ces améliorations portent sur trois points principaux :

- doter notre algorithme d'apprentissage d'une méthode de construction automatique de partitions floues afin d'augmenter son autonomie.

- améliorer l'apprentissage dans les domaines où il existe plus de deux classes à apprendre.

- trouver une méthode pour sélectionner et pour construire une bonne mesure de discrimination pour construire un arbre de décision flou.

Ce sont ces améliorations qui sont développées dans cette partie.

Dans ce chapitre, après avoir présenté avec un peu plus de détail les améliorations proposées qui seront décrites complètement dans les chapitres de cette partie, nous donnons et explicitons le système général d'apprentissage inductif par arbres de décision flous que nous avons mis au point.

3.2 Les améliorations proposées

3.2.1 Construire des partitions floues

Un élément important pour la prise en compte de données imprécises est l'existence d'une partition floue sur l'univers des valeurs de l'attribut numérique-symbolique correspondant. Cette partition est généralement donnée par un expert du domaine mais elle ne correspond pas souvent à la réalité du phénomène qui doit être appris. Pour qu'un système d'apprentissage inductif soit entièrement autonome, il est nécessaire de le doter d'une méthode automatique de construction de partitions floues. Il doit être capable de gérer les données numériques ou numériques-symboliques, de la manière qu'il considère être la meilleure.

C'est pourquoi, comme première amélioration des algorithmes d'apprentissage en environnement imparfait, une méthode de construction de partitions floues est proposée. Cette méthode est incorporée dans le système Salammbô pour la rendre entièrement autonome du point de vue du traitement des valeurs numériques.

3.2.2 Faire des forêts d'arbres de décision

Un autre problème pour construire des arbres de décision est inhérent à la mesure de discrimination choisie qui n'est pas toujours adaptée à la prise en compte de domaine où il existe plus de deux classes à reconnaître. Une solution pour gérer le problème des domaines à plus de deux classes est de construire une forêt d'arbres de décision où chaque arbre est dédié à la reconnaissance d'une unique classe contre toutes les autres réunies.

C'est la deuxième amélioration que nous proposons. L'application Tanit que nous présentons, autorise le lancement d'un ensemble de Salammbô, chacune dédiée à l'apprentissage d'une classe unique. Pour classer de nouveaux objets, Tanit récupère les degrés d'appartenance aux classes retournées par les Salammbô et les agrège pour obtenir un degré d'appartenance global à chacune des classes. Le problème de la mesure de discrimination en présence de plusieurs cas est donc contourné, il n'est nécessaire que d'utiliser une bonne mesure de discrimination qui sache reconnaître le pouvoir discriminant d'un attribut que relativement à deux classes.

3.2.3 Une méthode constructive pour caractériser une bonne mesure de discrimination

Comme il a été annoncé dans la première partie de cette thèse, une nouvelle méthode constructive pour caractériser une bonne mesure de discrimination est présentée ici. En plus de permettre la reconnaissance d'une bonne mesure de discrimination, cette méthode permet aussi de construire plus aisément de telles mesures. C'est un outil qui nous paraît intéressant dans le domaine des arbres de décision flous pour arriver à bien reconnaître les avantages d'une mesure par rapport à une autre.

Nous pensons qu'une bonne mesure de discrimination n'est pas forcément une bonne mesure d'information et les propriétés qui lui sont demandées ne sont pas nécessairement les mêmes. À travers cette méthode, la caractérisation des propriétés requises pour une bonne mesure doit s'effectuer en liaison directe avec l'objectif à atteindre avec cette mesure.

3.3 Un système d'apprentissage inductif en environnement imparfait

Finalement, Salammbô et Tanit constituent l'architecture globale de notre système d'apprentissage inductif en présence de données imprécises. Ce système est à même de prendre en compte et d'élucider différents types de problèmes susceptibles de survenir durant l'apprentissage.

Dans un premier niveau, l'application Salammbô est capable de gérer les données imprécises. Dans un second niveau, l'application Tanit autorise une meilleure gestion des domaines où il existe plus de deux classes.

3.3.1 L'application Salammbô pour gérer les données imprécises

La structure de l'application Salammbô est donnée dans la figure 3.1. Salammbô est le système de construction et d'utilisation d'arbres de décision flous présenté dans la première partie de cette thèse. À partir d'une base d'apprentissage qui lui est donnée, Salammbô construit un arbre de décision flou. La mesure de discrimination qu'elle doit utiliser pour comparer les attributs entre eux durant la construction de l'arbre est paramétrable et peut être choisie parmi un ensemble de mesures de discrimination sélectionnées ou construites par la hiérarchie de mesures présentée dans le chapitre 6.

Le cas échéant, Salammbô construit automatiquement une partition floue sur l'univers des valeurs d'un attribut numérique ou numérique-symbolique par la méthode de construction décrite dans le chapitre 4.

Une fois l'arbre de décision flou construit, pour classer de nouveaux objets, Salammbô utilise la méthode de classement présentée au chapitre 2, en se basant sur l'utilisation de différentes t-normes et différents opérateurs de modus ponens généralisé.

Finalement, lors de la classification d'un nouvel objet, Salammbô retourne un degré d'appartenance pour cet objet à chacune des classes qu'elle a appris à reconnaître. La classe de l'objet peut alors être choisie en prenant la classe qui obtient le plus haut degré d'appartenance ou bien toutes les classes et leurs degrés peuvent être conservés ainsi pour obtenir une décision plus graduelle.

3.3.2 En présence de plus de deux classes : Tanit

En présence de plus de deux classes dans la base d'apprentissage, selon la mesure de discrimination utilisée, les résultats d'apprentissage et de classification de Salammbô peuvent s'en trouver dégradés.

C'est pourquoi nous proposons la construction d'une forêt d'arbres de décision flous, chacun ne reconnaissant que deux classes. L'application générique Tanit permet de résoudre les problèmes engendrés par la présence de plus de deux classes dans la base d'apprentissage (figure 3.2). Tanit est une application qui crée et gère une forêt d'arbres

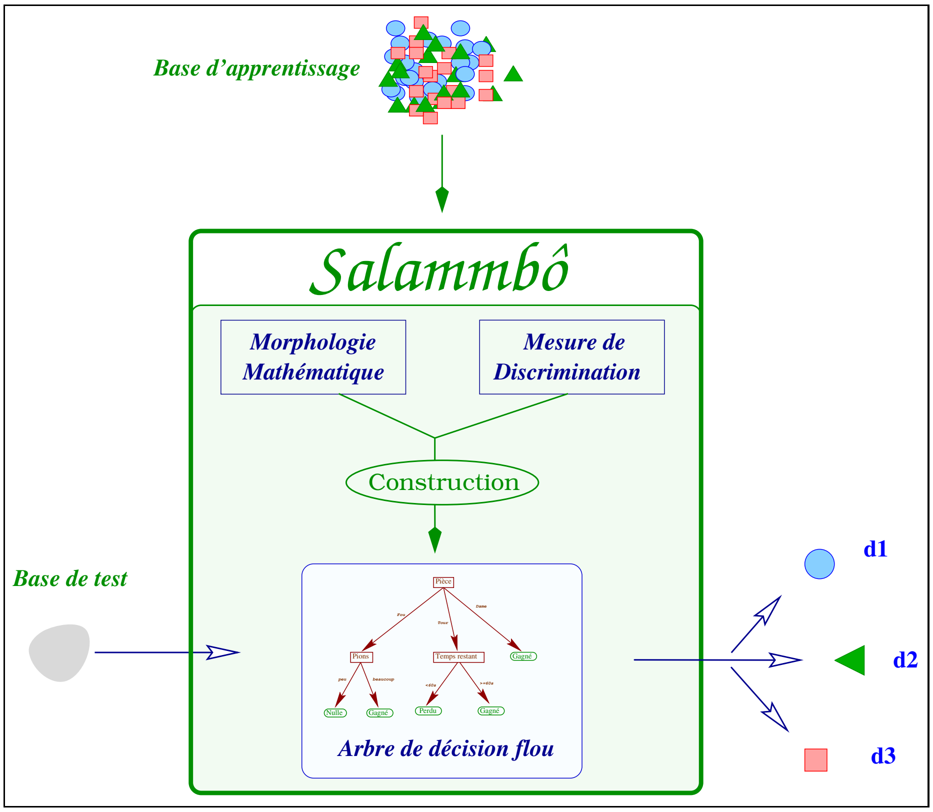


Figure 3.1 – *L'application Salammbô*

de décision flous, elle est décrite en détail dans le chapitre 5. À partir d'une base d'apprentissage comportant plus de deux classes à reconnaître, Tanit lance un ensemble de processus Salammbô, chacune pour construire, à partir de la base d'apprentissage, un arbre de décision flou ne reconnaissant que deux classes.

Ensuite, lors de la phase de classification, Tanit gère cette forêt d'arbres de décision flous en récupérant, pour chaque nouvel objet à classer, le résultat de la classification de chaque Salammbô pour cet objet, et en l'agrégeant par la méthode d'agrégation souhaitée.

3.4 Conclusion

Dans ce chapitre introductif de la partie concernant les améliorations supplémentaires que nous proposons, notre système général d'apprentissage inductif par arbres de décision flous a été présenté.

Ce système étend l'application Salammbô en l'augmentant d'une fonctionnalité qui la rend encore plus autonome, à savoir la capacité de construire automatiquement des partitions floues à partir d'un ensemble de valeurs numériques-symboliques. De plus, Tanit, une nouvelle application, a été proposée pour améliorer l'apprentissage dans des domaines où il existe plus de deux classes à apprendre, domaines où les résultats

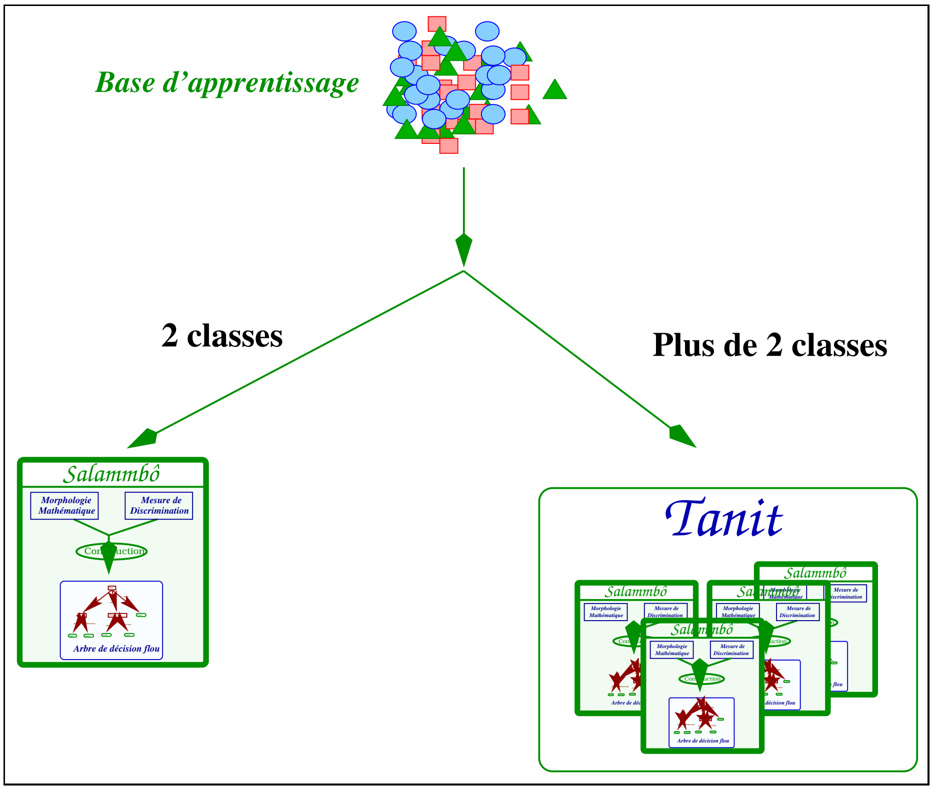


Figure 3.2 – *Schéma général : Tanit et Salammbô*

habituels des arbres de décision sont souvent assez décevant. Finalement, le paramétrage de la mesure de discrimination utilisée par Salammbô est facilité par la méthode de classification des mesures basée sur une hiérarchie de fonctions que nous proposons.

Ainsi, dans le premier chapitre de cette partie, nous proposons une technique automatique de construction de partition floues.

Dans le deuxième chapitre de cette partie, nous proposons une méthode de construction de forêts d'arbres de décision flous pour la prise en compte de problèmes où il existe plus de deux classes à apprendre.

Dans le troisième chapitre, nous introduisons la hiérarchie de fonction qui permet de sélectionner et de construire une bonne mesure de discrimination pour construire un arbre de décision flous.

Finalement, nous présentons les résultats des expérimentations que nous avons obtenus avec notre application Salammbô et les améliorations proposées.

Chapitre 4

Construction de partitions floues

Et si nous mettions un peu de flou ?...
Flou, comme la vérité...

D. Pennac – *Monsieur Malaussène*

4.1 Introduction

Apprendre à partir d'un ensemble de valeurs numériques entraîne des difficultés très importantes de spécialisation des données.

Ainsi, une valeur numérique est une donnée trop *parfaite* pour l'apprentissage dans le sens où elle est trop précise. Il n'est pas possible de généraliser avec des données précises, il est nécessaire de les regrouper entre elles pour leur faire perdre leur spécialisation et autoriser ainsi leur utilisation dans des situations futures. Une telle phase de regroupement s'appelle : la *discrétisation* des valeurs numériques.

Par exemple, pour déterminer si la personne C, âgée de 21 ans, a le droit de voter en ayant remarqué que mademoiselle A qui a 20 ans et que monsieur B qui a 24 ans peuvent voter, il est nécessaire de généraliser les valeurs numériques en les rendant moins précises : les personnes entre 20 et 24 ans votent.

Il existe différentes façons pour discrétiser des valeurs numériques. Les méthodes classiques procèdent par regroupement de valeurs, comme dans l'exemple donné, et cherchent à déterminer des points de coupure dans l'ensemble continu ou infini[1] des valeurs. Ces points de coupure sont alors utilisés comme seuil dans cet ensemble et ils permettent alors d'induire un ensemble fini de valeurs (dites alors *valeurs discrètes*). Les méthodes se différencient entre elles par le moyen utilisé pour trouver le point de coupure.

4.1.1 Assouplir les coupures

Le problème majeur de telles méthodes est alors que l'aspect continu des données est abandonné au profit de leur discrétisation. Ainsi, les points de coupure sont autant

1. Il est à noter qu'il suffit que l'ensemble considéré comporte un grand nombre de valeurs et ne soit pas obligatoirement un ensemble continu ou infini.

de frontières strictes introduites pour séparer les valeurs les unes des autres sans tenir compte des liens qui peuvent exister entre des valeurs de chaque côté immédiat de la frontière pourtant abstraite. En généralisation, cela amènera des différences de caractérisation de comportement pour des données qui sont identiques et donc, entraînera une augmentation du taux d'erreur. Au delà des chiffres, l'aberration intrinsèque d'une telle discrétisation apparaît quand elle est utilisée pour généraliser des propriétés.

Il est clair que l'Homme ne discrétise jamais les valeurs numériques de façon catégorique dans son mode de pensée. Une distance *humaine* s'exprime en *mètres* et non pas en [2] *fois 1650763,73 longueurs d'onde dans le vide de la radiation correspondant à la transition entre les niveaux* $2p_{10}$ *et* $5d_5$ *de l'atome de Krypton 86* tout comme un âge *humain* ne s'exprime pas en *secondes, minutes, heures, jours, mois, années* mais plutôt en *ans...* Ainsi, par exemple, il est évident que n'importe quelle personne à qui je dirais que le 2^{bis} se trouve à 100 mètres de Jussieu, serait bien consciente que je ne garantis nullement en cela que la distance qui sépare Jussieu du 2^{bis} est exactement de 165076373 longueurs d'onde dans le vide de la radiation correspondant à la transition entre les niveaux $2p_{10}$ et $5d_5$ de l'atome de Krypton 86. Humainement parlant, nous sommes attirés par le fond, l'ordre de grandeur, plutôt que par la forme, la précision rigoureuse.

C'est pourquoi, il est intéressant d'introduire un nouveau type de discrétisation qui autorise la considération de l'aspect imprécis des points de coupure générés. Une *partition floue* est alors construite sur l'ensemble continu des valeurs, qui est une discrétisation plus souple de cet ensemble de valeurs.

Dans ce cas, soit les anciennes méthodes de discrétisation sont utilisées telles quelles et les seuils trouvés sont considérés comme des seuils flous, soit de nouvelles méthodes de construction de partition floues sont proposées.

Il est aussi à noter que dans notre contexte, chaque valeur de l'ensemble qui doit être discrétisé possède une information supplémentaire, telle une deuxième dimension, qui est la classe qui lui est associée. Cette information doit être utile pour le processus de discrétisation qui devient alors une partie importante de la phase d'apprentissage.

4.1.2 Discrétiser au bon moment

Un autre aspect qui intervient dans notre contexte est le choix du moment où la discrétisation des attributs numériques doit avoir lieu. Dans le cadre des arbres de décision, cette discrétisation peut intervenir avant la construction de l'arbre. Dans ce cas, avant tout choix d'attribut, les points de coupure, flous ou non flous, sont recherchés et ils seront utilisés de façon systématique dans tout le reste du processus d'apprentissage. Cela paraît étonnant de procéder ainsi car il est évident que la discrétisation d'un attribut est dépendante de l'ensemble *courant* sur lequel cette discrétisation est effectuée. Pour les arbres de décision, les répercutions d'une discrétisation se révèlent quand un test sur l'attribut numérique doit être choisi. Le test considéré est fortement dépendant des questions qui ont été posées au préalable.

Par exemple, prenons le cas où l'âge requis pour voter doit être déterminé à partir d'une base d'apprentissage comportant des personnes, femmes et hommes, de tous âges.

2. *Dictionnaire de la langue française* – Hachette, 1990.

Admettons de plus, que cette base a été constituée avant l'année 1945, date à laquelle les femmes ont obtenu le droit de vote en France. La discrétisation brute de l'attribut âge pour cette base ne donnera rien de concluant, car les femmes n'ayant pas le droit de voter, aucun âge n'est discriminant pour arriver à séparer les votant des non votants. Par contre, après la question discriminant les femmes des hommes, la discrétisation de l'attribut âge sera immédiate et valide : 21 ans pour les hommes et aucune valeur pour les femmes qui de toutes façons ne pouvaient pas voter.

Encore une fois, cet exemple permet de ce rendre compte qu'il existe des cas, la plupart du temps, où il n'est pas judicieux de discrétiser *a priori*, dans une étape préalable du processus d'apprentissage.

Dans ce chapitre, dans un premier temps, quelques anciennes méthodes de discrétisation sont présentées, ainsi que leur extension pour considérer les coupures d'une manière floue. Ensuite, une méthode automatique de construction de partition floue est proposée, basée sur un processus de raisonnement proche du processus humain de raisonnement dans une situation similaire.

4.2 Discrétisation des attributs numériques

Comme il l'a été précisé dans l'introduction de ce chapitre, pour discrétiser un ensemble continu de valeurs numériques, les méthodes conventionnelles recherchent des points de coupure sur l'ensemble des valeurs. Dans le contexte des arbres de décision, ces points vont servir à libeller les tests aux nœuds de l'arbre. Par exemple, dans l'arbre donné (Figure 2.1), un point de coupure a été trouvé pour l'attribut *temps restant* à la valeur $60s$. Les chemins issus de ce nœud sont utilisés, pour l'un par les valeurs inférieures à 60, pour l'autre par les valeurs supérieures à 60.

Dans cette section, quelques méthodes de discrétisation, classiques et floues, sont présentées. Pour quelques précisions supplémentaires, il peut être intéressant de consulter les articles de (Dougherty et al., 1995) et de (Rabaséda-Loudcher et al., 1995), qui, sans être exhaustifs, étudient, chacun de leur côté, quelques autres méthodes de discrétisation. De même, un chapitre du livre de (Klir et Yuan, 1995) est consacré à la construction de sous-ensembles flous.

4.2.1 Les méthodes à seuils classiques

Les méthodes de discrétisation qui n'utilisent pas l'information telle la classe associée à chaque valeur de l'ensemble à discrétiser, ne sont pas très intéressantes à examiner. En effet, les discrétisations qu'elles engendrent ne sont en rien liées à la répartition que peuvent suivre les classes. Un exemple de telles méthodes est celle basée sur le découpage de l'univers des valeurs en intervalles de longueur égale. Il est alors permis de se demander dans quelle mesure la pertinence des points de coupure trouvés peut aider le processus d'apprentissage censé utiliser cette discrétisation. Ainsi, par exemple, cela équivaudrait à discrétiser l'attribut *âge* en utilisant l'espérance de vie des français pour déterminer un test évaluant la capacité de voter ou de ne pas voter... Il est clair que l'âge requis pour voter, l'âge de majorité, n'est pas dépendant de l'espérance de vie de la population, tout au moins dans la constitution française.

C'est pourquoi, les méthodes examinées par la suite font toutes usage de la (ou des) classe(s) associée(s) aux valeurs.

La méthode de discrétisation choisie dans le système CART (Breiman et al., 1984) est basée sur l'utilisation d'un critère d'entropie pour chercher les points de coupure servant à discrétiser l'ensemble des valeurs numériques. Un point de coupure décompose l'ensemble des valeurs numériques en deux sous-ensembles : le sous-ensemble des valeurs inférieures à ce point et le sous-ensemble des valeurs supérieures. Dans le cadre de l'apprentissage, la pureté, en termes de classification, de chaque sous-ensemble peut être alors évaluée, et, par conséquent, la pertinence du point de coupure peut être associée à une mesure d'information. Le point de coupure sélectionné est celui qui minimise la valeur d'entropie relativement à la classe des sous-ensembles qu'il engendre. C'est une méthode très naturelle et intuitive dans le sens où la mesure qui sera utilisée pour comparer les tests sur les attributs entre eux, sert aussi à chercher les points de discrétisation sur les ensembles continus. Elle souligne d'autant plus la similarité des deux processus qui s'attachent à comparer entre eux des tests sur des valeurs d'attributs. Dans le système CART, la phase de discrétisation des attributs numériques s'effectue à chaque étape de la construction de l'arbre, à chaque fois qu'un test sur un attribut doit être trouvé.

Ce type de méthode de discrétisation peut être raffiné pour améliorer la complexité de l'algorithme de recherche. Ainsi, il peut être démontré qu'il n'est pas nécessaire de considérer tous les points de coupure possibles, mais que seuls ceux qui séparent des valeurs de deux classes différentes sont intéressants (Fayyad et Irani, 1992b; Fayyad et Irani, 1993). De plus, (Fayyad et Irani, 1993) introduisent un nouveau critère, basé sur l'entropie et le principe de longueur de description minimum[3] pour chercher des points de coupure.

Par contre, l'amélioration proposée par (Catlett, 1991) qui propose de calculer tous les points de coupure dans une étape préalable du processus d'apprentissage, reste contestable. La raison avancée par l'auteur pour ce type d'amélioration est d'éviter de trier les valeurs des ensembles à discrétiser à chaque étape du choix d'un nœud dans la construction d'un arbre de décision. Cet argument est inepte dans le sens où il est dépendant de la structure de données utilisée pour implémenter l'algorithme de construction. Par exemple, si la base d'apprentissage est mémorisée sous la forme d'une listes d'exemples (en LISP ou en PROLOG) ou sous la forme d'un tableau (en langage C ou en PASCAL), il est alors nécessaire de trier cet ensemble à chaque étape de la construction de l'arbre. Par contre, en choisissant une structure de données plus élaborée comme une liste de listes pointées ou un ensemble de pointeurs, il est alors facile d'optimiser l'algorithme afin de ne pas avoir à retrier la base d'apprentissage à chaque étape.

La mesure d'information utilisée, souvent l'entropie de Shannon, peut aussi être remplacée par une mesure de contraste[4] pour évaluer les points de coupure entre eux (Van de Merckt, 1993). Toute seule, une mesure de contraste peut ne pas être satisfaisante dans le cadre de l'apprentissage, c'est pourquoi, (Van de Merckt, 1993) construit un indicateur en combinant une mesure d'entropie et une mesure de contraste pour chercher

3. Minimum Description Length.

4. La mesure de contraste de (Van de Merckt, 1993) est donné par l'équation 2.7 du paragraphe 2.2.5.

les meilleurs points de coupure possibles.

De nombreuses autres méthodes existent encore pour trouver ces points de coupure, par exemple l'algorithme du CHIMERGE de (Kerber, 1992) qui est basé sur l'utilisation de la mesure statistique du χ^2 pour fusionner des intervalles de valeurs entre eux jusqu'à ce qu'un certain seuil de χ^2 soit atteint. C'est un algorithme original dans le sens où il part des valeurs elles-mêmes et qu'il essaye, d'une manière ascendante, de les regrouper.

La méthode FUSINTER de (Zighed et al., 1996; Rabaséda-Loudcher, 1996) se propose, elle, de ne chercher des points de coupure que sur les ensembles de valeurs qui en valent la peine, c'est-à-dire, ceux sur lesquels la répartition des classes laisse augurer un découpage intéressant. Un découpage intéressant est un découpage où un point de coupure sépare convenablement et efficacement un grand nombre de valeurs. Pour évaluer l'intérêt d'un ensemble en termes de qualité dans la répartition des classes, l'algorithme FUSINTER utilise un test statistique (le test de Wald et Wolfowitz généralisé en test de Mood (Zighed et al., 1996; Rabaséda-Loudcher, 1996)) capable de rendre compte de la séparabilité des courbes de répartition. Ensuite, pour les ensembles candidats à la discrétisation, les points de coupure sont recherchés à l'aide d'une mesure de discrimination particulière, la mesure d'entropie quadratique qui vérifie un lot de propriétés requises par les auteurs.

Il y a aussi l'algorithme de discrétisation proposé par (Pazzani, 1995) dans le domaine des classifieurs bayésiens dont l'originalité est de procéder par regroupement ou éclatement d'intervalles, le tout supervisé par une méthode de validation croisée.

Cette méthode est proche de celle de (Henrichon et Fu, 1969). Dans cet article, les auteurs procèdent aussi par regroupement d'intervalles proches quand ceux-ci sont libellés par la même classe. De plus, ils introduisent, et c'est un aspect très original de leur approche, une nouvelle classe en supplément des classes existantes. Cette nouvelle classe dénote l'"indécision". Elle sert à marquer les intervalles de valeurs dans lesquels aucune classe existante n'est majoritaire. Ce sont ces intervalles qui, dans le cas où il existe plusieurs dimensions pour décrire les données, sont, par la suite, à nouveau discrétisés sur une nouvelle dimension. Leur algorithme est finalement assez similaire aux futurs algorithmes de construction d'arbres de décision car c'est l'enchaînement des discrétisations sur les attributs qui permettront de déterminer la classe des exemples à classer.

Toutes ces méthodes ont l'avantage de tenir compte de la répartition des classes sur l'ensemble des valeurs de l'attribut numérique dans la recherche de ce point de coupure. Elles paraissent, en cela, meilleures que les méthodes de découpage qui ne tiennent pas compte de cette répartition.

4.2.2 Les méthodes à seuils flous

Comme il a été précisé dans l'introduction, l'inconvénient des méthodes de discrétisation classiques vient du fait que le point de coupure engendré durant la discrétisation aura un impact très important durant la phase de généralisation. Le résultat du test vis-à-vis de cette valeur est déterminant pour la suite de la classification. En fonction de ce résultat, l'exemple à classer cheminera dans une partie de l'arbre complètement différente. Ce seuil a donc une influence prépondérante et décisive alors que sa validité n'est pas intrinsèquement irréfutable en raison de bruit dans les données, ou de données manquantes. Par exemple, dans l'arbre de la Figure 2.1, si le *temps restant* est de $59s$ ou de $60s$, la décision prise ne sera pas la même alors que la différence entre ces deux valeurs n'est pas réellement importante.

(Carter et Catlett, 1987) ont alors introduit un moyen de rendre ces tests moins catégoriques en utilisant des probabilités pour visiter toutes les branches issues du nœud en question. Cette méthode a aussi été reprise par (Quinlan, 1990) pour construire ce qu'il appelle des *arbres probabilistes*.

Une méthode similaire mais fondée sur des degrés d'appartenance à chacune des branches issues d'un nœud peut alléger encore la complexité de cette construction (Marsala, 1994),(Bouchon-Meunier et al., 1995). Dans cette méthode, un attribut numérique est discrétisé à l'aide de la mesure d'entropie de Shannon. Habituellement, avec cette discrétisation, deux valeurs consécutives de la base d'apprentissage sont trouvées pour servir à calculer, par la moyenne, la valeur test du nœud de l'arbre de décision. En fait, ces deux valeurs peuvent aussi être utilisées pour déterminer les noyaux de deux sous-ensembles flous. Ainsi, une partition floue rudimentaire est construite sur un ensemble de valeurs numériques.

Cette méthode est équivalente à celle proposée par (Jang, 1994) dans laquelle ce sont des courbes gaussiennes qui sont utilisées pour modéliser les degrés d'appartenance, et non plus des fonctions affines par morceaux.

(Zeidler et Schlosser, 1996) proposent aussi une méthode identique mais en considérant un intervalle autour des points de coupure trouvés comme zone floue.

Cependant, une limite de ce type de méthodes provient de l'importance de la zone floue ainsi créée autour des points de test. Cette zone reste peu étendue et peu de valeurs en bénéficieront durant la phase de classification. C'est pourquoi il peut être intéressant d'utiliser une vraie partition floue sur l'univers des valeurs d'un attribut numérique, au lieu d'utiliser de telles méthodes classiques étendues.

4.2.3 Des méthodes de construction de partitions floues

La méthode SAFI pour construire des arbres de décision flous repose sur la connaissance de partitions floues de l'univers des attributs numériques. Ces partitions floues doivent être données par un expert du domaine afin d'assurer un minimum de cohérence à l'arbre inféré.

Mais, même un expert peut se tromper, et les partitions qu'il est capable de donner sont susceptibles de ne pas prendre en compte entièrement la réalité du phénomène observé. Dans le système SAFI, l'expert peut modifier, de façon interactive, ses partitions floues en fonction des données d'apprentissage, en ajustant un degré d'étalement entre les modalités qui composent la partition (Ramdani, 1994). Cette méthode reste approximative et un *degré d'étalement minimal* entre les noyaux des sous-ensembles flous doit être imposé pour éviter que l'optimisation des partitions floues ne génère une partition ordinaire de l'univers des valeurs. La détermination de ce degré est alors très arbitraire et alourdit encore la paramétrisation du processus d'apprentissage.

D'autre part, il n'est pas toujours possible d'obtenir des partitions floues de la part des experts du domaine étudié, soit parce que le phénomène à comprendre est totalement inconnu, soit parce que de telles partitions floues ne sont pas *a priori* évidentes.

Il est donc crucial de se donner une méthode automatique susceptible de fournir une partition floue sur un univers de valeurs numériques.

Il existe un grand nombre de méthodes de construction de fonctions d'appartenance. La plupart sont des méthodes statistiques de construction mais il existe aussi quelques méthodes automatiques issues des travaux sur les réseaux neuronaux ou les algorithmes génétiques (Aladenise et Bouchon-Meunier, 1997). Mais souvent, comme le soulignent ces auteurs dans leur conclusion "*Il existe bien d'autres méthodes d'obtention de fonctions d'appartenance propres aux systèmes ou au domaine pour lequel elles ont été conçues*".

Dans le contexte d'apprentissage inductif, il est indéniable qu'une méthode automatique, fonctionnant de façon autonome et basée sur la base d'apprentissage, est plus intéressante et plus satisfaisante. En effet, le but des systèmes d'apprentissage à concevoir est aussi de leur assurer une certaine autonomie dans le sens où il doit leur être possible d'évoluer de façon autonome dans leur environnement. Cela s'entend aussi bien dans un environnement dit du monde réel que dans un environnement tel qu'une base de données dans le cadre de la découverte de connaissances. Il est clair qu'un système automatique de découverte de connaissances se doit de retrouver les liens entre les connaissances par lui-même.

Comme exemple de méthode automatique, et sans revenir sur les méthodes neuronales (Ménage, 1996) et génétiques (Sanchez et al., 1997), la méthode de (Yuan et Shaw, 1995) a, de plus, l'avantage pour nous d'être une méthode proposée pour un système de construction d'arbres de décision flous. Dans cette méthode, pour déterminer une partition d'un univers de valeurs numériques en plusieurs modalités triangulaires floues, un certain nombre d'intervalles de taille égale, correspondant au nombre de modalités désirées, sont créés sur l'univers des valeurs numériques. Sur chacun de ces intervalles, un sous-ensemble flou de base est défini de façon triangulaire, son noyau est pris égal au milieu de l'intervalle considéré. Ensuite, un algorithme d'itération essaye d'optimiser la valeur du centre de chaque modalité en mesurant la distance de chaque valeur de la base d'apprentissage aux centres existants et en modifiant la valeur des centres les plus proches. L'inconvénient d'une telle méthode est que la détermination de la partition floue pour un ensemble numérique est indépendante de la classe à reconnaître par l'apprentissage.

Une autre méthode issue d'un algorithme de construction d'arbres de décision flous et qui permet d'inférer une partition floue sur l'ensemble des valeurs d'un attribut, est la méthode proposée par (Wehenkel, 1997). Elle est basée sur l'utilisation de la mesure de discrimination de Kolmogorov-Smirnov [5], normalisée à l'aide d'une mesure de contraste. Cette mesure a beaucoup de qualités en particulier celle de n'être pas convexe ce qui, comme il a été montré par les auteurs (Wehenkel, 1996), a l'avantage de ne pas favoriser

5. Dans un cadre où il existe deux classes à reconnaître, la mesure de Kolmogorov-Smirnov retourne la distance maximale qui sépare les distributions de probabilité de chacune des classes :

$$\mathcal{K}(c_1, c_2) = \max_x(|P_{c_1}(x) - P_{c_1}(x)|)$$

avec P_{c_1} (resp. P_{c_2}) la distribution de probabilité de la classe c_1 (resp. c_2).

les cas non flous vis-à-vis des cas flous. Les auteurs précisent aussi que, de par sa forme, sans que cela soit une limitation, elle ne peut s'appliquer que dans le cadre de recherche de partitions binaires et de classes aussi binaires.

(Narazaki et Ralescu, 1994) proposent eux aussi une méthode pour inférer une fonction d'appartenance dans un contexte d'apprentissage. Sur un ensemble de valeurs numériques, appelées dans ce cas *éléments*, la valeur moyenne des éléments possédant chacune des classes est calculée. Cette valeur moyenne permet alors de calculer, pour tout élément, la distance de cet élément à la classe. Ensuite, connaissant la distance d'un élément à chaque classe de l'ensemble, il est aisé d'en déduire un degré d'appartenance de cet élément à chacune des classes en effectuant une répartition rationnelle. Ainsi, une fonction d'appartenance est construite et peut même directement être utilisée comme moyen de classification. On retrouve là des notions de classification prototypique : un ensemble de prototypes est extrait d'une base d'apprentissage, puis, pour classer une nouvelle valeur, sa distance à chaque prototype est calculée afin de lui associer la classe du prototype le plus proche (pour plus de détails sur les prototypes, lire (Rifqi, 1996)). Chez (Narazaki et Ralescu, 1994), et comme il est question de prototypes de valeurs numériques, la distance aux prototypes de chaque élément de la base d'apprentissage sert à construire une fonction d'appartenance aux classes de ces prototypes.

4.3 Une nouvelle méthode automatique de construction de partition floue

La nouvelle méthode que nous proposons dans cette section a l'avantage d'être automatique, simple à mettre en œuvre, de tenir compte de la répartition des classes sur l'ensemble des valeurs dans la base d'apprentissage et de mimer le comportement d'un être humain qui aurait à effectuer le même travail (Marsala, 1995; Marsala et Bouchon-Meunier, 1996).

Cette méthode de construction de partitions floues ne dépend que de la base d'apprentissage $\mathcal{E}$. Elle est calquée sur un processus intuitif de construction de telles partitions floues que pourrait mettre en œuvre une personne qui aurait cette tâche à accomplir. Ce processus est formalisé par le biais de la théorie de la morphologie mathématique (Serra, 1982), (Coster et Chermant, 1989). Cette théorie autorise un prétraitement des données proche du prétraitement opéré par l'esprit humain dans le même type de problèmes.

En raison des spécificités du problème de la construction de partitions floues sur un univers de valeurs numériques, les opérateurs morphologiques peuvent se modéliser à l'aide de la théorie des langages formels (Ginsburg, 1966), (Autebert, 1987). Les opérateurs morphologiques sont traduits sous la forme d'automates issus de la théorie des langages pour être plus facilement implémentables.

Le but de cette partie est donc de proposer cette méthode de construction d'une partition floue sur un univers continu de valeurs numériques respectant, au mieux, une certaine répartition des classes associées à certaines valeurs de l'univers (les valeurs d'apprentissage). Dans un premier temps, le processus humain intuitif de construction de partitions floues est présenté. La deuxième partie de cette section comporte la description des opérateurs de la morphologie mathématique utilisés pour formaliser ce processus. La troisième partie présente la modélisation des opérateurs morphologiques sous la forme d'automates, dans le contexte particulier de la construction de partitions floues sur un univers de valeurs numériques. Finalement, l'algorithme complet de construction de partitions floues, implémenté dans l'application Salammbô, est présenté.

4.3.1 Un processus humain intuitif

La méthode intuitivement mise en œuvre par un esprit humain pour la construction d'une partition floue relative à un ensemble de classes, sur un univers de valeurs numériques est basée sur la création de groupe de valeurs qui, *a priori*, se ressemblent en possédant la même classe. La souplesse de l'esprit humain l'autorise à admettre la présence, dans un groupe de valeurs d'une même classe, d'un certain nombre de valeurs de classes différentes.

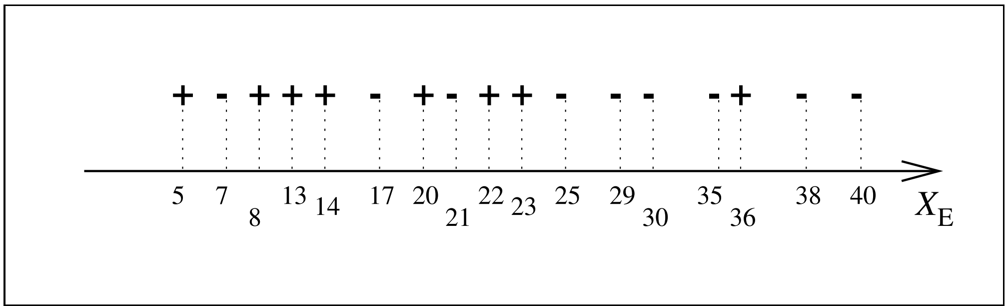


Figure 4.1 – *La base d'apprentissage vue graphiquement*

Par exemple, soit $\mathcal{E}$ la base d'apprentissage suivante : {(5, +), (7, -), (8, +), (13, +), (14, +), (17, -), (20, +), (21, -), (22, +), (23, +), (25, -), (29, -), (30, -), (35, -), (36, +), (38, -), (40, -)}. L'univers continu des valeurs numériques sur lequel certaines valeurs (celles données dans $\mathcal{E}$) sont associées à l'une des deux classes + ou - est noté X_E. X_E est un univers de valeurs ordonnées. Sa borne inférieure est notée B_{inf} et sa borne supérieure est notée B_{sup}. Si l'univers des valeurs est l'ensemble de toutes les valeurs réelles possibles, B_{inf} sera $-\infty$ et B_{sup} sera $+\infty$. $\mathcal{E}$ peut se représenter graphiquement, en une dimension selon un axe portant X_E (Figure 4.1).

Pour définir une partition floue à partir de $\mathcal{E}$, un être humain aura plutôt tendance à travailler sur la représentation graphique de $\mathcal{E}$ plutôt que sur sa représentation formelle. C'est le processus humain sur cette représentation graphique qui est examiné par la suite.

Dans le cas où X_E doit être partitionné en deux sous-ensembles flous, la première étape à effectuer est la définition *a priori* des noyaux de ces sous-ensembles flous. C'est, en tout cas, celle qui semble la plus naturelle étant donné que le noyau d'un sous-ensemble flou est l'ensemble des éléments qui *appartiennent absolument* à ce sous-ensemble et donc l'ensemble le plus facilement caractérisable. Ici, les éléments sont des valeurs numériques associées à une classe et deux intervalles disjoints de X_E doivent être définis pour servir de noyaux aux deux sous-ensembles flous de la partition.

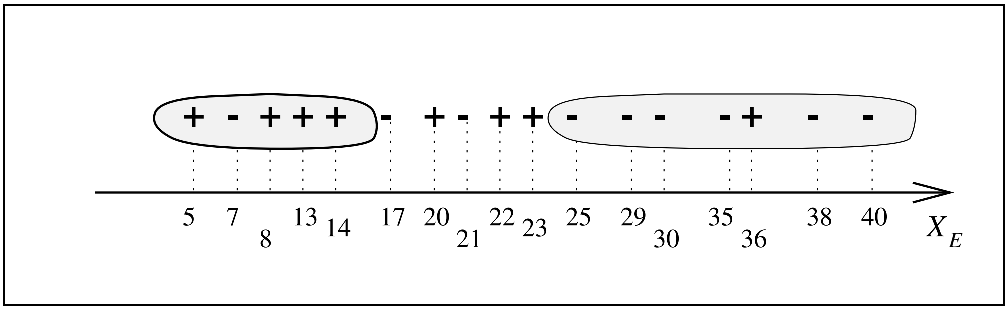


Figure 4.2 – *La base d'apprentissage vue graphiquement avec 2 sous-ensembles possibles*

Définir formellement la notion d'*appartenance absolue* n'est pas une chose des plus évidentes. Cela peut vouloir signifier "l'élément appartient absolument à l'ensemble si sa classe est la même que la classe majoritaire dans l'ensemble" mais cela signifie aussi "intuitivement, il appartient absolument à l'ensemble, bien qu'il n'ait pas la même classe, mais il est cerné par de nombreux éléments de la classe différente".

D'après la vue graphique de $\mathcal{E}$, deux tels ensembles peuvent apparaître correspondant aux deux intervalles $[5, 14]$ et $[25, 40]$ (Figure 4.2). À partir de ces deux noyaux, la partition est immédiate : la partie floue de chaque sous-ensemble flou correspond aux éléments compris entre les deux noyaux (Figure 4.3).

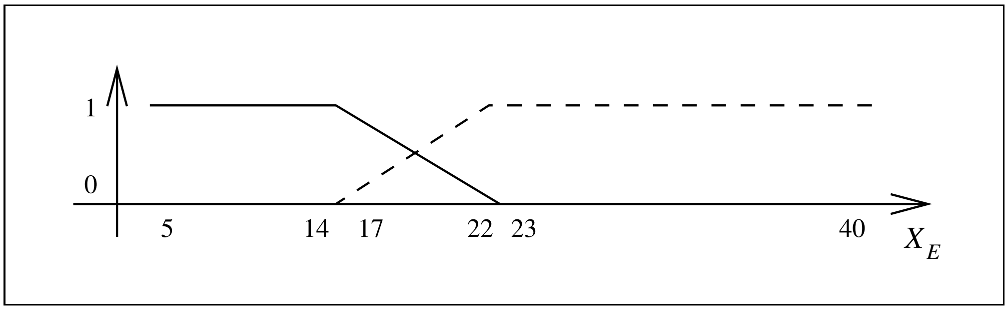


Figure 4.3 – *Partition floue possible de* X_E

Informatiser ce processus

Le processus qui vient d'être décrit pour inférer une partition floue est un processus naturel, en ce sens qu'il est naturellement mis en œuvre par la plupart des êtres humains qui ont à construire une telle partition à partir d'une telle base d'apprentissage. Mais doter un ordinateur d'un processus de construction équivalent n'est pas immédiat. Un moyen de le formaliser est d'utiliser les opérateurs de la morphologie mathématique. Ces opérateurs reflètent d'une façon assez fidèle les étapes de la recherche des meilleurs ensembles comme noyaux des sous-ensembles flous et semblent convenir à la formalisation de la notion d'*appartenance absolue* exprimée précédemment.

4.3.2 La théorie de la morphologie mathématique pour modéliser ce processus intuitif

En traitement d'images, les opérateurs de la théorie de la morphologie mathématique sont utilisés à la fois pour lisser des objets, c'est-à-dire gommer les petites aspérités de sa forme, mais aussi pour éliminer les artefacts de mesures comme, par exemple, des points blancs qui apparaîtraient sur une image noire par suite de bruit (Coster et Chermant, 1989), (Serra, 1982). Mais ils sont aussi utilisés dans d'autres domaines, tels que, par exemple, la programmation du jeu de Go (Bouzy, 1995), pour isoler des blocs d'éléments similaires dans un univers uniforme. L'analogie de la recherche des noyaux des sous-ensembles flous et ces types de traitement d'images apparaît suffisamment éloquente pour utiliser ces opérateurs pour la construction de partitions floues.

Dans cette section, les opérateurs de la théorie de la morphologie mathématique sont présentés. À partir des deux opérateurs de base, l'érosion et la dilatation, les deux opérateurs d'ouverture et de fermeture sont définis. Ces derniers opérateurs sont les constituants de l'opérateur de filtrage.

Les opérateurs morphologiques de base : l'érosion et la dilatation

Les opérateurs de base de la morphologie mathématique, l'*érosion* et la *dilatation*, agissent sur des corps morphologiques existant dans un espace donné, en les modifiant. Cette modification se fait relativement à un *élément structurant* noté $\mathcal{E}$. L'*érosion* d'un corps morphologique $\mathcal{C}$ relativement à l'élément structurant $\mathcal{E}$ est une certaine soustraction de $\mathcal{E}$ dans $\mathcal{C}$, alors que la *dilatation* de $\mathcal{C}$ relativement à $\mathcal{E}$ est une addition particulière de $\mathcal{E}$ dans $\mathcal{C}$ (Figure 4.4).

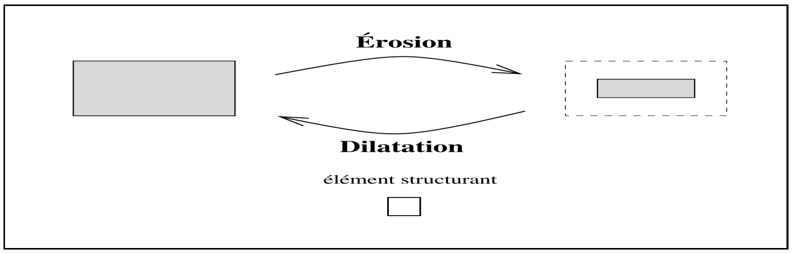


Figure 4.4 – *Opérateurs de base de la morphologie mathématique*

Ces deux opérateurs ne sont pas exactement duaux, leur application en séquence ne conserve pas forcément tous les corps morphologiques de l'espace. Chaque opérateur a des effets irréversibles que l'autre ne peut pas pallier : l'érosion annihile les petites entités morphologiques, la dilatation fusionne les entités imposantes séparées par des petits corps. Ainsi, il existe deux possibilités pour combiner en séquence ces deux opérateurs, ce qui donne naissance aux deux opérateurs suivants.

Les opérateurs d'ouverture et de fermeture

Ces deux opérateurs agissent eux aussi sur des corps morphologiques. Une *ouverture* morphologique est la combinaison d'une érosion suivie d'une dilatation relative à un élément structurant $\mathcal{E}$ sur le même corps morphologique $\mathcal{C}$. Elle permet la destruction des petits corps de l'espace, en fonction de la taille de l'élément structurant $\mathcal{E}$. D'un point de vue topographique, *la transformation par ouverture adoucit les contours, coupe les isthmes étroits, supprime les petites îles et adoucit les caps étroits* (Coster et Chermant, 1989).

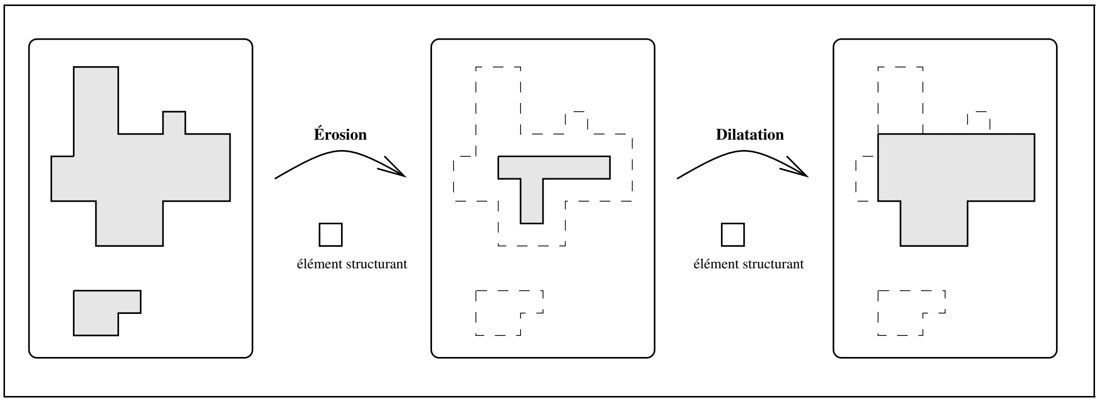


Figure 4.5 – *L'opérateur d'ouverture*

Une *fermeture* morphologique est la combinaison d'une dilatation suivie d'une érosion relative à un élément structurant $\mathcal{E}$ sur le même corps morphologique $\mathcal{C}$. Elle permet la destruction des petits "vides" de l'espace séparant des corps importants, ou le comblement de petites cavités. *La transformation par fermeture bouche les canaux étroits, supprime les petits lacs et les golfes étroits* (Coster et Chermant, 1989).

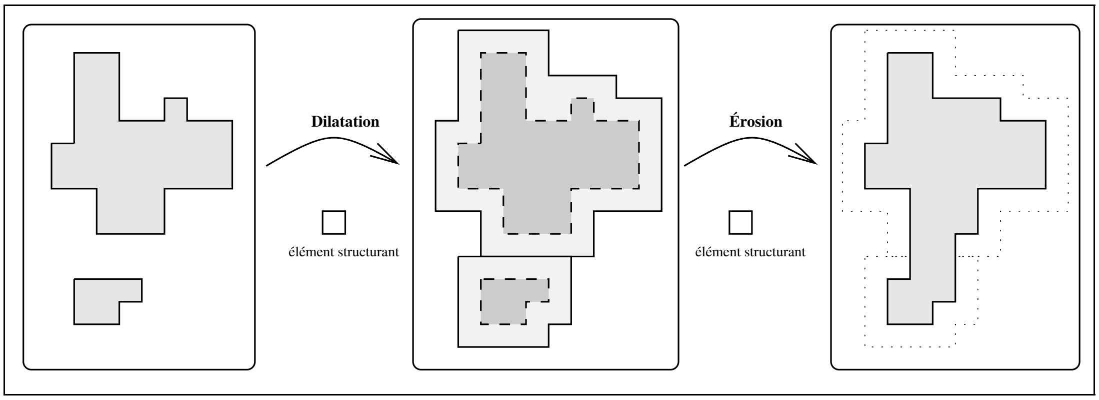


Figure 4.6 – *L'opérateur de fermeture*

Ces deux opérateurs peuvent eux-mêmes être combinés pour obtenir un traitement progressif des objets de l'espace considéré. Ce traitement autorisera le filtrage de l'espace est des éléments morphologiques qui le composent pour en faire ressortir les éléments caractéristiques et en éliminer les entités peu représentatives.

Le filtre alterné

Un *filtre alterné* est la combinaison d'ouvertures suivies par des fermetures. Il est composé par n ($n \in \mathbb{N}$) ouvertures successives suivies par n fermetures successives, appliquées à tous les corps de l'espace, avec le même élément structurant $\mathcal{E}$. Il permet la destruction des petits corps de l'espace, en fonction de la taille de $\mathcal{E}$ et, simultanément, il permet de combler les espaces inutiles entre et dans les corps de l'espace. La valeur de n sert à mesurer la force avec laquelle les éléments de l'espace vont être traités. Une grande valeur ne fera survivre que les corps vraiment conséquents et éradiquera les aspérités et les espaces imposants.

4.3.3 De la morphologie mathématique à la théorie des langages

Les éléments de la morphologie mathématique sont généralement appliqués en traitement d'image sur des espaces à deux dimensions. Or, à propos de la représentation graphique de la Figure 4.1, il a déjà été souligné l'aspect unidimensionnel d'une base d'apprentissage. Par conséquent, pour formaliser les opérateurs morphologiques présentés dans la section précédente, il est suffisant de ne considérer que la restriction de ces opérateurs permettant d'agir sur des objets ayant une unique dimension. Un mot est un exemple typique d'objet à une seule dimension physique. C'est pourquoi, la théorie des langages formels (*cf.* Annexe A) est utilisée pour formaliser et utiliser les éléments de la morphologie mathématique présentés.

Un mot d'un langage donné est une entité à une dimension sur laquelle des opérateurs sont définis pour l'altérer, ou plutôt pour la *réécrire*. De tels opérateurs sont des *systèmes de réécriture*. A l'aide de règles de réécriture, ils autorisent la réécriture d'un mot en un autre mot, suivant certaines règles imposées. Ainsi, des transformations sur des entités unidimensionnelles peuvent-elles être définies au moyen de tels systèmes de réécriture.

Les automates sont des formes particulières de systèmes de réécriture qui n'agissent que sur certains type de langages, les langages dit *rationnels*. Ils sont intéressants dans le sens où ce sont les plus simples à étudier des systèmes de réécriture. De plus, ils sont aisément implémentables en machine. Les automates utilisés ici sont des transductions rationnelles, qui sont des extensions des automates classiques. Une transduction rationnelle, ou plus simplement transduction, est un automate avec sortie. Chaque transition d'un automate possède une entrée, ce qui est lu, et qui permet de changer d'état. En plus, chaque transition d'une transduction possède une sortie, ce qui est écrit, qui n'a aucun rôle sur le changement d'état de l'automate, mais qui est écrit quand la transition est activée.

Dans cette section, une modélisation des opérateurs de la théorie de la morphologie mathématique est présentée. La première partie de la section décrit la phase de traduction d'une base d'apprentissage en un mot d'un langage. Le reste de la section est orientée vers la construction d'un opérateur morphologique de filtrage s'appliquant sur les mots d'un tel langage. Comme on l'a vu dans la section 4.3.2 précédente, un opérateur de filtrage est défini par deux opérateurs d'ouverture et de fermeture, eux-mêmes construits à l'aide d'opérateurs d'érosion et de dilatation. C'est pourquoi, pour définir un opérateur de filtrage sur des mots d'un langage, il faut d'abord définir ces deux opérateurs de base que sont l'érosion et la dilatation. Ces deux opérateurs autorisent alors la construction des opérateurs d'ouverture et de fermeture qui constituent les deux éléments de base de l'opérateur de filtrage.

La base d'apprentissage est un mot

Une base d'apprentissage donnée $\mathcal{E}$ permet de définir un alphabet[6] $\mathcal{L}$, qui est un ensemble de lettres, pour lequel chaque lettre est associée à une classe et une seule de la base. Par conséquent, pour un attribut donné, la base d'apprentissage peut être transformée en un mot sur $\mathcal{L}^*$, l'ensemble de tous les mots possibles construits avec les lettres de $\mathcal{L}$.

Par exemple, soit la base d'apprentissage E ={(25, bas), (27, élevé), (33, élevé), (41, bas), (42, élevé)}. L'ensemble des classes est {bas, élevé} et $\{25, 27, 33, 41, 42\}$ est l'ensemble de valeurs pour l'attribut en question (par exemple, l'attribut "âge"). L'alphabet associé à cette base est $\mathcal{L} = \{b, e\}$, en considérant que *bas* est associé à la lettre b et que *élevé* est associé à la lettre e. La base $\mathcal{E}$ considérée pour l'attribut en question permet donc de déduire le mot *beebe* sur $\mathcal{L}^*$.

Pour modéliser les opérateurs de la morphologie mathématique dans la théorie des langages formels, il est nécessaire d'introduire une lettre particulière en supplément des lettres associées aux classes de la base. Cette lettre supplémentaire est censée représenter le *fond* de l'espace où se situent les corps morphologiques, elle autorisera le marquage des zones susceptibles d'apparaître après une érosion ou de disparaître après une dilatation et qui ne peuvent pas être associées à une classe de la base. Cette lettre, u, permet la construction de l'alphabet $\mathcal{L}_u = \mathcal{L} \cup \{u\}$ avec $u \notin \mathcal{L}$. La lettre u a encore une signification particulière dans le système car elle sera aussi utilisée pour marquer les séquences qui seront dites *incertaines*, c'est-à-dire, les séquences qui, après le processus de filtrage d'un mot issu d'une base d'apprentissage, ne peuvent être associées à aucune classe connue.

Il faut noter que l'inconvénient de transcrire de cette façon une base d'apprentissage est qu'une information importante sur les données est perdue : l'écart entre les valeurs successives. Ainsi, deux bases structurellement différentes peuvent donner le même mot. Par exemple, dans la Figure 4.7, les deux bases apparaissent bien différentes, pourtant,

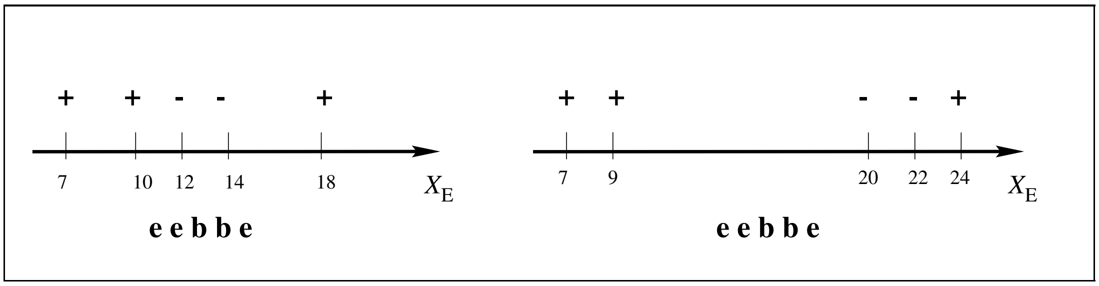


Figure 4.7 – *Un même mot pour deux bases d'apprentissage*

elles donnent le même mot[7].

6. Pour tout rappel sur la théorie des langages formels, consulter l'annexe A.

7. La classe *bas* est représentée sur le graphique par $-$ et est associée à la lettre b et la classe *élevé* est représentée par $+$ est associée à la lettre e.

Pour éviter cela, nous introduisons une lettre neutre dans le langage, notée par exemple la lettre n, pour marquer les écarts entre les éléments de la base d'apprentissage. L'écart entre chaque élément de base est alors mesuré en fonction d'un écart de base tel que l'écart minimum entre deux valeurs présentes dans la base . Ensuite, entre deux éléments de la base, la lettre n est insérée autant de fois que l'écart entre les deux éléments est supérieur à l'écart de base.

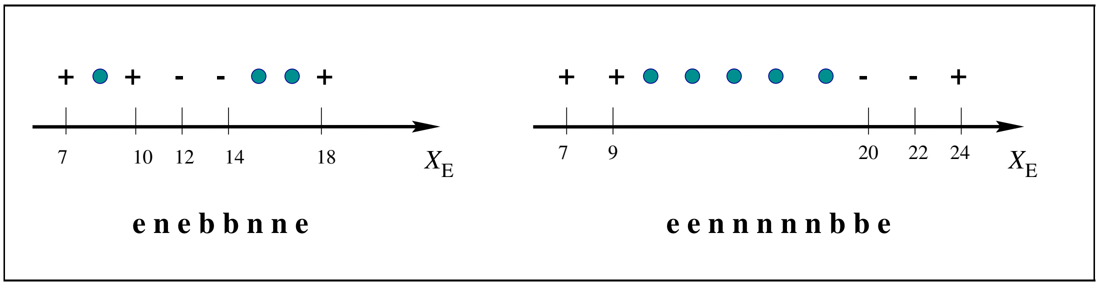


Figure 4.8 – *Deux mots différents pour deux bases différentes*

Par exemple, dans la Figure 4.8, l'écart de base calculé est 2 et la lettre n, représentée graphiquement par un rond, est insérée une fois entre 7 et 10 et deux fois entre 14 et 18 pour la première base d'apprentissage. Pour la seconde base, l'écart de base est encore 2 et la lettre n est insérée cinq fois entre 9 et 20.

La nouvelle lettre ainsi introduite est considérée d'une façon différente des autres. Elle représente une zone où la classification est inconnue à la différence de la lettre u qui représente les zones incertaines ou des autres lettres qui représentent une classe existante.

Comme il a été précisé en introduction de cette section, les mots d'un langage ainsi définis, les transformations morphologiques vont être représentées par des transductions qui agiront sur le mot induit par la base d'apprentissage.

Une transduction pour éroder

L'opérateur d'érosion transposé dans la théorie des langages formels peut être défini par la transduction Er_x avec[8] $\mathcal{A} = \mathcal{B} = L_u$ (*cf.* Figure 4.9). Dans cette figure, les états de l'automate sont numérotés. L'état initial, d'où débute l'érosion d'un mot, est l'état numéro 1. L'état terminal où l'automate s'arrête à la fin d'une érosion est l'état numéro 4. Les transitions sont libellées par deux parties *entrée* | *sortie* : l'entrée qui est lue, et la sortie qui est écrite. Quand l'entrée est lue, l'automate écrit la sortie et change d'état. Il existe deux caractères spéciaux dans cette transduction la lettre \$ qui marque la fin d'un mot et qui permet de savoir quand il n'y a plus de lettres à lire, et la lettre ε qui est la lettre vide qui signale que rien n'est à écrire en sortie.

La transduction de la Figure 4.9 permet l'érosion d'un mot avec la lettre $x \in \mathcal{L}$ comme élément structurant. Elle permet la réduction des séquences de x dans le mot. Du mot v, $v \in \mathcal{L}_u^*$, on obtient le mot $Er_x(v) \in \mathcal{L}_u^*$. Par exemple, le mot $v = beeee$ produit $Er_b(beeee) = ueeee$ et $Er_e(beeee) = bueeu$.

8. Les notations utilisées font référence à celles présentées en Annexe A.

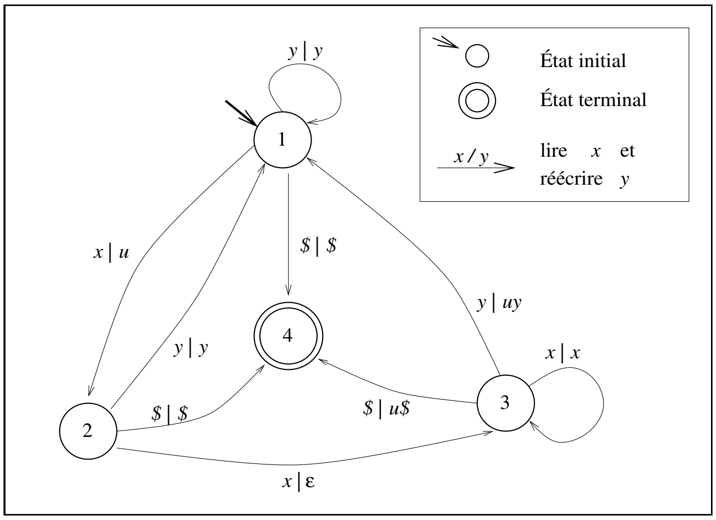


Figure 4.9 – *Transduction Er_x pour l'érosion par $x \in \mathcal{L}$ de $y \in \mathcal{L}_u$, $y \neq x$*

Il est important de vérifier que l'utilisation de ce système de réécriture est correcte, c'est-à-dire, d'une part que cette transduction donne un irréductible pour tout mot du langage donné en entrée et, d'autre part, que le mot obtenu peut bien être considéré comme une version érodée du mot donné en entrée. C'est donc cette preuve de la correction de cet automate pour réaliser cet érosion que nous présentons maintenant.

L'existence d'un irréductible pour tout mot du langage s'énonce sous la forme du théorème suivant :

Théorème 4.1 *Pour tout mot v de $\mathcal{L}_u^*$, pour toute lettre x de $\mathcal{L}$, le mot érodé $Er_x(v)$ obtenu à partir de v existe et est unique.*

Pour prouver ce théorème, il suffit de montrer que le système de réécriture défini par la transduction est confluent. Cela se fait aisément par l'utilisation de techniques classiques de la théorie des langages formels : il suffit de prouver que le système est nœthérien et local confluent.

Preuve du théorème 4.1

∘ Le système Er_x est nœthérien.

À partir de chacun des états non terminaux (états 1, 2 et 3), l'état terminal (l'état 4) est atteint si la fin du mot est lue ($) et, de plus, la réécriture s'arrête obligatoirement une fois l'état terminal atteint car aucune transition n'est alors possible. Par conséquent, pour tout mot fini, il n'existe pas de séquence infinie de réécriture et le système défini par cette transduction est donc bien nœthérien.

∘ Le système Er_x est confluent.

Pour tout mot de $\mathcal{L}_u^*$, il n'existe pas deux réécritures différentes par cette transduction. En effet, cette transduction est un automate déterministe, à chaque état il n'existe qu'une et une seule transition possible correspondante à une lettre donnée. Il n'y a donc pas d'ambiguïté, pour tout mot de $\mathcal{L}_u^*$, et pour toute position dans ce mot, il ne peut donc pas exister deux dérivations différentes. Par conséquent, pour tout triplet de mots de $\mathcal{L}_u^*$, u, v et w tels que $u \longrightarrow v$ et $u \longrightarrow w$, il est alors clair que $v = w$. Le système étant nœthérien, il existe aussi un mot t tel que $v \xrightarrow{*} t$ (et donc $w \xrightarrow{*} t$).

On en déduit que Er_x est local confluent.

∘ Er_x étant nœthérien et local confluent, le théorème A.3 permet de déduire qu'il est confluent et que, de plus, grâce au théorème A.2, aucun mot ne possède plus d'un irréductible.

Le théorème 4.1 est donc prouvé.

⋄

Maintenant, il faut vérifier que ce système de réécriture agit d'une façon équivalente à une érosion sur les mots donnés en entrée, sachant que l'érosion d'un mot doit s'appliquer sur les bords de ce mot et transformer ses deux lettres extrêmes en lettres incertaines. Pour cela, il faut vérifier le théorème suivant.

Théorème 4.2 *Toute séquence complète x^{n+2} $(n > 0)$ d'un mot v de $\mathcal{L}_u^*$ se réécrit ux^nu dans $Er_x(v)$.*

Preuve du théorème 4.2

∘ Si $v = x^{n+2}w$ avec $w \in \mathcal{L}_u^*$, la réécriture de v commence de l'état 1 à l'état 2, x est lu et remplacé par u, puis de l'état 2 à l'état 3, x est lu et remplacé par ε, puis les x^n restants sont lus et remplacés par x^n, la transduction demeurant dans l'état 3.

La séquence x^{n+2} est supposée complète, donc après l'avoir lue, et de l'état 3, soit la fin du mot est atteinte et un u est écrit, soit une lettre y est lue et uy est écrit. Dans tous les cas, il est clair que $Er_x(v) = Er_x(x^{n+2}w) = ux^nuEr_x(w)$

∘ Si $v = tx^{n+2}w$ avec $w, t \in \mathcal{L}_u^*$, étant donné que x^{n+2} est une séquence complète, il est clair que, après la réécriture de t, la transduction se trouve dans l'état 1 et il apparaît que $Er_x(v) = Er_x(tx^{n+2}w) = Er_x(t)Er_x(x^{n+2}w)$ et, d'après le cas précédent, $Er_x(v) = Er_x(t)ux^nuEr_x(w)$.

Ainsi, le théorème 4.2 est donc prouvé.

⋄

La définition de la transduction réalisation une dilatation sur des mots s'effectue d'une manière équivalente.

Une transduction pour dilater

Comme l'opérateur d'érosion, l'opérateur de dilatation transposé dans la théorie des langages formels peut être défini par une transduction Di_x sur $\mathcal{A} = \mathcal{B} = L_u$ avec $x \in \mathcal{L}$ (*cf.* Figure 4.10). Cette transduction autorise la dilatation d'un mot avec la lettre $x \in \mathcal{L}$ comme élément structurant. Elle a pour effet de *dilater* une séquence de plusieurs mêmes lettres dans un mot entourée par la lettre u.

Comme dans le cas de l'érosion, il est facile de montrer que tout mot peut subir une dilatation.

Théorème 4.3 *Pour tout mot v de $\mathcal{L}_u^*$, pour toute lettre x de $\mathcal{L}$, le mot dilaté $Di_x(v)$ obtenu à partir de v existe et est unique.*

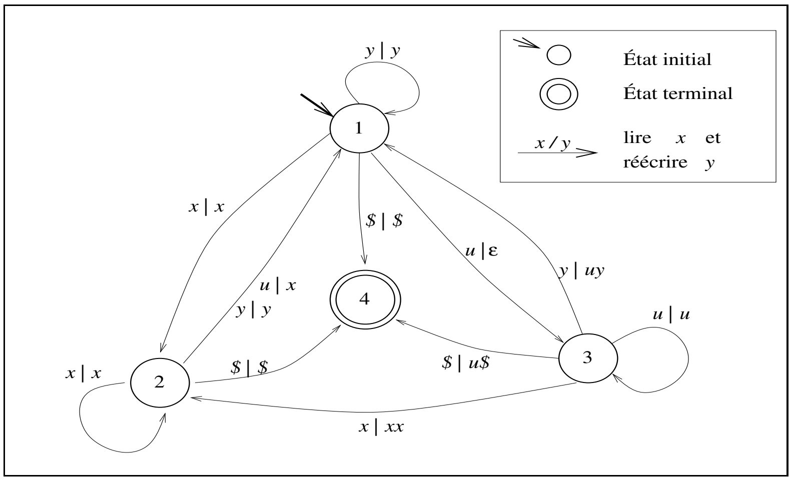


Figure 4.10 – *Transduction Di_x pour la dilatation par $x \in \mathcal{L}$ de $y \in \mathcal{L}_u$, $y \neq x$*

Preuve du théorème 4.3

La preuve est identique à celle donnée pour le théorème 4.1.

$\diamond$

Comme dans le cas de l'érosion, il faut prouver que la transformation définie par cette transduction réalise bien une transformation proche d'une dilatation. La dilatation d'un mot doit lui permettre de s'étendre sur les lettres incertaines qui l'entourent. Donc, il faut prouver le théorème suivant :

Théorème 4.4 *Toute séquence ux^nu $(n > 0)$ d'un mot v de $\mathcal{L}_u^*$ se réécrit x^{n+2} dans $Di_x(v)$.*

Preuve du théorème 4.4

$\circ$ (1) Si $v = ux^nuw$ avec $w \in \mathcal{L}_u^*$, la réécriture de v commence de l'état 1 à l'état 3, u est lu et remplacé par ε, puis de l'état 3 à l'état 2, x est lu et remplacé par xx, puis les x^{n-1} restants sont lus et remplacés par des x^{n-1}, la transduction demeurant dans l'état 2 (si $n = 1$, rien ne change). Ensuite, u est lu et remplacé par x et la transduction passe dans l'état 1 à partir duquel la dilatation se poursuit par le mot w. Il est alors clair que $Di_x(v) = Di_x(ux^nuw) = xxx^{n-1}xDi_x(w)$ soit $Di_x(v) = x^{n+2}Di_x(w)$.

$\circ$ (2) Si $v = tux^nuw$ avec $w, t \in \mathcal{L}_u^*$. Si t a pour dernière lettre un y, alors après la dilatation de t, la transduction se trouve dans l'état 1 (la destination de toutes les transitions libellées par y) et $Di_x(v) = Di_x(t)Di_x(ux^nuw)$.

Si t a pour dernière lettre un x, alors après la dilatation de t, la transduction se trouve dans l'état 2 (la destination de toutes les transitions libellées par x). Par la suite, la lettre u est remplacée par un x et la transduction passe dans l'état 1, puis un x est lu, est remplacé par x et la transduction passe dans l'état 2 où les x^{n-1} sont lus et remplacés par des x^{n-1} en restant dans l'état 2. Finalement, le u final est remplacé par un x et la transduction passe dans l'état 1, ce qui donne alors $Di_x(v) = Di_x(tux^nuw) = Di_x(t)xxx^{n-1}xDi_x(w)$ soit $Di_x(v) = Di_x(t)x^{n+2}Di_x(w)$.

Si t a pour dernière lettre un u, alors après la dilatation de t, la transduction se trouve dans l'état 1 ou dans l'état 3 (les deux seules destinations des transitions libellées par u). Si la transduction est dans l'état 1, il est clair que $Di_x(v) = Di_x(tux^n uw) = Di_x(t)Di_x(ux^n uw)$ et le cas a déjà été examiné. Si la transduction est dans l'état 3, la lettre u à lire est remplacée par un u et la transduction reste dans l'état 3, et la situation est équivalente à celle examinée dans le cas (1). Ainsi, encore une fois, $Di_x(v) = Di_x(t)x^{n+2}Di_x(w)$.

Le théorème 4.4 est ainsi prouvé.

◇

Il apparaît donc que les deux transductions présentées permettent d'implémenter des opérations d'érosion et de dilatation sur des mots d'un langage. Comme les opérateurs de la morphologie mathématique, ces systèmes de réécriture agissent relativement à un élément structurant qui est ici une lettre particulière choisie.

Pour la suite, pour tout mot $v \in \mathcal{L}_u^*$, le mot $Er_x^n(v)$ (resp. $Di_x^n(v)$), avec $n > 0$, représentera le mot réécrit à partir de v après n érosions (resp. dilatations) consécutives.

Maintenant, comme pour la théorie de la morphologie mathématique, il est intéressant d'utiliser les opérateurs d'érosion et de dilatation pour définir les deux opérateurs d'ouverture et de fermeture. Finalement, un opérateur de filtrage pourra alors être défini à l'image des opérateurs de filtrage de la théorie de la morphologie mathématique.

Les opérateurs d'ouverture et de fermeture

Une ouverture est la composition $Di_x \circ Er_x$ des deux opérateurs précédents. La n-ouverture ($n \in \mathbb{N}$) d'un mot $v \in \mathcal{L}_u^*$ par la lettre $x \in \mathcal{L}$ est définie par $Op_x^n(v) = Di_x^n(Er_x^n(v))$. La n-ouverture d'un mot permet d'effacer les petites séquences avec une longueur, en lettres, inférieure à $2n$. L'avantage de cet opérateur est qu'il permet de gommer toutes les séquences de v avec une longueur inférieure à une valeur fixée. Par exemple, avec le mot $v = eebbbebeb$, on a $Op_b^1(v) = eebbbeueu$, $Op_e^1(v) = uubbbubub$ et $Op_b^2(v) = Di_l^2(Er_b^2(v)) = Di_b^2(eeuuueueu) = eeuuueueu$.

La fermeture est la composition $Er_x \circ Di_x$ des deux opérateurs d'érosion et de dilatation. Cette composition permet de rassembler en un seul bloc des séquences de moins de 2 lettres u. La n-fermeture ($n \in \mathbb{N}$) par la lettre $x \in \mathcal{L}$ du mot $v \in \mathcal{L}_u^*$ est définie par $Cl_x^n(v) = Er_x^n(Di_x^n(v))$. Avec cet opérateur, les séquences de $2n$ lettres u sont donc unifiées. Par exemple, à partir du mot $v = uueeuuueeueuuu$, on obtient $Cl_e^1(v) = uueeuuueeeeuuu$ et $Cl_e^2(v) = uueeeeeeeeeuuu$.

Le filtre alterné

Un filtre est la composition des deux opérateurs d'ouverture et de fermeture. Soit $v \in \mathcal{L}_u^*$, $x \in \mathcal{L}$ and $n \in \mathbb{N}$. Le n-filtrage par la lettre x du mot v est défini par :

$$\begin{array}{ll} \text{si } n = 1 & Fil_x^1(v) = Cl_x^1(Op_x^1(v)) \\ \text{si } n > 1 & Fil_x^n(v) = Cl_x^n(Op_x^n(Fil_x^{n-1}(v))) \end{array}$$

Cette combinaison particulière de ces opérateurs amène d'intéressantes propriétés. Un filtre permet d'*estomper*[9] un mot. Dans un premier temps, les séquences de moins de

9. Traduction du terme anglais *fuzzify*.

$2n$ lettres sont effacées (*ie.* remplacées par des lettres u), ensuite, les séquences séparées par $2n$ lettres sont unifiées.

4.3.4 Un nouvel algorithme automatique pour construire une partition floue

Lorsque l'opérateur de filtrage est appliqué sur un mot induit par une base d'apprentissage, les petites séquences sont traduites en séquences incertaines et les séquences de même type sont regroupées.

Un n-filtre permet donc d'estomper le mot correspondant à une base d'apprentissage. Ainsi, un mot avec des séquences importantes (d'une taille supérieure à $2n$) est obtenu. Dans l'application de construction d'arbres de décision flous Salammbô présentée au chapitre 2, la valeur de n n'est pas choisie de façon fixe, c'est la plus grande valeur qui laisse des séquences de lettres non incertaines. Après chaque application d'un filtre de degré n, s'il ne reste plus de séquence certaine, le degré du filtrage est réduit de un, sinon le filtrage se poursuit avec $n + 1$.

Les séquences de u représentent les zones *incertaines* où les classes sont très mélangées. Les séquences de lettres $x, x \in \mathcal{L}$ sont des séquences associées à une classe unique, du moins, globalement, elles sont appelées : séquences *certaines*, quelle que soit $x, x \neq u$. Ces deux types de séquences sont utilisées pour construire la partition floue associée à l'attribut en question, pour la base d'apprentissage $\mathcal{E}$. Les séquences *certaines* d'une lettre x correspondent aux noyaux des sous-ensembles flous de la partition.

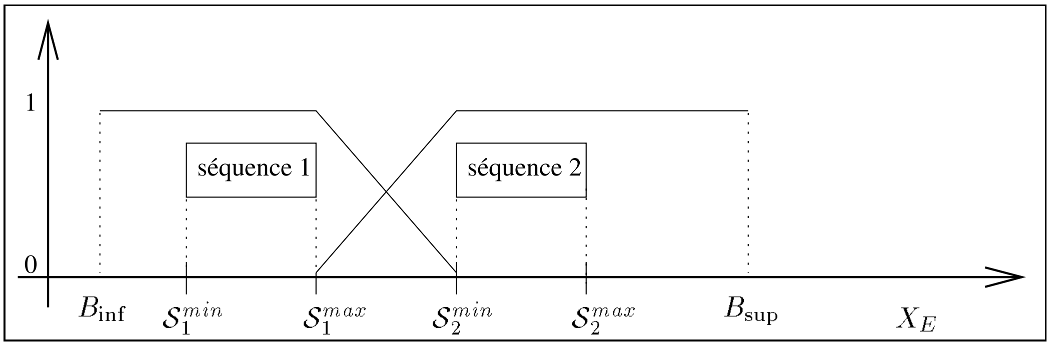


Figure 4.11 – *Construction d'une partition floue*

Soit r le nombre de modalités floues à obtenir pour l'attribut considéré. Les r plus grandes séquences certaines ne contenant qu'une seule classe sont choisies. Elles sont associées à des intervalles flous de X_E, par exemple $[\mathcal{S}_1^{min}, \mathcal{S}_1^{max}]$ et $[\mathcal{S}_2^{min}, \mathcal{S}_2^{max}]$ quand $r = 2$. Dans le cas où r séquences n'existent pas, il faut réduire soit le nombre de modalités voulues, soit le nombre de filtrages à appliquer. Pour $r = 2$, ce processus est résumé par l'algorithme FPMM[10] (Marsala et Bouchon-Meunier, 1996) :

10. Fuzzy Partioning using Mathematical Morphology. Cet algorithme porte un nom anglais parce que sa première apparition (Marsala et Bouchon-Meunier, 1996) a eu lieu dans une conférence américaine.

Algorithme 4.1 (FPMM) *Pour inférer une partition floue en deux modalités d'un attribut de la base d'apprentissage $\mathcal{E}$ défini sur X_E*

1. *Traduire $\mathcal{E}$ en un mot v.*
2. *Estomper v en faisant varier n jusqu'à la plus grande valeur qui laisse des séquences certaines dans le mot.*
3. *Chercher les deux plus grandes séquences certaines $\mathcal{S}_1$ et $\mathcal{S}_2$.*
4. *Soit $\mathcal{S}_i^{min}$ (resp. $\mathcal{S}_i^{max}$) la valeur associée à la première (resp. dernière) lettre de $\mathcal{S}_i$ dans X, et soit $\mathcal{S}_1 \equiv [\mathcal{S}_1^{min}, \mathcal{S}_1^{max}]$ (resp. $\mathcal{S}_2 \equiv [\mathcal{S}_2^{min}, \mathcal{S}_2^{max}]$) avec $\mathcal{S}_1^{max} < \mathcal{S}_2^{min}$. La partition floue en deux modalités est définie par deux sous-ensembles flous. Le premier a pour noyau $]B_{inf}, \mathcal{S}_1^{max}]$ et pour support $]B_{inf}, \mathcal{S}_2^{min}]$, et le second a pour noyau $[\mathcal{S}_1^{max}, B_{sup}[$ et pour support $[\mathcal{S}_2^{min}, B_{sup}[$*

Généralement, les noyaux de ces sous-ensembles flous sont étendus à $[B_{\text{inf}}, B_{\text{sup}}]$ (Figure 4.11) mais les valeurs appartenant effectivement aux séquences trouvées peuvent être conservées telles quelles (Figure 4.12).

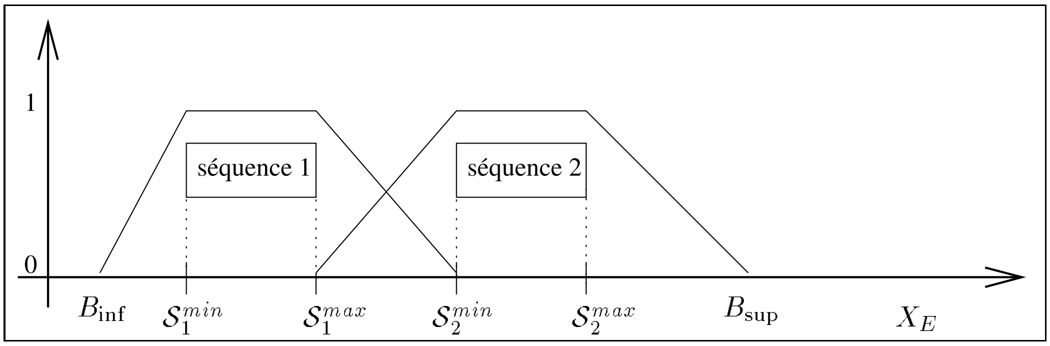


Figure 4.12 – *Une autre partition floue possible*

4.4 Extension de l'algorithme pour la prise en compte de données floues

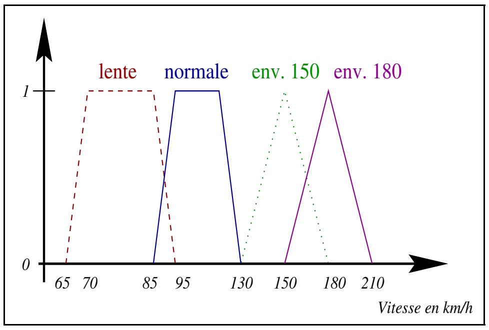
En présence de données numériques-symboliques, c'est-à-dire de valeurs floues pour un attribut dans la base d'apprentissage, ou alors de classes floues, il convient d'adapter les opérateurs d'érosion et de dilatation pour prendre en compte ce type de données. L'association d'une valeur floue à chaque donnée est comme l'ajout d'une dimension supplémentaire dans l'espace des données.

Par exemple, dans le cas de la base donnée Table 4.1, chaque moto est définie par sa vitesse moyenne et est associée à une classe de prix. L'attribut *Vitesse moyenne* peut

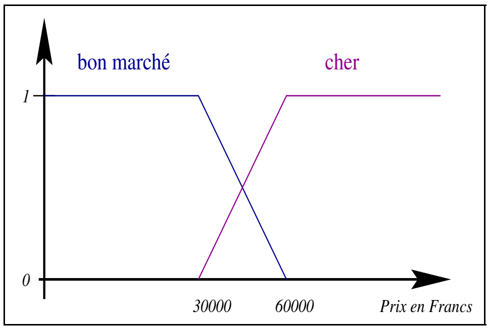
Moto	*500 GSE*	*CBR 1100*	*125 Rebel*	*900 Bandit*
Vitesse Moyenne	normale	environ 150	lente	environ 180
Classe Bon marché	0.7	0.25	1.0	0.5
Classe Cher	0.3	0.75	0.0	0.5

Table 4.1 – *Exemple de base avec des données floues*

prendre des valeurs floues, comme *environ 150 km/h* ou *lente*, dont les sous-ensembles correspondants sont donnés dans la Figure 4.13. La classe de prix est, elle aussi, à valeurs floues, et un degré d'appartenance à chaque valeur est donné pour chaque exemple.



Modalités d'attribut floues



Classe à modalités floues

Figure 4.13 – *Données floues dans la base d'apprentissage*

Il existe deux façons de traiter ce type de données. Une première solution est d'utiliser ces valeurs comme autant de valeurs floues disponibles. L'avantage de cette façon de procéder est que l'algorithme de construction d'arbres de décision peut s'appliquer tel quel, sans aucun prétraitement des données. Dans ce cas, il peut être délicat d'apprendre à partir de tels sous-ensembles flous dans la mesure où ils représentent des sous-ensembles flous trop spécialisés pour arriver à en déduire des concepts suffisamment généraux. De plus, la multiplication des sous-ensembles flous que cela engendrerait serait grandement néfaste au processus d'apprentissage, comme l'est un ensemble de valeurs numériques qui est utilisé sans être discrétisé.

Par exemple, avec la base donnée par la Table 4.1, cela revient à considérer les quatre valeurs floues de la vitesse comme quatre valeurs indépendantes. À la suite de l'apprentissage, un test sur la valeur de cet attribut aurait quatre issues possibles, correspondantes chacune à une valeur floue. L'inconvénient de cette approche peut être que les valeurs floues utilisées, comme par exemple *environ 180*, ne soient pas des valeurs suffisamment générales.

Une autre solution est de construire, pour l'attribut A_j et pour chaque modalité c_k de la classe, un sous-ensemble flou μ_{c_k} correspondant à la totalité des données de la base d'apprentissage. Ce sous-ensemble flou, défini sur l'univers X_j des valeurs de l'attribut

A_j, peut être alors considéré comme une nouvelle modalité v_{c_k} de cet attribut ; modalité qui combine toutes les données de la base d'apprentissage concernant l'attribut A_j et la classe c_k.

Pour construire cette nouvelle modalité floue v_{c_k}, il faut *agréger* toutes les modalités numériques-symboliques v_{jl} de l'attribut A_j présentes dans la base d'apprentissage. De plus, pour tenir compte de la classe c_k, il faut *pondérer* chaque modalité v_{jl} par le degré d'appartenance à la classe qui lui est associé. Cette construction de la nouvelle modalité v_{c_k} est présentée dans la première partie de cette section.

Cet ensemble de modalités v_{c_k} ainsi construites sur X_j peut être utilisé tel quel pour libeller les arcs sortant d'un nœud testant la valeur de cet attribut. Mais, il faut reconnaître que, dans ce cas, les modalités v_{c_k} reflètent exactement les données de la base d'apprentissage, et elles ont tendance à être par trop spécialisées, en particulier, en présence de données numériques dans la base d'apprentissage

La deuxième partie de cette section, des méthodes sont alors présentées pour généraliser les modalités v_{c_k} dans la même optique que celle développée par l'algorithme FPMM. Cette généralisation s'effectue en lissant les sous-ensembles flous, toujours à l'aide des concepts de la théorie des morphologie mathématique.

C'est cette manière de procéder que nous proposons et que nous allons développer par la suite.

4.4.1 Agrégation des données floues

Comme il est précisé dans l'introduction de cette section, dans un premier temps, les degrés d'appartenance de chaque exemple aux classes mais aussi aux valeurs de l'attribut en question sont agrégés afin d'obtenir un seul sous-ensemble flou pour chaque classe.

Considérons un attribut A_j, d'univers de valeurs X_j, et dont m_j modalités numériques-symboliques v_{jl}, de fonction d'appartenance $\mu_{v_{jl}}$, sont utilisées dans la base d'apprentissage $\mathcal{E}$. Pour chaque exemple e_i de $\mathcal{E}$, la modalité de A_j qui lui est associée est notée $e_i(A_j)$. De plus, chaque exemple e_i possède un degré d'appartenance $\delta_{c_k}(e_i)$ à chaque classe c_k de la base.

Pour une même modalité v_{jl} de A_i, il peut exister plusieurs exemples e_i dans $\mathcal{E}$ possédant cette modalité, chaque exemple pouvant posséder un degré $\delta_{c_k}(e_i)$ différent de celui des autres exemples. On note $\gamma_{c_k}(v_{jl})$ le degré d'appartenance à la classe c_k, associé à la modalité v_{jl}, déduit de tous les exemples possédant cette modalité :

$$\gamma_{c_k}(v_{jl}) = \max_{\{e_i \mid e_i(A_j)=v_{jl}\}} (\delta_{c_k}(e_j))$$

Pour calculer le degré d'appartenance $\gamma_{c_k}(v_{jl})$ de la modalité v_{jl}, les degrés d'appartenance à la classe c_k de tous les exemples possédant cette modalité sont agrégés en utilisant une t-conorme (comme le *maximum*) afin de rendre compte de l'union de tels degrés entre eux.

Le degré $\gamma_{c_k}(v_{jl})$ permet de modifier la fonction d'appartenance $\mu_{v_{jl}}$ de la modalité v_{jl} en fonction des degrés d'appartenance aux classes observés pour les exemples ayant cette modalité. Cette modification permet d'obtenir une nouvelle fonction d'appartenance

μ_{v_{jl},c_k} caractérisant l'occurrence simultanée de la modalité v_{jl} et de la classe c_k avec le degré $\gamma_{c_k}(v_{jl})$ correspondant.

L'agrégation s'effectue grâce à l'utilisation d'un opérateur d'intersection (une t-norme comme, par exemple, le *minimum*) :

$$\forall x \in X_j,\ \mu_{v_{jl},c_k}(x) = \min(\mu_{v_{jl}}(x), \gamma_{c_k}(v_{jl}))$$

La t-norme permet de rendre compte de l'agrégation naturelle qui est faite : *x appartient à la modalité floue v_{jl}* **et** *à l'exemple qui a pour degré d'appartenance $\delta_{c_k}(e_i)$ à la classe c_k.*

Par exemple, avec les données de la Table 4.1, l'attribut A_j est la vitesse définie sur $X_j = [0, 250]$. Il possède 4 modalités floues : *lente*, *normale*, *environ 150* et *environ 180*. Pour la modalité *normale*, il n'y a qu'un seul exemple, noté *500 GSE*, avec cette modalité de degrés d'appartenance $\mu_{\text{bon marché}}(\textit{500 GSE}) = 0.7$ et $\mu_{\text{cher}}(\textit{500 GSE}) = 0.3$. Par conséquent, on a $\gamma_{\text{bon marché}}(\text{normale}) = 0.7$ et $\gamma_{\text{cher}}(\text{normale}) = 0.3$. La figure 4.14 donne alors la fonction d'appartenance $\mu_{\text{normale, bon marché}}$ résultant de l'agrégation de la classe et de la modalité.

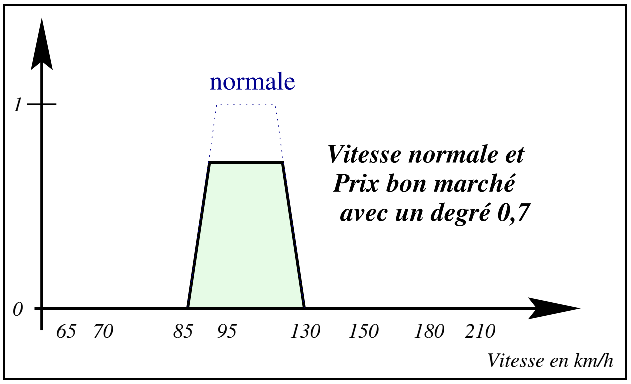


Figure 4.14 – *Exemple de calcul :* $\mu_{\textit{normale, bon marché}}$

Finalement, il reste à agréger tous les degrés μ_{v_{jl},c_k} de toutes les modalités v_{jl} ainsi calculés pour obtenir une nouvelle modalité v_{c_k} pour l'attribut A_j. Cette modalité a pour fonction d'appartenance μ_{c_k} qui reflète le degré d'appartenance à la classe c_k pour l'ensemble de la base d'apprentissage, relativement à l'attribut A_j. Ce sous-ensemble flou se déduit grâce à l'utilisation d'un opérateur d'union (une t-conorme comme, par exemple, le *maximum*) :

$$\forall x \in X_j,\ \mu_{c_k}(x) = \max_l(\mu_{v_{jl},c_k}(x))$$

La t-conorme permet de rendre compte de l'union de toutes les modalités floues présentes dans la base. Le résultat d'une telle agrégation de sous-ensembles flous et de la classe étudiée, par la méthode proposée, permet de déduire un sous-ensemble synthétisant toutes ces données.

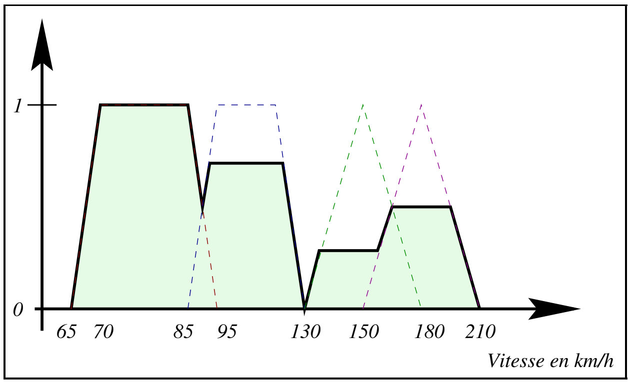


Figure 4.15 – *Union des données floues attachées à la classe* bon marché

Dans le cadre de l'exemple, la figure 4.15 donne le résultat d'une telle agrégation pour la classe *bon marché*.

Pour faciliter le traitement de ce sous-ensemble flou, il est utilisé sous la forme d'une liste de points associés chacun à un degré d'appartenance. La distance entre chaque point dépend du pas d'itération que l'on choisit de prendre. Plus il sera petit, et plus la finesse du traitement sera importante. Par exemple, le sous-ensemble flou de la figure 4.15, en prenant un pas de 10, pourra être utilisé sous la forme des 16 points suivants :

point 1	*point 2*	*point 3*	*point 4*	*point 5*	*point 6*	*point 7*	*point 8*
60	70	80	90	100	110	120	130
0	1	1	0.5	0.7	0.7	0.35	0

point 9	*point 10*	*point 11*	*point 12*	*point 13*	*point 14*	*point 15*	*point 16*
140	150	160	170	180	190	200	210
0.25	0.25	0.4	0.5	0.5	0.5	0.25	0

Table 4.2 – *Le sous-ensemble flou sous la forme d'une liste de points*

Par la méthode présentée, il est donc possible de définir, pour chaque classe de la base, un sous-ensemble flou sur l'ensemble des valeurs de l'attribut A_j censé représenter l'ensemble des valeurs de la base d'apprentissage. Cependant, comme les données brutes de la base, ce sous-ensemble flou est encore trop spécialisé pour être utilisé avec profit en apprentissage. Sa forme est encore trop complexe pour être suffisamment générale. Comme dans le cadre non flou, un algorithme de filtrage doit être mis au point pour lisser ce sous-ensemble flou et le rendre plus général.

Dans la suite de cette section, deux tels algorithmes de filtrage sont présentés. Le premier repose sur une simple utilisation de l'algorithme FPMM présenté dans la section 4.3.4 mais il souffre de l'inconvénient de ne pas tenir compte du supplément d'information apportée par les modalités floues. Le second algorithme, lui, repose sur une extension de l'algorithme FPMM pour la prise en compte de telles modalités floues. Il tire profit de toute l'information apportée par les modalités floues et paraît en ce point préférable.

4.4.2 Une méthode utilisant l'algorithme FPMM

La première méthode pour utiliser de tels sous-ensembles flous est de considérer qu'ils induisent directement des mots. Ainsi, les éléments de degrés d'appartenance à une classe égaux à 1 se traduisent par la lettre correspondant à cette classe. Les éléments dont aucun degré d'appartenance n'est égal à 1 se traduisent, eux, par la lettre u. Cette traduction de la base d'apprentissage est unique si l'on considère que, pour chaque élément de la base, il n'existe qu'au plus une classe pour laquelle le degré d'appartenance est 1.

Par exemple, le mot donné par la Table 4.2 est alors *baauuuubuuuuuuub*, en considérant que la classe *bon marché* est associée à la lettre a et que la classe *cher* est associée à la lettre b.

Ce mot peut alors être utilisé dans l'algorithme FPMM pour inférer une partition floue, avec les opérateurs d'érosion et de dilatation définis dans le cadre non flou.

L'intérêt de cette méthode est sa facilité de mise en œuvre et le fait qu'elle autorise l'utilisation de l'algorithme FPMM et des transductions définies précédemment, sans aucune modification supplémentaires. Son inconvénient est, par contre, de ne pas tenir suffisamment compte des degrés d'appartenance existants. Ainsi, un élément dont tous les degrés d'appartenance aux classes sont inférieurs à 1 sera traduit par la lettre u, quelles que soient les valeurs effectives de ces degrés, c'est-à-dire que son traitement est équivalent que ses degrés soient égaux à 0.1 ou à 0.9. C'est typiquement une façon non floue de procéder, et par conséquent, une façon peu élégante puisqu'elle induit une grande perte d'information disponible par ailleurs avec les degrés d'appartenance.

4.4.3 Une méthode utilisant une extension de l'algorithme FPMM

Une autre solution se doit de tenir compte des degrés d'appartenance aux classes associées aux éléments de la base. L'érosion et la dilatation à utiliser doivent réagir aux différences d'appartenance susceptibles d'exister, qui varient entre 0 et 1. Ces opérations morphologiques vont altérer les degrés d'appartenance associés de chaque élément, d'une façon douce et sensible aux degrés d'appartenance des éléments voisins.

Ainsi, tous les éléments auront à subir les effets d'une érosion et d'une dilatation et non plus seulement ceux qui se situent aux frontières des séquences. En substance, tous les éléments sont des frontières de séquence. C'est donc un enrichissement conséquent des transductions données au paragraphe 4.3.3. En effet, celles-ci s'attachaient à reconnaître les lettres qui séparent des séquences pour leur appliquer une transformation morphologique. Or maintenant, toutes les lettres vont avoir la même signification dans le processus morphologique et subiront une modification dans leur degré d'appartenance dépendante, d'une manière équivalente, des degrés des éléments voisins et indépendante de la valeur même de la lettre.

Formalisation des processus de transformation

Soit α_i le degré d'appartenance de l'élément e_i avant une opération morphologique et α'_i son degré d'appartenance après cette opération. L'étude des transductions du paragraphe 4.3.3 permet de remarquer que la transformation morphologique que subit un point e_i relativement à une classe prend en compte les classes associées aux points e_{i-1} et e_{i+1} qui l'entourent directement.

Ainsi, la nouvelle valeur α'_i ne dépend que des valeurs α_i, α_{i-1} et α_{i+1} et l'opération morphologique est par conséquent une fonction $f : [0,1] \times [0,1] \times [0,1] \longrightarrow [0,1]$ telle que $\alpha'_i = f(\alpha_{i-1}, \alpha_i, \alpha_{i+1})$.

Dans ce cas là, on peut remarquer qu'il n'y a plus de distinction entre les lettres. Ce sont les degrés d'appartenance à une classe qui seuls interviennent dans le processus de transformation. Deux solutions sont alors envisageables pour formaliser le processus de transformation. La première est de construire une simple transduction et la deuxième est d'utiliser un outil plus puissant que les automates, comme les grammaires contextuelles, pour une telle formalisation. Une grammaire contextuelle est un système de réécriture qui tient compte du contexte d'une lettre pour la réécrire (pour plus de détails, consulter l'Annexe A).

Une nouvelle transduction pour transformer des sous-ensembles flous

La première solution se met en œuvre sous la forme d'une transduction qui reconnaît des lettres associées à un degré d'appartenance et qui réécrit cette même lettre mais en modifiant le degré d'appartenance (Figure 4.16). Dans cet automate, il n'y a qu'une seule

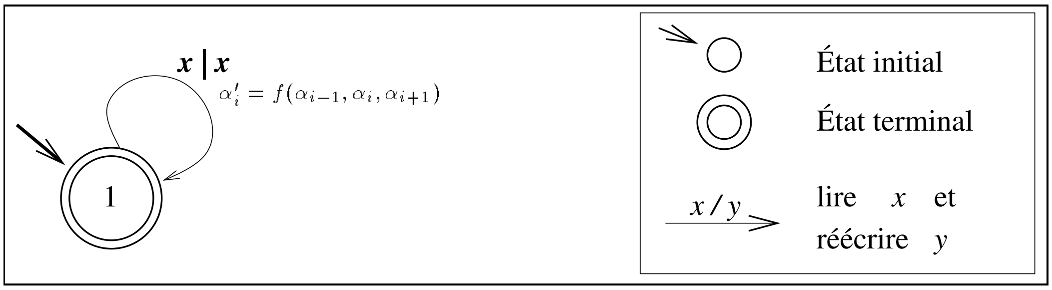


Figure 4.16 – *Transduction pour les opérations morphologiques floues*

transition et qu'un seul état. Quelle que soit la lettre lue en entrée, elle est réécrite en sortie mais son degré d'appartenance est, lui, modifié. Cette modification n'est sensible que sur les degrés des lettres qui entourent la lettre lue. Il est clair qu'il est difficile de trouver un automate plus simple mais il reste peu satisfaisant dans le sens où il laisse apparaître que les lettres qui libellent les transitions sont moins importantes que leur degré d'appartenance. Ainsi, il semblerait donc plus judicieux d'utiliser ces degrés d'appartenance pour libeller directement les transitions de l'automate. Mais dans ce cas, il faut considérer un automate qui mémoriserait les trois dernières lettres rencontrées pour en réécrire une, et ceci, de façon dynamique. À chaque lettre lue, cet automate devrait modifier non seulement son état, mais aussi les trois précédents états rencontrés. C'est une tâche ardue de trouver un tel type d'automate, en supposant qu'il existe.

Un système de réécriture pour transformer des sous-ensembles flous

En fait, il apparaît qu'il est plus immédiat de définir un système de réécriture qui effectue ce travail. C'est ainsi que la deuxième solution se met en œuvre sous la forme d'un système de réécriture dont le langage reconnu est composé par des lettres qui sont les degrés d'appartenance eux-mêmes. Soit l'alphabet des mots composés par les degrés d'appartenance, enrichi de trois caractères spéciaux :

- le caractère "#" marque le début du mot,
- le caractère "\$" marque la fin du mot,
- le caractère "T" est utilisé pour désigner la tête de lecture et, à travers cela, la lettre en cours de réécriture.

Ces caractères permettent de gérer l'ordre des réécritures dans le système de réécriture suivant, défini sur cet alphabet :

Transformation morphologique floue:

$$\begin{array}{lcl} \mathtt{T}\#\alpha_1\alpha_2 & \longrightarrow & f(0,\alpha_1,\alpha_2)\alpha_1\mathtt{T}\alpha_2 \\ \alpha_1\mathtt{T}\alpha_2\alpha_3 & \longrightarrow & f(\alpha_1,\alpha_2,\alpha_3)\alpha_2\mathtt{T}\alpha_3 \\ \alpha_1\mathtt{T}\alpha_2\$ & \longrightarrow & f(\alpha_1,\alpha_2,0) \end{array}$$

Dans ce système, $\forall i$, les $\alpha_i \in [0,1]$ représentent les degrés d'appartenance des éléments dans l'ordre des valeurs croissantes d'attributs. $f(\alpha_{i-1},\alpha_i,\alpha_{i+1}) \in [0,1]$ est la valeur transformée du degré d'appartenance α_i après l'opération morphologique (la forme de la fonction sera donnée par la suite en fonction de la transformation souhaitée).

Avant de réécrire un mot m donné, ce mot est complété par les trois lettres #, \$ et T : $m \longrightarrow \mathtt{T}\#m\$$. Cela permet de pouvoir contrôler la réécriture de façon séquentielle, du début à la fin du mot. Une fois la réécriture terminée, les trois lettres sont effacées d'elles même.

Ce système de réécriture procède comme suit : la lettre α_i qui se positionne juste après la tête de lecture T est réécrite en utilisant la fonction $f(\alpha_{i-1},\alpha_i,\alpha_{i+1})$. Ensuite, la lettre α_i est conservée dans la partie gauche de la règle de réécriture, mais positionnée derrière la tête de lecture. Ainsi, elle pourra être utilisée pour calculer la transformation de la lettre suivante α_{i+1}. Une fois qu'elle aura servie à calculer la transformation de la lettre qui la suit, elle ne sera plus réécrite. Un exemple plus complet d'utilisation de cette grammaire est donnée dans le paragraphe de l'érosion de sous-ensemble flous.

Une grammaire contextuelle sur un alphabet de lettres floues

Ce système de réécriture définit une grammaire contextuelle et ne peut donc pas se formaliser plus simplement sous la forme d'un automate[11]. L'alphabet des lettres sur lequel il est défini n'est pas de cardinal infini car il est construit à partir des degrés d'appartenance qui composent la base d'apprentissage, qui comporte un nombre fini

11. C'est un résultat bien connu de la théorie des langages : un automate reconnaît un langage *rationnel* alors qu'une grammaire contextuelle reconnaît des langages qui ne sont pas obligatoirement rationnels.

d'exemples. Cet alphabet est enrichi en lui rajoutant les nouvelles valeurs (en nombre fini) calculées par la fonction f. La notation "avec variables" de la grammaire est utilisée pour la présenter en évitant de développer l'écriture de toutes les règles correspondantes à toutes les possibilités de valeurs.

Les lettres traitées sont des valeurs numériques et elles peuvent ainsi être transformées en de nouvelles lettres en fonction du contexte qui les entoure. Ainsi, c'est un outil différent des automates flous ou des grammaires floues généralement utilisés (Kaufmann, 1975), (Klir et Yuan, 1995). Ici, ce ne sont pas les transitions des règles de réécritures qui sont libellées par un degré de satisfaction flou, mais ce sont directement les lettres de l'alphabet qui sont des valeurs considérées comme floues. La grammaire est, en elle-même, une grammaire contextuelle classique, non floue.

Il peut aussi être intéressant d'étendre la grammaire donnée, en remplaçant les triplets de valeurs numériques par des valeurs triangulaires floues afin de donner une grammaire contextuelle dont les mots reconnus sont composés par les lettres d'un alphabet flou. Cela n'est pas développé ici.

À partir de la forme générique de grammaire contextuelle ainsi définie, il est maintenant possible de formaliser les opérateurs d'érosion et de dilatation qui permettent de construire les opérateurs d'ouverture, de fermeture et de filtrage nécessaire pour lisser les modalités floues construites à l'étape précédente. Cela amène la définition de l'algorithme FPMM', qui est une extension de l'algorithme FPMM pour la prise en compte de données numériques-symboliques dans la base d'apprentissage dans la construction de partitions floues.

Comme il est précisé dans l'expression du système de réécriture, la spécificité de l'opérateur morphologique est rendu par le choix de la fonction f choisie pour réécrire les lettres.

Une fonction adaptée pour chaque type de transformation

La définition de la fonction f en fonction de la transformation souhaitée est maintenant un problème de choix. Il existe plusieurs extensions de la théorie de la morphologie mathématique aux sous-ensembles flous dont celle proposée par (Bloch et Maître, 1992; Bloch et Maître, 1993a; Bloch et Maître, 1993b) ou celle de (De Baets et al., 1997). Dans l'extension proposée par (Bloch et Maître, 1992), les sous-ensembles flous sont érodés par un élément structurant qui peut être un autre sous-ensemble flou. Les opérateurs d'érosion et de dilatation sont représentés par l'utilisation de t-normes et t-conormes. C'est sous une forme équivalente que f peut-être définie, mais, comme pour les transductions de la partie précédente, en utilisant un élément structurant implicite.

Les fonctions présentées dans ce qui suit pour formaliser les opérations d'érosion et de dilatation de sous-ensembles flous ne sont que des formes particulières que nous proposons. Il est tout à fait possible de choisir d'autres fonctions pour éroder et dilater, la différence des fonctions entres elles venant essentiellement de la puissance avec laquelle un degré sera altéré (en gain ou en perte).

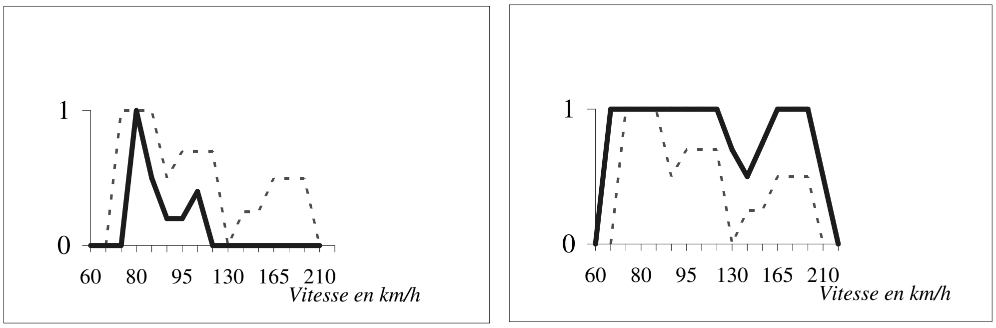


Figure 4.17 – *Érosion et dilatation de sous-ensembles flous*

Un opérateur d'érosion de sous-ensembles flous

La fonction $f_{\mathrm{Er}} : [0,1] \times [0,1] \times [0,1] \longrightarrow [0,1]$, telle que $\alpha'_i = f_{\mathrm{Er}}(\alpha_{i-1}, \alpha_i, \alpha_{i+1})$ qui permet de réaliser une érosion, doit permettre de vérifier $\alpha'_i \leq \alpha_i$ avec $\alpha'_i = \alpha_i$ si et seulement si $\alpha_{i-1} = \alpha_{i+1} = 1$, c'est-à-dire que le degré d'un élément ne doit pas s'éroder s'il est entouré d'éléments de degré d'appartenance égal à 1. Il est souhaitable que l'érosion d'un degré α_i soit d'autant plus forte que les éléments qui l'entourent appartiennent à une classe différente. Ou encore, qu'ils possèdent un degré faible à la classe en question, ce qui correspond, dans un cas à deux classes, à un degré élevé pour la classe opposée.

Une fonction f_{Er} qui réalise une telle érosion est la suivante :

$$f_{\mathrm{Er}}(\alpha_{i-1}, \alpha_i, \alpha_{i+1}) = \max(0;\ \alpha_i - \max(\bar{\alpha}_{i-1}, \bar{\alpha}_{i+1}))$$

où $\bar{\alpha}_i$ correspond au plus fort des degrés de l'élément e_i de la classe différente de celle de α_i. L'inconvénient d'utiliser $\bar{\alpha}_i$ est qu'il faut alors connaître pour éroder chaque élément, tous ses degrés d'appartenance aux différentes classes. Cela peut se contourner en utilisant la fonction f'_{Er} pour éroder un élément seulement en fonction des degrés de ses voisins pour la même classe :

$$f'_{\mathrm{Er}}(\alpha_{i-1}, \alpha_i, \alpha_{i+1}) = \max(0;\ \alpha_i - \max(1 - \alpha_{i-1}, 1 - \alpha_{i+1}))$$

Par exemple, dans le cas de la base donnée précédemment (Table 4.2), le mot à éroder au départ est :

$$0\ \ 1\ \ 1\ \ 0.5\ \ 0.7\ \ 0.7\ \ 0.35\ \ 0\ \ 0.25\ \ 0.25\ \ 0.4\ \ 0.5\ \ 0.5\ \ 0.5\ \ 0.25\ \ 0$$

Les lettres de contrôle sont d'abord rajoutées pour obtenir le mot

$$\mathtt{T}\ \ \#\ \ 0\ \ 1\ \ 1\ \ 0.5\ \ 0.7\ \ 0.7\ \ 0.35\ \ 0\ \ 0.25\ \ 0.25\ \ 0.4\ \ 0.5\ \ 0.5\ \ 0.5\ \ 0.25\ \ 0\ \ \$$$

La première règle à s'appliquer est la règle

$$\mathtt{T}\#\alpha_1\alpha_2 \ \longrightarrow\ f(0, \alpha_1, \alpha_2)\alpha_1\mathtt{T}\alpha_2$$

en instanciant $\alpha_1 = 0$ et $\alpha_2 = 1$. Dans ce cas, $f_{\mathrm{Er}}(0, 0, 1) = 0$, en supposant que l'on se place dans un cas à deux classes où l'on a $\forall i\ \bar{\alpha}_i = 1 - \alpha_i$, et la règle devient alors

$$\mathtt{T}\#0\ 1 \ \longrightarrow\ 0\ 0\ \mathtt{T}\ 1$$

La suite des règles ainsi appliquées est

T#0 1	$\longrightarrow$	0 0 T 1	sortie :	**0** 0 T 1 1 0.5 0.7 0.7 0.35...
0 T 1 1	$\longrightarrow$	0 1 T 1	sortie :	**0 0** 1 T 1 0.5 0.7 0.7 0.35...
1 T 1 0.5	$\longrightarrow$	0.5 1 T 0.5	sortie :	**0 0 0.5** 1 T 0.5 0.7 0.7 0.35...
1 T 0.5 0.7	$\longrightarrow$	0.2 0.5 T 0.7	sortie :	**0 0 0.5 0.2** 0.5 T 0.7 0.7 0.35...
0.5 T 0.7 0.7	$\longrightarrow$	0.2 0.7 T 0.7	sortie :	**0 0 0.5 0.2 0.2** 0.7 T 0.7 0.35...
0.7 T 0.7 0.35	$\longrightarrow$	0 0.7 T 0.35	sortie :	**0 0 0.5 0.2 0.2 0** 0.7 T 0.35...

etc...

La Figure 4.17 montre le résultat de l'application d'une telle érosion sur l'ensemble flou de la base d'exemples, mais en prenant un pas entre les lettres plus fin que le pas de 10 choisit initialement.

Un opérateur de dilatation de sous-ensembles flous

De la même façon que pour l'érosion, on peut définir une fonction $f_{\mathrm{Di}} : [0,1]\times[0,1]\times[0,1] \longrightarrow [0,1]$, telle que $\alpha'_i = f_{\mathrm{Di}}(\alpha_{i-1}, \alpha_i, \alpha_{i+1})$ qui permette de réaliser la dilatation d'un mot. En termes de degrés d'appartenance, une dilatation se traduit par le fait que $\alpha'_i \geq \alpha_i$. De même que pour l'érosion, on souhaite que la dilatation, c'est-à-dire le gain en terme de degré d'appartenance, soit d'autant plus importante que les degrés qui entourent α_i sont élevés. Une fonction f_{Di} qui réalise la dilatation peut s'écrire :

$$f_{\mathrm{Di}}(\alpha_{i-1}, \alpha_i, \alpha_{i+1}) = \min(1;\ \alpha_i + \max(\alpha_{i-1}, \alpha_{i+1}))$$

La Figure 4.17 donne un exemple de l'application d'une telle dilatation sur un ensemble flou.

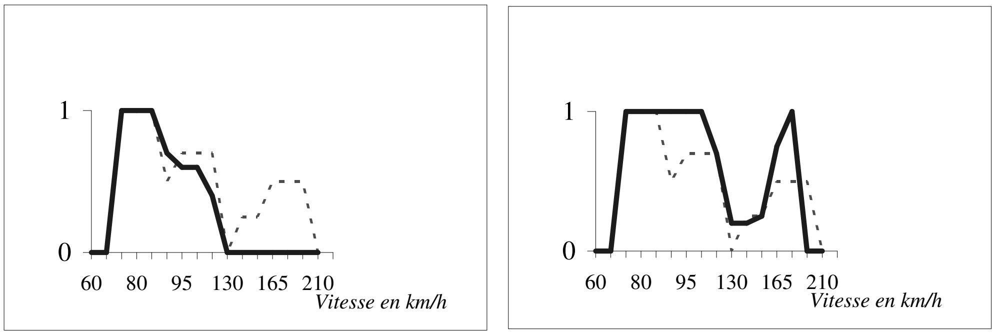


Figure 4.18 – *Ouverture et fermeture de sous-ensembles flous*

Les opérateurs d'ouverture et de fermeture de sous-ensembles flous

Comme dans le cas non flou, ces opérateurs sont construits par la combinaison des opérateurs d'érosion et de dilatation (Figure 4.18).

Ainsi, l'ouverture d'un sous-ensemble flou est le système de réécriture qui résulte de la composition des deux systèmes de réécriture d'ensembles flous définis précédemment. Un sous-ensemble flou est d'abord érodé puis dilaté.

De même, la fermeture d'un sous-ensemble flou est le système de réécriture qui résulte de l'autre façon de composer les deux systèmes de réécriture d'ensembles flous définis précédemment. Un sous-ensemble flou est d'abord dilaté puis érodé.

Le filtrage d'un sous-ensembles flous

L'algorithme FPMM est étendu pour s'appliquer aux sous-ensembles flous en substituant les opérateurs d'érosion et de dilatation par les opérateurs de transformation de sous-ensembles flous présentés dans cette partie. Ainsi, grâce à cet algorithme, une partition floue *générale* est construite à partir d'une base d'apprentissage dans laquelle chaque élément possède un degré d'appartenance pour chaque classe de la base.

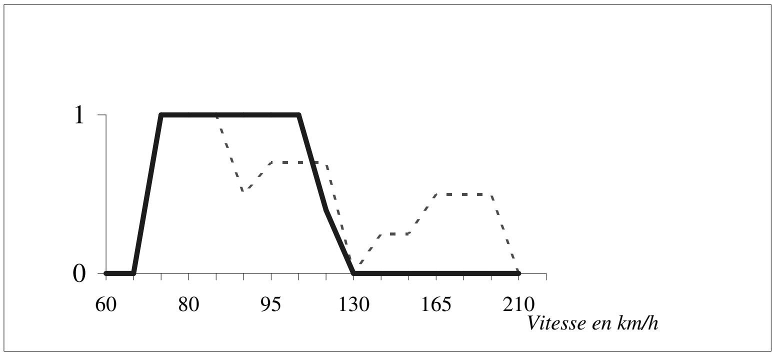


Figure 4.19 – *Filtrage simple d'un sous-ensemble flou*

La reformulation de l'algorithme FPMM en tenant compte des nouveaux opérateurs présentés dans cette partie, donne le nouvel algorithme FPMM' suivant :

Algorithme 4.2 (FPMM') *Pour inférer une partition floue relative aux classes pour un attribut à modalités floues, pour chaque classe :*

1. *Agréger les modalités de l'attribut ainsi que leur degré d'appartenance à la classe selon la méthode présentée dans le paragraphe 4.4.1.*

2. *Estomper le sous-ensemble flou résultant de l'union précédente en utilisant les nouveaux opérateurs présentés pour lisser un sous-ensemble flou.*

Les sous-ensembles flous ainsi construits relativement à chacune des classes sont utilisés directement pour le calcul de l'information apportée par une coupure sur l'attribut en question. Chaque sous-ensemble flou est une nouvelle modalité de l'attribut, calculée à partir des degrés d'appartenance à une classe. Elle a l'avantage d'être plus générale que l'union classique des valeurs existantes dans la base d'apprentissage pour cet attribut et d'être de plus liée à une classe.

4.5 Conclusion

Dans ce chapitre, une méthode automatique pour inférer une partition floue à partir d'un ensemble de données numériques ou numériques-symboliques a été présentée.

Cette méthode tient compte de la présence de classes floues ou non dans le processus de construction. Elle est fondée sur l'utilisation d'opérateurs dérivés de la théorie de la morphologie mathématique. Les opérateurs de base que sont l'opérateur d'érosion et l'opérateur de dilatation sont définis dans le contexte spécifique des données à traiter :

- transductions pour le traitement de mots d'un langage.
- grammaires contextuelles pour transformer des sous-ensembles flous.

La distinction entre les deux cas est faite dans un souci de simplicité de mise en œuvre : une transduction est un outil plus facilement implémentable et manipulable qu'une grammaire contextuelle. Les transductions présentées sont une forme adaptée à la spécificité du cadre non flou des mots à transformer. Il est à noter qu'elles sont la simplification de grammaires contextuelles qui peuvent être aussi définies dans ce cadre là (Marsala, 1995).

En présence de données non floues, ces deux méthodes sont équivalentes. Dans ce cas là, la grammaire contextuelle présentée se simplifie en transduction en utilisant le fai t que les degrés d'appartenance ne peuvent alors prendre que deux valeurs 0 ou 1.

Dans le cadre des grammaires contextuelles, les transformations morphologiques sont réalisées par le biais de fonctions d'altération des degrés d'appartenance. Ces fonctions ne sont pas uniques et les fonctions usuelles de la morphologie mathématique floue peuvent aussi être utilisées.

A la différence des opérateurs classiques de la morphologie mathématique, l'élément structurant qui est utilisé pour les transformations morphologiques ainsi définies est ici implicite dans le sens où la transformation subie par point de l'ensemble ne dépend que de ses deux voisins immédiats. L'élément structurant est donc fixé une fois pour toute.

Finalement, il faut remarquer que cette méthode automatique n'est pas en elle-même une méthode d'apprentissage comme le sont, par exemple, les méthodes de regroupement [12] classiques. C'est une méthode qui fait partie d'un processus d'apprentissage, en l'occurrence la construction d'arbres de décision flous. Dans ce cas, elle permet d'inférer de *bonnes* partitions floues pour chaque attribut qui seront comparées entre elles par une mesure de discrimination durant le développement de l'arbre de décision.

12. *clustering*

Chapitre 5

Forêt d'arbres de décision flous

Tous ensemble nous savons plus de choses qu'un seul d'entre nous n'en peut connaître.

J. Brunner – *Sur l'onde de choc*

5.1 Introduction

Lorsqu'il s'agit d'apprendre à reconnaître plus de deux classes, de nouveaux problèmes apparaissent et perturbent l'efficacité des algorithmes de construction d'arbres de décision.

En effet, comme il a été précisé dans le chapitre 2, les algorithmes usuels de construction d'arbres de décision – les algorithmes descendants – recherchent le meilleur attribut pour partitionner la base d'apprentissage à l'aide d'une mesure de discrimination. Cette mesure rend compte du pouvoir discriminant d'un attribut relativement aux classes et permet ainsi de comparer les attributs entre eux. Cependant, les mesures utilisées n'arrivent pas à rendre compte, de façon intuitivement convenable, de ce pouvoir de discrimination d'un attribut quand il existe plus deux classes à séparer. Elles restent efficaces en présence de deux classes mais elles ne sont plus adaptées à ce nouveau type de problèmes.

Étant donné que le problème principal pour construire un arbre de décision quand il existe plus de deux classes dans la base d'apprentissage est lié à la mesure de discrimination utilisée pendant la construction de l'arbre, il est alors possible, soit d'adapter cette mesure à la spécificité du problème multiclasses, soit d'adapter le problème à la spécificité de la mesure. Un autre choix possible est de ne pas raisonner sur la mesure utilisée mais de modifier les données du problème pour les adapter aux moyens disponibles. Donc, si les mesures usuelles sont efficaces dans les problèmes à deux classes mais rencontrent des difficultés à traiter les problèmes à plus de deux classes, la solution naturelle est de se ramener, dans tous les cas, à un problème à deux classes. C'est une approche très intuitive et naturelle ; pour un élément à classer donné, il est souvent plus efficace de procéder en plusieurs étapes, pour chaque classe existante, en se demandant s'il appartient à une classe ou s'il n'y appartient pas, plutôt que d'essayer de savoir globalement à quelle classe il appartient. Nous nous basons sur une telle solution et nous proposons le système *Tanit* qui permet la construction d'un ensemble – une forêt – d'arbres de décision flous.

En plus de construire une telle forêt, il faut aussi pouvoir l'utiliser pour classer de nouveaux exemples. Lors de l'utilisation d'un tel système de forêt d'arbres en classification, pour chaque exemple à classer, un degré d'appartenance aux classes sera donné par tous les arbres flous construits. Ces degrés doivent alors être agrégés pour permettre d'associer un degré unique à chacune des classes possibles. Ici, ce type de méthode tire profit de la propriété des arbres de décision flous qui est d'associer un degré d'appartenance à chaque classe.

Dans ce chapitre, différentes solutions existantes à ce problème ainsi que la solution que nous avons adoptée, basée sur la construction d'une forêt d'arbres de décision flous, sont présentées. La première section de ce chapitre concerne les différentes méthodes possibles pour apprendre en présence de plus de deux classes. La deuxième section présente ensuite les différentes possibilités pour agréger les résultats de classification de plusieurs classifieurs. Ensuite, notre application générique *Tanit* de construction et d'utilisation d'une forêt d'arbres de décision flous est décrite. Quelques résultats obtenus par Tanit sur quelques bases d'apprentissage sont ensuite donnés. Finalement, la dernière section étudie la complexité informatique d'une telle structure de forêt d'arbres de décision flous.

5.2 Apprendre en présence de plus de deux classes

5.2.1 Le problème pour discriminer plus de deux classes

L'algorithme de construction de l'arbre de décision procède toujours de la même façon quel que soit le nombre de classes présentes dans la base d'apprentissage : grâce à une mesure de discrimination, un attribut est sélectionné pour décomposer la base et former un nœud de l'arbre. L'attribut choisi est celui qui possède le plus grand pouvoir discriminant vis-à-vis des classes de la base, relativement à la mesure de discrimination utilisée. L'utilisation d'une mesure de discrimination est une heuristique qui doit permettre de minimiser la taille des arbres construits.

Or, si les mesures généralement utilisées arrivent à rendre compte du pouvoir discriminant d'un attribut dans un problème à deux classes, en présence de plus de deux classes, ces mesures doivent alors rendre compte du pouvoir discriminant d'un attribut pour toutes les classes *simultanément*. Il est clair que l'évaluation d'un tel pouvoir devient alors un problème très délicat et les mesures usuelles sont rarement capables d'y faire face convenablement, ou du moins, d'une façon *intuitivement* convenable.

Nilsson relève déjà ce type de problèmes en 1965. Pour lui, un classifieur définit un hyperplan de séparation entre des classes (Nilsson, 1965). Or, il apparaît des cas pour lesquels il peut être impossible d'obtenir une surface de séparation simple pour toutes les classes simultanément. Ceci est d'autant plus vérifié dans le cas des arbres de décision que la surface de séparation définie par une question binaire posée sur un attribut est un plan dans l'espace. Par exemple, dans le cas où l'on travaille dans un plan, la surface de séparation définie par une question sur la valeur d'un attribut est une droite qui ne peut donc séparer que deux classes.

Dans le domaine des arbres de décision, (Fayyad et Irani, 1992a) donne l'exemple de la Figure 5.1 où une partition *naturelle* sera rejetée par une mesure comme l'entropie de Shannon, au détriment d'une partition moins intuitive.

Dans cet exemple, la partition engendrée par l'attribut A obtient un meilleur gain d'information que celle engendrée par l'attribut B. L'attribut A sera donc choisi pour partitionner l'ensemble et libeller le nœud de l'arbre en cours de construction. Pourtant, la partition engendrée par B paraît plus efficace dans le sens qu'elle est capable de discriminer complètement une des trois classes par rapport aux deux autres. L'attribut choisi d'une telle façon est celui qui optimise la mesure utilisée, ce qui n'est pas obligatoirement celui qui possède le meilleur pouvoir discriminant pour une classe donnée.

C'est un fait connu car l'entropie de Shannon n'est censée mesurer que le niveau de désordre dans un ensemble et non pas la prépondérance d'un des éléments de cet ensemble vis-à-vis des autres éléments.

L'utilisation des mesures usuelles, comme l'entropie de Shannon, est une heuristique pour construire un arbre de décision et par conséquent, comme pour toute heuristique, il peut exister des cas particuliers où elle ne s'applique pas d'une façon optimale. Ainsi, l'entropie de Shannon permet de choisir la décomposition d'un ensemble en sous-ensembles en minimisant le désordre dans chaque sous-ensemble, d'une façon globale. Elle n'est pas construite pour favoriser un sous-ensemble, même partiellement ordonné, au détriment des autres.

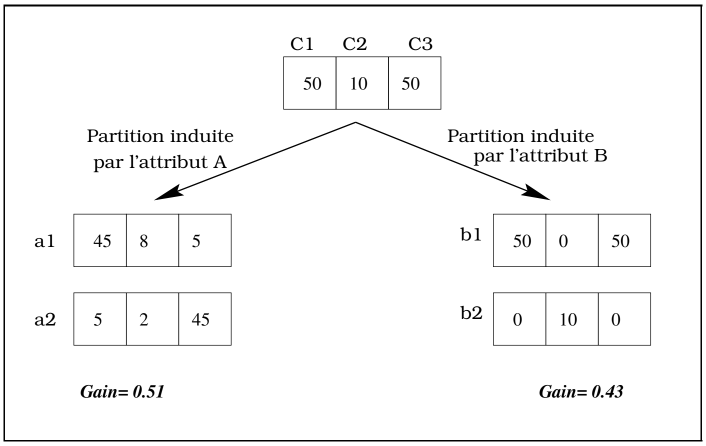


Figure 5.1 – *Problème de partition multiclasses (Exemple de (Fayyad et Irani, 1992a))*

5.2.2 Une première solution : améliorer la mesure

Pour pallier ce problème, un choix possible est d'améliorer la mesure de discrimination. Cela revient donc à trouver une mesure capable de rendre compte du pouvoir discriminant d'un attribut relativement à plus de deux classes simultanément.

C'est la solution proposée par (Fayyad et Irani, 1992a). Après s'être rendu compte des déficiences d'une mesure telle l'entropie de Shannon dans le cas où il existe plus de deux classes, les auteurs suggèrent d'utiliser un autre type de mesure pour évaluer ce pouvoir discriminant. La mesure qu'ils proposent est basée sur le produit scalaire utilisé en géométrie classique :

$$\cos(\widehat{V, W}) = \frac{V \circ W}{\|V\| \cdot \|W\|}$$

en choisissant

$$V \circ W = \sum_{i=1}^{k} V_i W_i \quad ,$$

où V et W sont les vecteurs correspondant à la répartition des classes dans les deux sous-ensembles engendrés par la partition binaire. V_i (resp. W_i) est la i-ème composante du vecteur V (resp. W).

C'est une solution séduisante et très intéressante, mais, dans ce cas, il apparaît alors de nouvelles difficultés. Ainsi, comment arriver avec ce type de mesure à différencier les répartitions optimales ? Par exemple, en référence à la Figure 5.1, l'ensemble $(50, 10, 50)$ peut se partitionner en $(50, 0, 50)$ et $(0, 10, 0)$ ce qui donnera une mesure optimale de 0 pour cette partition. Mais il peut aussi se partitionner en $(50, 10, 0)$ et $(0, 0, 50)$ qui donnera la même mesure de 0. Or, il est plus logique de préférer la deuxième partition à la première car elle permet de classer immédiatement un plus grand nombre d'individus.

Comme il a été précisé au paragraphe précédent, l'utilisation d'une mesure pour comparer les attributs entre eux, est une heuristique pour laquelle il peut exister des cas pathologiques où elle ne s'applique pas de façon optimale. L'amélioration de la mesure est une voie qui reste encore à approfondir et qui engendre aussi de nouveaux problèmes spécifiques.

5.2.3 Une autre solution : regrouper les classes

Une autre solution est de conserver les mesures de discrimination usuelles, reconnues comme très efficaces dans les cas à deux classes, et de se ramener, dans tous les cas, à un problème à deux classes. Ainsi, il s'agit de regrouper les classes afin d'avoir à discriminer une classe contre toutes les autres réunies.

C'est la solution adoptée par (Breiman et al., 1984) dans leur algorithme CART. Dans un cas multiclasses, ils proposent de regrouper, à chaque nœud, les classes en deux *super-classes* qui regroupent chacune un ensemble des classes de l'ensemble. La mesure de discrimination à deux classes permet alors d'être utilisée telle qu'elle pour comparer le meilleur choix de super-classes. Mais cela peut devenir alors un processus coûteux en temps de calcul et, dans certains cas, il peut être fastidieux de tenter de tels regroupements.

En fait, il est vain de vouloir rechercher un regroupement optimal de classes. Il est suffisant de regrouper les classes une fois pour toutes, avant le processus d'apprentissage. Cela reste ainsi un processus peu coûteux et facile à mettre en œuvre. La base d'apprentissage est transformée en plusieurs bases, induites chacune par une classe à reconnaître. En fait, cela revient alors à créer, pour toutes les classes possibles, une base d'apprentissage avec cette classe contre toutes les autres réunies. Chaque base d'apprentissage ainsi constituée peut être alors traitée par un classifieur spécifique d'un ensemble multi-classifieurs construit dans le but de reconnaître le même ensemble de classes. Chaque classe sera donc confrontée à toutes les autres. Dans le cadre des arbres de décision, cela revient à dupliquer la base d'apprentissage en autant de bases qu'il y a de classes. Dans chaque base, une seule classe est conservée, les autres sont regroupées en une classe unique.

La solution proposée par (Nilsson, 1965) est de cet ordre. Dans le domaine des réseaux de neurones, il introduit la notion de *comité*[1]. Un comité est un système de classifieurs, chaque classifieur étant adapté à la définition d'un hyperplan particulier, c'est-à-dire que chaque classifieur est adapté à la reconnaissance d'une classe. Pour classifier un nouvel exemple, chaque classifieur donne son opinion sur l'appartenance (ou la non appartenance) à la classe que ce classifieur est censé reconnaître. Les résultats de tous les classifieurs sont ensuite agrégés pour obtenir la classe de l'objet à classifier.

Dans le domaine de la classification non paramétrique, (Friedman, 1977) propose lui aussi ce type de méthodes dans les cas multiclasses. En fait, il suggère d'appliquer l'idée de (Nilsson, 1965) mais dans le cadre des arbres de décision et non plus dans le cadre des réseaux de neurones. Les classifieurs qu'il utilise sont donc des arbres de décision. Il envisage la construction d'autant d'arbres de décision que de classes et utilisera une technique proche de la technique du vote que nous allons présenter pour assigner une classe à un objet à classer.

De même, dans sa théorie sur l'apprentissage inductif, (Michalski, 1983) ne recense que deux types d'instances pour caractériser un concept :

- les instances qui valident le concept, les instances positives ;
- les instances qui infirment le concept, les instances négatives.

Ainsi, tout problème de reconnaissance de concept se réduit donc à la reconnaissance de la distinction d'éléments positifs et d'éléments négatifs de ce concept. Il ne considère donc pas de problème multiclasses mais se ramène toujours à un problème à deux classes.

5.2.4 Notre solution : construire une forêt d'arbres de décision flous

Par conséquent, nous proposons de dupliquer la base d'apprentissage en autant de bases que de classes à reconnaître, et pour chaque base associée à une classe particulière, de construire un arbre de décision flou pour reconnaître la classe en question contre toutes les autres réunies. Un tel système donne naissance à une *forêt d'arbres de décision flous*.

La construction d'une structure d'ensemble d'arbres de décision n'est pas récente. Déjà (Friedman, 1977) proposait de construire un ensemble d'arbres de décision pour gérer les problèmes multiclasses.

Plus récemment, (Feray et de Brucq, 1996) ont étudié un système qui construit plusieurs arbres de décision en faisant varier plusieurs mesures d'information lors de la recherche d'un attribut comme test à un nœud de l'arbre. Ces différentes mesures

1. Traduction de l'expression anglaise "Committee machine" donnée par Nilsson.

peuvent élire différents attributs, chacun d'eux donnant alors naissance à un arbre différent. Mais à la différence de la méthode que nous proposons, leur façon de procéder ne résout pas pour autant le problème du traitement des cas multiclasses.

À la différence de ces structures d'arbres de décision, les forêts que nous proposons d'utiliser sont des forêts d'arbres de décision *flous*. L'avantage d'utiliser de tels arbres flous réside dans l'apport de la gradualité dans la décision qu'ils sont capables de donner. À un nouvel objet, un arbre de décision flou associe un ensemble de modalités de la classe, chaque modalité associée à un degré d'appartenance. Cette gradualité est un atout majeur pour la combinaison des décisions de plusieurs arbres de décision flous concernant un nouvel exemple à classer.

L'application *Tanit* qui construit une forêt d'arbres de décision flous, chaque arbre étant construit à l'aide de l'application Salammbô, est décrite dans la section 5.4 de ce chapitre.

5.3 Classer grâce à une forêt d'arbres de décision

Il est ensuite important de pouvoir utiliser une forêt d'arbres de décision flous construite par la méthode précédente pour classer les exemples d'une base de test.

Dans cette phase de classement, chaque arbre de décision flou de la forêt sait classer les exemples en fonction de la classe pour lequel il a été construit, en déterminant des degrés d'appartenance aux classes comme il l'a été précisé dans le paragraphe 2.5. Ainsi, pour chaque exemple, pour chaque arbre de décision flou, un degré d'appartenance $m_{c_i}(\{c_i\})$ à la classe c_i qu'il est sensé reconnaître, ainsi qu'un degré d'appartenance $m_{c_i}(\{\overline{c_i}\})$ aux classes différentes $\overline{c_i}$.

Pour obtenir la classe d'un nouveau cas à classer, il faut agréger tous ces degrés d'appartenance pour n'obtenir qu'un degré d'appartenance à chaque classe.

Il existe différentes méthodes pour agréger de tels résultats. C'est ici le domaine de la fusion d'informations multi-classifieurs. Une décision devra être prise concernant un exemple à classer en tenant compte des opinions données par tous les différents arbres de décision construits.

L'agrégation d'informations est un domaine à part entière. De nombreux travaux sont consacrés à l'étude des meilleurs moyens pour agréger un ensemble de données pour en déduire une donnée synthétique capable d'être utilisée pour prendre une décision ou pour être comparée avec les valeurs issues d'autres ensembles de données (Bloch, 1996), (Kelman, 1996), (ISIS, 1996). En particulier, plusieurs auteurs ont étudié le cas de l'agrégation d'informations provenant de plusieurs classifieurs (Nilsson, 1965), (Friedman, 1977), (Ho et al., 1992), (Xu et al., 1992), (Bracker, 1996), (Loonis, 1996), (Kuncheva, 1997), (Bernhardt et al., 1997).

Dans cette section, après avoir rappelé brièvement ces différentes méthodes, nous présentons la hiérarchie de (Xu et al., 1992) qui permet de classifier les différentes méthodes d'agrégation et, peut être, d'en choisir une. Ensuite, nous détaillons les deux méthodes d'agrégation les plus courantes dans les systèmes multi-classifieurs : la méthode du vote et la méthode de combinaison d'évidence de Dempster.

La théorie de l'évidence permet, dans notre cadre, d'introduire une autre méthode d'agrégation des degrés. Cette méthode n'est pas nouvelle car elle se contente de remarquer l'analogie entre le système de classifieur et la répartition des masses de croyance dans le cadre de la théorie de l'évidence.

Finalement, nous terminons sur notre proposition de solution concernant le problème de l'indépendance des classifieurs d'un système multi-classifieurs. Le concept d'indépendance que nous choisissons est celui de Kampé de Fériet qui permet de considérer tous les classifieurs de notre système comme indépendants.

5.3.1 L'agrégation dans les systèmes multi-classifieurs

Les premières méthodes proposées pour agréger les résultats provenant de plusieurs classifieurs sont celles de (Nilsson, 1965) et de (Friedman, 1977).

(Nilsson, 1965) propose d'utiliser la technique du vote pour obtenir le consensus de la décision. Les sorties binaires de ses classifieurs sont additionnées et la somme est comparée à un seuil voulu pour déterminer la classe de l'exemple.

Quant à lui, (Friedman, 1977) suggère de compter, pour chaque arbre i construit, le nombre C_i d'éléments d'apprentissage ayant la classe i reconnue par l'arbre et le nombre O_i d'éléments ayant les autres classes dans la feuille atteinte par l'exemple à classer. La classe finale de l'exemple sera celle pour laquelle $C_i - O_i$ sera maximum sur l'ensemble des arbres.

Chez (Ho et al., 1992), pour un exemple à classer, chaque classifieur retourne un ordre sur les classes, de la classe en laquelle le classifieur croit le plus à celle en laquelle il croit le moins. Ce sont les ordonnancements de tous les classifieurs qui vont être agrégés pour trouver l'ordre consensuel sur les classes pour la totalité des classes. Pour une même classe, tous les rangs retournés par l'ensemble des m classifieurs du système constituent un vecteur $R =< r_1, ..., r_m >$ qui est associé à la classe pour l'exemple en question. La valeur de confiance S en la classe, pour cet ensemble de classifieurs, est calculée par une fonction de ce vecteur $S = f(R)$. (Ho et al., 1992) recensent trois types de méthodes pour calculer cette valeur de confiance :

- la méthode du *plus haut degré* qui n'utilise que le meilleur classement obtenu : $f(R) = \min(r_1, ..., r_m)$
- la méthode de *comptage de Borda* (proche de la méthode du vote) : $f(R) = \sum_{i=1}^{m} r_i$
- la méthode de *comptage de Borda généralisé*: $f(R) = \sum_{i=1}^{m} \omega_i r_i$ (Les ω_i sont des cœfficients pour lesquels il existe différentes méthodes de détermination)

Le problème de ce dernier type de méthodes, comme pour la méthode des opérateurs OWA [2], reste que la détermination des poids associés aux classifieurs est très difficile et délicate. Mal faite, elle peut dégrader les performances de l'ensemble et faire perdre l'avantage d'utiliser plusieurs classifieurs.

En plus de la méthode du vote, (Xu et al., 1992), (Bracker, 1996) et (Loonis, 1996) présentent la règle de combinaison d'évidences de Dempster (Shafer, 1976) comme autre méthode d'agrégation de données provenant de plusieurs classifieurs.

2. *Ordered Weighted Averaging Operators* (Yager, 1988)

(Kuncheva, 1997) introduit elle l'utilisation des opérateurs OWA pour combiner plusieurs décisions en classification.

5.3.2 Une hiérarchie des méthodes d'agrégation

Pour clarifier la présentation des différentes méthodes, nous allons présenter ici la hiérarchie des méthodes d'agrégation proposée par (Xu et al., 1992) et reprise par (Loonis, 1996). Cette hiérarchie sera très utile pour repérer les différences ou les similarités entre les différentes méthodes présentées.

(Xu et al., 1992) définissent d'abord trois niveaux d'algorithmes de classification en fonction du type de leurs sorties :

- niveau *abstrait*: la sortie du classifieur est une classe unique.
- niveau *classement*: la sortie du classifieur est un ordre de préférence sur les classes.
- niveau *mesure* : la sortie du classifieur est une classe associée à une mesure reflétant le degré avec lequel l'exemple à classer appartient à cette classe.

Tous les types de sorties de classifieurs sont couverts par ces trois niveaux. Les auteurs présentent alors trois types d'agrégation induits chacun par les différents niveaux de sortie :

- type I : Combinaison des informations de niveau abstrait.
- type II : Combinaison des informations de niveau classement.
- type III : Combinaison des informations de niveau mesure.

Par conséquent, pour (Xu et al., 1992), une méthode d'agrégation se différencie des autres méthodes par ce qu'elle agrège, et non pas de la façon dont elle agrège. Par contre, cette façon de cataloguer un ensemble de méthodes par les entrées qu'elles peuvent prendre, est seulement une possibilité de classification parmi tant d'autres. Il pourrait paraître plus naturel d'étudier une hiérarchisation de ces méthodes en fonction de l'algorithme qui les compose. Par exemple, la méthode du vote, qui est de type I, est une méthode qui, si elle s'applique à des données de niveau *mesure*, devient une méthode de type III.

5.3.3 Quelques méthodes d'agrégation usuelles

Comme il a été précisé plus haut, pour chaque exemple, chaque arbre de décision flou détermine un degré d'appartenance $m_{c_i}(\{c_i\})$ à la classe c_i qu'il est censé reconnaître, ainsi qu'un degré d'appartenance $m_{c_i}(\{\overline{c_i}\})$ aux autres classes (notées $\overline{c_i}$). Ainsi, chaque arbre est un classifieur du niveau *mesure* dans la hiérarchie de (Xu et al., 1992). Cela est pratique dans le sens où les deux autres niveaux peuvent se déduire de ce niveau là.

Pourtant, pour nous, il est important de garder un tel niveau de classifieurs car cela nous autorise à profiter pleinement des spécificités de classification floue offertes par les arbres de décision flous. Ce sont donc ces degrés qui vont être agrégés et ce sont des méthodes de type III qui vont donc nous intéresser.

Définir ce qu'est une bonne méthode d'agrégation est encore un problème complexe. Une méthode peut s'avérer bonne sur un certain type de domaine d'apprentissage mais elle peut aussi donner de mauvais résultats sur d'autres domaines d'apprentissage. L'étude de plusieurs méthodes s'avère alors nécessaire afin soit d'en découvrir une "universelle" qui donne d'excellents résultats quel que soit le domaine étudié[3], soit de relever les propriétés des méthodes dans les domaines où elles excellent.

Dans les paragraphes suivants, deux méthodes d'agrégation sont présentées dans le cadre spécifique de leur utilisation pour une forêt d'arbres de décision flous. Ensuite, une méthode d'agrégation basée sur les fondements de la théorie de l'évidence de (Shafer, 1976) est rappelée et adaptée à notre problème.

Agrégation par vote

La méthode la plus simple pour agréger ces degrés est la méthode du vote. Dans cette méthode, des votes pour chaque classe, donnés par chaque arbres de décision flous, sont comptabilisés pour élire la classe la plus populaire. Ces degrés donnés par chaque arbres sont ensuite additionnés à l'aide d'une pondération :

$$\mu(c_i) = m_{c_i}(\{c_i\}) + \frac{1}{|n-1|} \sum_{c_j \neq c_i} m_{c_j}(\{\overline{c_j}\})$$

Les degrés associés aux classes concurrentes ($\overline{c_j}$) sont atténués par le nombre de classes n pour limiter l'influence des arbres qui ne sont pas spécialisés dans la reconnaissance de la classe c_i et pour rendre compte du fait que le degré $m_{c_i}(\{\overline{c_i}\})$ doit être réparti dans le calcul des degrés de toutes les classes appartenant à l'ensemble $\overline{c_i}$. Finalement, la classe c_i élue est celle qui a obtenu la plus grande valeur pour $\mu(c_i)$.

En fait, la méthode du vote originelle est une méthode de type I car elle se base sur le vote d'un classifieur pour un candidat. Elle considère donc que les nuances d'opinion d'un classifieur ne sont pas intéressantes à prendre en compte, ce qui enlève alors une des qualités des arbres de décision flous de justement donner une classification nuancée pour chaque exemple. Ici, ce cas particulier de méthode de vote utilisant les degrés donnés par les Salammbô est classifiée de type III dans la hiérarchie des mesures, alors que la méthode est intrinsèquement la même. On est alors amené à penser qu'il paraît assez artificiel de vouloir cataloguer des méthodes seulement sur les entrées qu'elles admettent sans tenir compte de l'algorithme réel qui les compose.

Agrégation par la règle de Dempster-Shafer

Une autre méthode d'agrégation de données provenant de plusieurs classifieurs souvent utilisée est fondée sur la théorie de l'évidence (Shafer, 1976)[4]. Dans le cadre de cette théorie, chaque Salammbô peut être considérée comme un *observateur qui fournit une appréciation sur une situation donnée* (Bouchon-Meunier et Nguyen, 1996). L'appréciation est alors une valeur d'une fonction de masse qui peut servir à déterminer un

3. L'existence d'une telle méthode paraît, à l'étude des travaux existants, peu probable.

4. Les définitions de base de la théorie de l'évidence utiles dans cette partie sont rappelées succinctement en annexe B.

degré de croyance et un degré de plausibilité associés aux classes pour un exemple à classer.

L'univers de référence est alors $\Omega = \{c_1, ...c_n\}$, et les deux valeurs $m_{c_i}(\{c_i\})$ et $m_{c_i}(\{\overline{c_i}\})$ retournées par un arbre de décision flou sont des valeurs d'une fonction de masse. Ces degrés sont alors combinés par la règle de Dempster-Shafer, qui donne alors dans ce cadre-ci :

$$\mu(c_i) = m_{c_i}(\{c_i\})\Pi_{j \neq i} m_{c_j}(\{\overline{c_j}\})$$

C'est afin de pouvoir appliquer cette méthode, qu'il faut vérifier que les arbres de décision d'une forêt peuvent être considérés comme indépendants afin de conserver le sens d'une telle façon de combiner. L'étude de cette indépendance est effectuée dans la section 5.3.5 de ce chapitre.

5.3.4 Une conception de l'agrégation issue de la théorie de l'évidence

Tout le système de la forêt peut aussi être vu comme un système d'expertise qui fournit des masses de croyances pour divers éléments d'un cadre de discernement. Dans ce cas, il est naturel d'essayer de déduire de ces masses une fonction de croyance et une fonction de plausibilité, comme la théorie de l'évidence de Shafer nous permet de le faire.

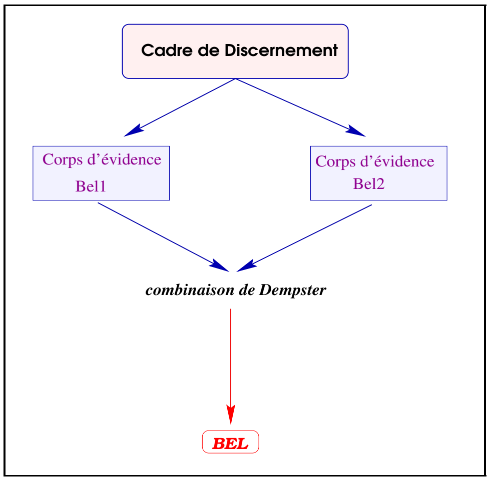
On peut remarquer que, dans le cadre de d'une forêt d'arbres de décision, il est possible de considérer de deux façons distinctes le problème de l'agrégation des degrés renvoyés par chaque arbre (Figure 5.2). Étant donné le cadre de discernement que représente l'espace des classes à reconnaître, soit on considère que chaque arbre retourne une fonction de masse particulière et dans ce cas on est dans le cadre d'application de la règle de Dempster, soit on considère la forêt comme un seul expert qui retourne des valeurs particulières de la même fonction de masse et, dans ce cas, il faut trouver un moyen différent d'agréger les données. Dans le cas où il n'y a qu'un seul même corps d'évidence (Figure 5.2), le cadre de discernement correspondant à notre système est toujours l'ensemble des classes possibles mais chaque arbre de décision flou retourne une valeur spécifique d'une même fonction de masse. Partant de là, il est possible de calculer la fonction de croyance et la fonction de plausibilité associées à cette fonction de masse. On a alors, en utilisant les définitions 16 et 17 données en Annexe B :

$$\mathrm{Bel}(c_i) = m_{c_i}(c_i)$$

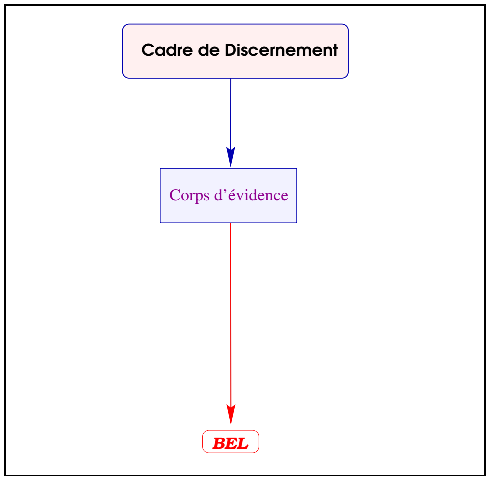
et

$$\mathrm{Pl}(c_i) = m_{c_i}(c_i) + \sum_{j \neq i} m_{c_j}(\overline{c_j})$$

après avoir normalisé les degrés retournés par les Salammbô $m_{c_i}(c_j)$, $\forall i, j$ par la somme $\sum_{i,j} m_{c_i}(c_j)$ de tous les degrés. Une décision peut alors être prise à l'aide de ces deux degrés associés à chaque classe. Il existe alors de nombreuses façons de procéder. Par exemple, une solution est de choisir la classe avec le degré de croyance le plus élevé, ou bien de combiner les deux degrés comme pour le calcul d'une valeur moyenne ($\frac{\mathrm{Bel}(c_i)+\mathrm{Pl}(c_i)}{2}$).



Corps d'évidence distincts



Même corps d'évidence

Figure 5.2 – *Le problème des corps d'évidence*

5.3.5 L'indépendance des classifieurs

Le problème auquel on doit faire face est le suivant : chaque arbre de la forêt détermine deux degrés d'appartenance à deux sous-ensembles des classes de la base. À partir des degrés de tous les arbres de la forêt, un *degré d'appartenance* pour chaque classe [5] doit être déduit.

Le problème majeur dans le choix de la bonne méthode d'agrégation à prendre est, dans un premier temps, d'examiner dans quelle mesure chaque classifieur peut être considéré comme indépendant des autres. En effet, les résultats de classifieurs dépendants ne peuvent pas (et ne doivent pas) se combiner de la même manière que les résultats de classifieurs indépendants pour la détermination de la classe finale de l'exemple.

Le problème de l'indépendance est multiple et très complexe. Ainsi, il est souvent fait référence à cette notion d'indépendance sans pour autant que soit donner une définition formelle de cette indépendance. Dans un premier temps, il faut choisir une *bonne* définition de ce que l'on entend par indépendance. Ensuite, il faut vérifier si notre problème vérifie cette définition.

Notions usuelles d'indépendance

Définir l'indépendance n'est pas trivial. Des travaux existent depuis de nombreuses années pour arriver à préciser ce que l'on peut bien entendre par indépendance. La base de recherche se situe dans le domaine de la théorie des probabilités où l'indépendance est définie de façon *claire* et précise : si deux événements sont indépendants alors leur probabilité d'occurrence conjointe est égale au produit de leurs probabilités respectives.

5. On donne encore ici le nom de *degré d'appartenance* au résultat de la combinaison de tous les degrés, mais en précisant bien que l'on ne doit pas certifier être encore en présence après la combinaison du même *type* de degré qu'avant la combinaison

Ainsi, il est possible de trouver la définition suivante dans tout manuel de probabilité (comme celui de (Kaufmann, 1965) par exemple) :

Définition 4 (Indépendance stochastique) *Soit deux événements A et B appartenant au même ensemble $\mathbb{P}(E)$ des parties d'un référentiel E ; ils sont dits* stochastiquement indépendants *si*

$$p(A|B) = p(A)$$

On dit aussi qu'ils sont statistiquement indépendants *ou* indépendants. *On a alors*

$$p(A \cap B) = p(A)p(B)$$

Cette définition est parfaite dans un cadre probabiliste théorique mais, dès que l'on désire appliquer cette théorie à des problèmes réels, on se heurte à des difficultés pour arriver à la vérifier qui ont amené les scientifiques à reformuler ce problème de l'indépendance en tenant compte justement de la réalité des événements.

Il existe de nombreux travaux qui étudie cette notion d'indépendance et cela reste un important axe de recherche (Groupe LSD, 1997).

Ainsi, par exemple, on peut citer la notion d'indépendance *qualitative* qui a été introduite pour rendre compte de la complexité de ce type de relation (Shafer et al., 1987), (Pearl, 1988), (Zhang, 1993).

De même, l'apparition des théories étendant la théorie des probabilités (comme la théorie de l'évidence, suivie par la théorie des possibilités) a amené à se poser la question d'étendre la définition de l'indépendance pour prendre en compte ces nouvelles théories (Shafer, 1976), (Dubois et al., 1994), et surtout les travaux sur la théorie généralisée de l'information de (Kampé de Fériet, 1975).

On peut aussi citer les travaux de (Zadeh, 1975; Zadeh, 1978) sur la non-interactivité.

Pour nos classifieurs, nous retiendrons la définition de l'indépendance introduite par (Kampé de Fériet, 1975), car cette définition nous paraît s'appliquer à notre système de classifieurs de façon suffisament satisfaisante.

La notion de Kampé de Fériet de l'indépendance

Kampé de Fériet (Kampé de Fériet, 1975) définit deux observateurs comme indépendants si la valeur donnée par l'un ne permet en rien de connaître la valeur que donnera l'autre :

Définition 5 (Indépendance) *Étant donné un système dont les états sont représentés par les points ω d'un espace des phases Ω, soient Ω_1 et Ω_2 l'ensemble des points ω_1 et ω_2, que les observateurs 1 et 2 font respectivement correspondre à l'état ω ; les observateurs sont* indépendants*, si la connaissance de $\omega_1 \in \Omega_1$ laisse complètement arbitraire la localisation de ω_2 dans Ω_2 et réciproquement.*

Kampé de Fériet complète cette définition par la remarque que *l'indépendance de[s] deux événements [...] est une conséquence évidente de l'indépendance des observateurs.*

Dans notre cadre, chaque arbre de décision définit une surface de séparation entre une classe et les autres, il est donc clair que chaque arbre définit une surface différente

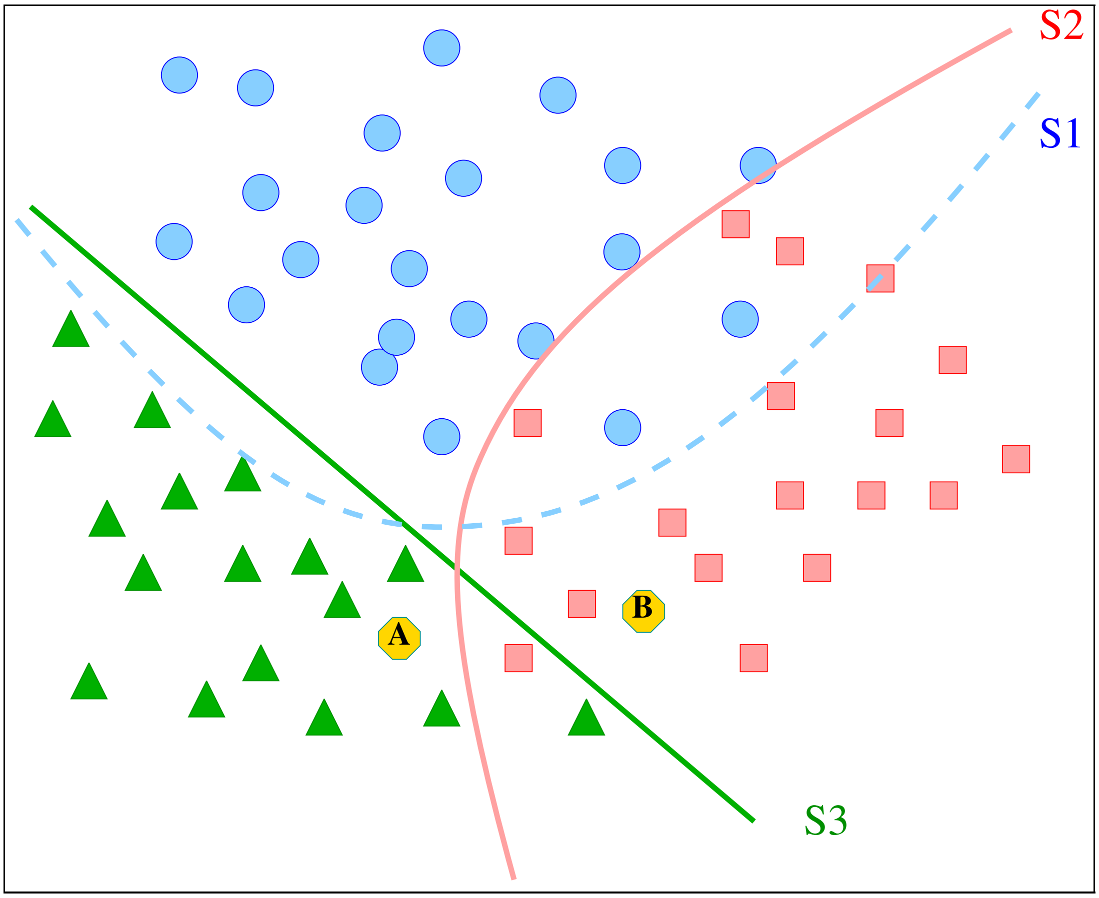


Figure 5.3 – *Surfaces de séparation*

des autres. En effet, si deux arbres de décision définissaient le même hyperplan, cela signifierait

- soit qu'il n'existe que deux classes dans la base.
- soit que deux classes sont confondues.

En ce sens, comme chaque arbre de décision flou définit une surface, la valeur qu'elle associera à un exemple (d'un côté ou d'un autre de la surface de séparation) ne permettra en rien de connaître la valeur que donnera un autre arbre pour le même exemple. Chaque arbre de décision flou est, par conséquent, indépendant des autres au sens de Kampé de Fériet.

Par exemple, dans la Figure 5.3, dans cet univers à trois classes (les ronds, les carrés et les triangles), trois arbres de décision flous seront construits, chacun définissant une surface de séparation entre une classe et les deux autres : S_1 pour séparer les ronds des autres classes, S_2 pour séparer les carrés et S_3 pour séparer les triangles. Pour l'exemple A (représenté par un hexagone car sa classe n'est pas encore connue) à classer, S_1 le rejettera comme n'étant pas un rond, mais cela ne nous donnera aucune information sur ce que déduiront S_2 et S_3 pour cet exemple. Ici, A sera accepté comme un triangle par S_3 et rejeté comme un carré par S_2. Mais si on compare avec l'exemple B, qui lui aussi est rejeté par S_1, le résultat de S_2 et de S_3 est inversé. De la même façon, un exemple qui serait accepté par S_1 ne laisserait rien préjuger de son acceptation par S_2 et S_3.

Ainsi, les arbres d'une forêt étant reconnus comme indépendant, un grand nombre de méthodes d'agrégation pourront être utilisées pour agréger les résultats de classification de chaque arbre.

Dans la prochaine section, nous présentons le système Tanit de construction d'arbres de décision flous. Pour sa phase de classification et l'agrégation des degrés d'appartenance venant de chaque arbre, Tanit utilise les trois méthodes présentées dans cette section. La partie sur les résultats de ces différentes méthodes sur de bases de test permettront alors de comparer leurs avantages respectifs.

5.4 L'application *Tanit*

La méthode de construction de forêt d'arbres de décision flous est implémentée dans l'application *Tanit*. Chaque arbre de décision flou est construit par une application Salammbô, décrite au chapitre 2.

Ensuite, Tanit utilise différentes méthodes d'agrégation pour combiner les résultats en classification de chaque arbre pour déterminer la classe d'un nouvel exemple.

5.4.1 *Tanit* pour construire une forêt d'arbres de décision flous

Tanit supervise tout le processus d'apprentissage. Elle est chargée de créer autant de Salammbô qu'il y a de classes à apprendre dans la base d'apprentissage. À sa création, chaque Salammbô reçoit de Tanit le nom d'une classe. C'est cette classe qu'elle devra reconnaître, toutes les autres classes étant pour elle mélangées en une seule classe. Chaque Salammbô construit un arbre de décision flou avec la base d'apprentissage donnée en ne considérant qu'une classe contre toutes les autres.

Un ensemble de classes $\{c_1, c_2 ... c_n\}$ se binarise de n façons différentes $\{c_1, \overline{c_1}\}$, $\{c_2, \overline{c_2}\}...\{c_n, \overline{c_n}\}$ où $\overline{c_i}$ représente l'union des classes différentes de c_i. Cela revient donc à apprendre, pour toutes les classes possibles, une classe contre toutes les autres.

Ainsi, à partir d'une base d'apprentissage $\mathcal{E}$ à apprendre, Tanit procède en trois phases :

- Tout d'abord, Tanit prend connaissance du nombre de modalités de la classe dans la base d'apprentissage $\mathcal{E}$.
- Ensuite, Tanit crée autant d'applications Salammbô que nécessaire, en ordonnant à chaque Salammbô de construire, à partir de $\mathcal{E}$, un arbre de décision flous reconnaissant une modalité c_k de la classe contre toutes les autres modalités réunies.
- Finalement, Tanit se met en attente de la terminaison de toutes les Salammbô.

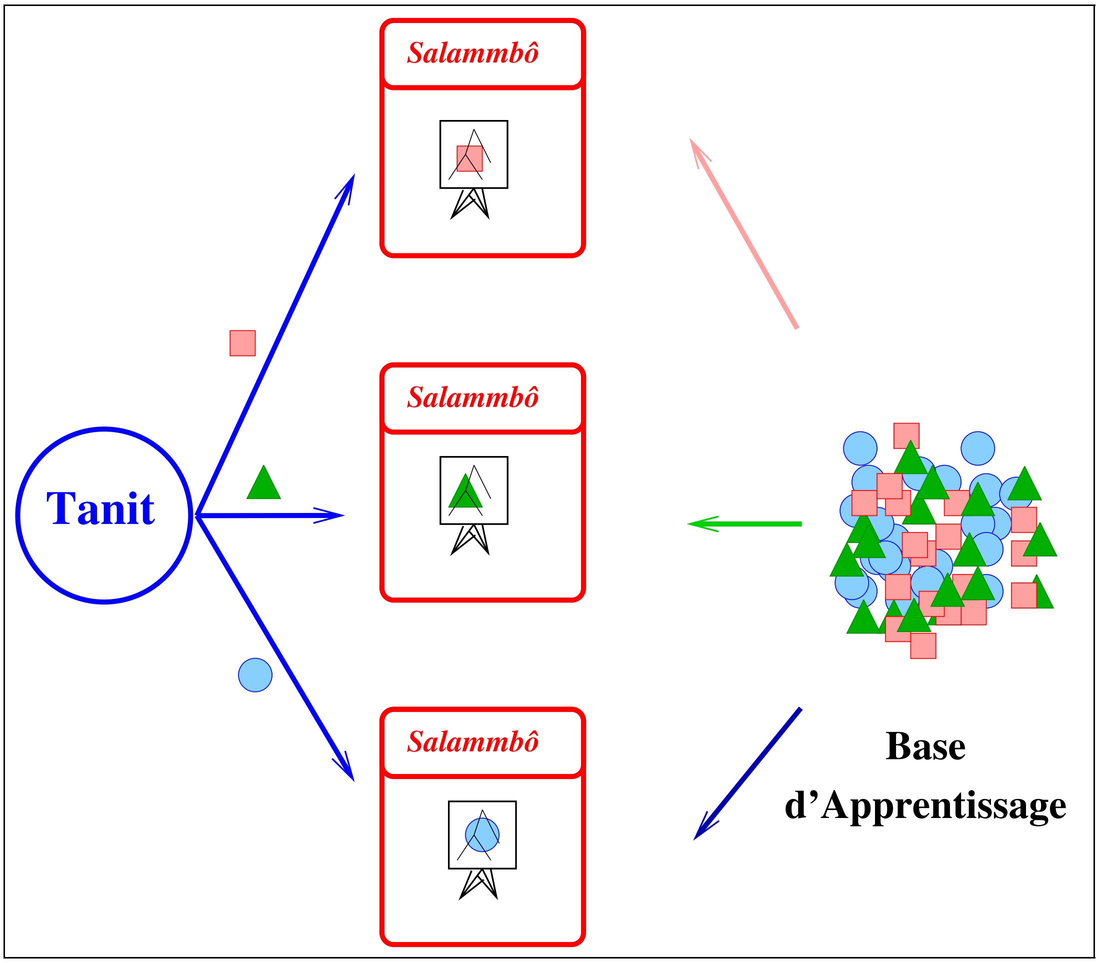


Figure 5.4 – *Tanit : phase d'apprentissage*

Par exemple, dans le cas le plus simple, c'est-à-dire dans le cas où il y a trois classes à reconnaître dans la base (par exemple des carrés, des ronds et des triangles (Figure 5.4)), Tanit crée trois Salammbô, une pour construire un arbre de décision flou qui reconnaîtra la classe des carrés, une autre pour la classe des ronds et la dernière pour la classe des triangles.

Dans le cas où des exemples inconnus sont à classer, pour chaque exemple, chaque Salammbô retourne à Tanit un degré d'appartenance à la classe qu'elle a appris à reconnaître, et un degré d'appartenance aux autres classes. Tanit se charge alors d'agréger ces degrés pour en déduire la classe de l'exemple. Cette phase d'agrégation est décrite en détails dans la section suivante.

5.4.2 *Tanit* pour agréger les degrés donnés par les Salammbô

Dans cette phase de classement, chaque Salammbô sait classer les exemples en fonction de la classe qu'elle a appris à reconnaître, en déterminant des degrés d'appartenance aux classes. Par un arbre de décision flou, un degré d'appartenance à une classe est obtenu pour chaque exemple à classer comme il l'a été précisé dans le paragraphe 2.4. Ainsi, pour chaque exemple, chaque Salammbô détermine un degré d'appartenance $m_{c_i}(\{c_i\})$ à la classe c_i qu'elle est chargée de reconnaître, ainsi qu'un degré d'appartenance $m_{c_i}(\{\overline{c_i}\})$ aux classes différentes $\overline{c_i}$.

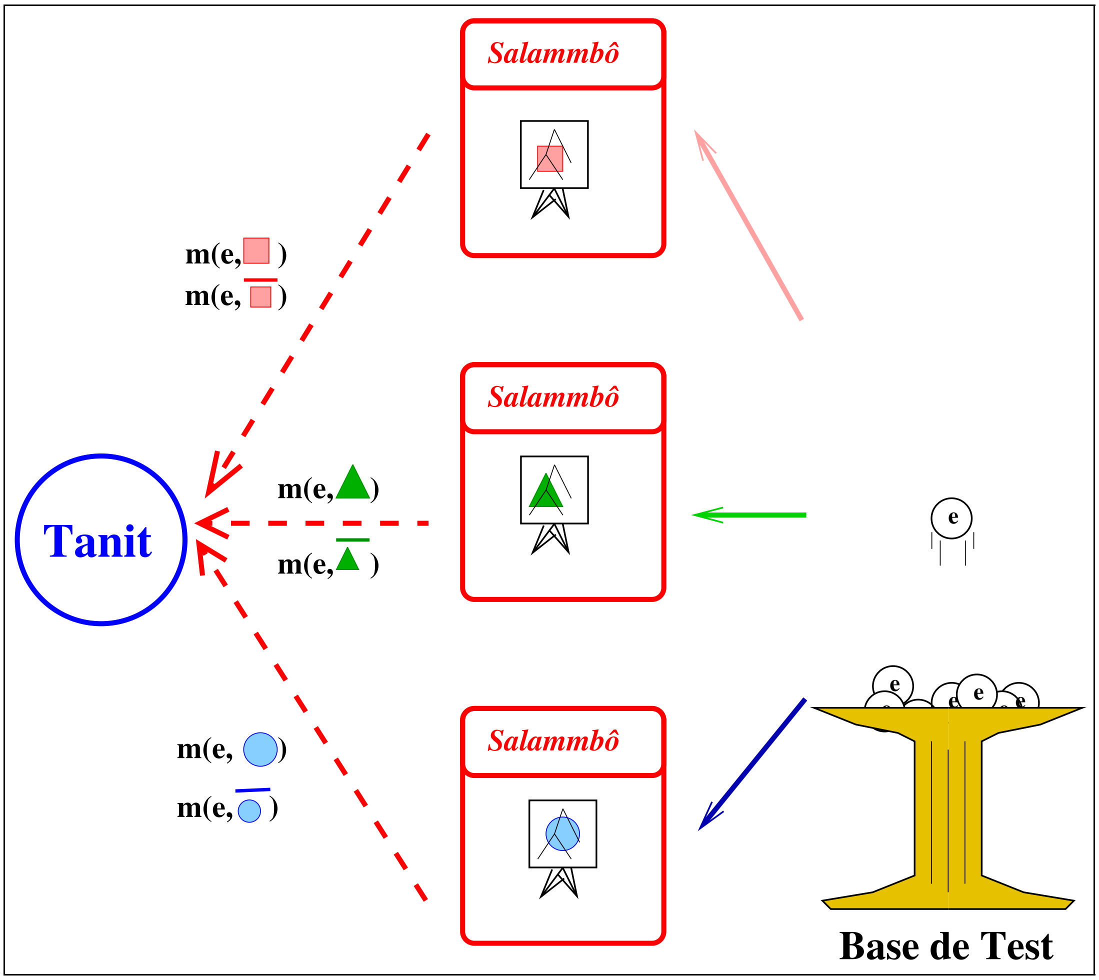


Figure 5.5 – *Tanit : phase de classification*

Par exemple, dans le cas de la Figure 5.5, pour un nouveau cas à classer, la Salammbô chargée de reconnaître les carrés retourne son degré d'appartenance à la classe des carrés ainsi que son degré de rejet de la classe des carrés. Elle renvoie alors ces deux degrés à Tanit qui centralise tous les degrés d'appartenance qui lui sont donnés par chacune des Salammbô. Tanit se charge alors d'agréger les résultats renvoyés pour déterminer la classe à associer à l'exemple à classer (*cf.* Table 5.1).

L'agrégation des degrés donnés par les Salammbô

Chaque Salammbô étant reconnue comme indépendante des autres, le choix de la méthode d'agrégation la plus appropriée sera plus grand. En dehors de la méthode du vote, des méthodes comme celle de Dempster pourront aussi être utilisée pour agréger les données des Salammbô. Il est important de remarquer que cette indépendance entre les classifieurs est une condition primordiale pour pouvoir utiliser cette règle de combinaison bien que cette théorie de l'évidence ne fournisse pas de définition précise de l'indépendance impliquée.

Quelques exemples de résultats agrégés sont donnés dans les tableaux 5.1 à 5.4. Dans ces tableaux, les résultats venant de trois Salammbô sont agrégés pour déterminer la classe d'un exemple donné. Les méthodes d'agrégation utilisées varient afin de remarquer le comportement de chaque méthode sur quelques valeurs particulières.

Pour les résultats présentés, chaque Salammbô retourne deux degrés d'appartenance aux classes. Un degré pour la classe reconnu par la Salammbô et un degré d'appartenance aux autres classes. Pour les résultats donnés Table 5.1, ces deux degrés sont normalisés.

Ex.	*Arbre c1*		*Arbre c2*		*Arbre c3*		*Vote*			*Dempster*		
	c1	$\overline{c1}$	*c2*	$\overline{c2}$	*c3*	$\overline{c3}$	*c1*	*c2*	*c3*	*c1*	*c2*	*c3*
e1	1	0	0	1	0	1	**2**	0,5	0,5	**1**	0	0
e2	0,9	0,1	0,3	0,7	0,4	0,6	**1,55**	0,65	0,8	**0,38**	0,02	0,03
e3	1	0	0,89	0,11	0	1	**1,55**	1,39	0,06	**0,11**	0	0

Table 5.1 – *Agrégation des résultats*

Il est à noter que l'agrégation par la méthode de Dempster donnent des résultats très prononcés dans le sens où, dès qu'une Salammbô a retourné une valeur nulle pour une classe, le degré final associé à cette classe est nul, quels que soient les degrés donnés par les autres Salammbô. Dans une certaine mesure, cela engendre une décision trop catégorique mais c'est aussi le sens de cette règle qui est basée sur l'utilisation du produit qui est une norme triangulaire.

Ex.	*Arbre c1*		*Arbre c2*		*Arbre c3*		*Vote*		
	c1	$\overline{c1}$	*c2*	$\overline{c2}$	*c3*	$\overline{c3}$	*c1*	*c2*	*c3*
e1	1	0	0	0,3	0	0,8	**1,55**	0,4	0,15
e2	0,55	0,06	0,3	0,7	0,32	0,48	**1,14**	0,57	0,7
e3	0,1	0	0,8	0,1	0	0,6	0,45	**1.1**	0.05

Table 5.2 – *Agrégation de résultats non normalisés*

Pour les deux méthodes présentées dans cette table, on peut aussi remarquer que la condition de normalisation oblige à ne pas tenir compte de l'écart de reconnaissance entre une classe et son complément par une même Salammbô.

Les résultats suivants (Table 5.2) permettent d'étudier le comportement de la méthode d'agrégation du vote quand les degrés retournés par les Salammbô ne sont pas normalisés. Dans ce cas, les résultats sont plus nuancés car l'écart entre les degrés retournés est alors pris en compte.

Les résultats donnés par la méthode de la fonction de croyance unique sont aussi très instructifs. Tout d'abord, pour mettre en œuvre cette méthode, il faut normaliser tous les degrés entres eux (Table 5.3). Ensuite (Table 5.4), les fonctions de croyance et de plausibilité sont calculées et la décision finale – le degré de décision final associé à chaque classe – est calculée en prenant la valeur moyenne entre les deux degrés de croyance et de plausibilité. Encore une fois, il est à noter que les résultats obtenus pour la décision sont plus nuancés et permettent de rendre encore mieux compte des nuances des décisions de chaque Salammbô.

Ex.	*Arbre c1*		*Arbre c2*		*Arbre c3*	
	c1	$\overline{c1}$	*c2*	$\overline{c2}$	*c3*	$\overline{c3}$
e1	0,48	0	0	0,14	0	0,38
e2	0,23	0,02	0,12	0,29	0,13	0,2
e3	0,06	0	0,5	0,06	0	0,38

Table 5.3 – *Normalisation globale des degrés*

Ex.	*Fonction de croyance*						*Décision*		
	Bel(c1)	*Pl(c1)*	*Bel(c2*	*Pl(c2*	*Bel(c3*	*Pl(c3*	*c1*	*c2*	*c3*
e1	0,48	1,0	0	0,38	0	0,14	**0,74**	0,19	0,07
e2	0,23	0,72	0,12	0,34	0,3	0,61	**0,47**	0,23	0,45
e3	0,06	0,5	0,5	0,88	0	0,06	0,28	**0,69**	0,03

Table 5.4 – *Agrégation de résultats en termes de fonctions de croyance*

Dans certain cas, comme pour l'exemple *e3* étudié, la normalisation des degrés amène une décision différente mais qui parait rendre plus parfaitement compte de la valeur avec laquelle chaque Salammbô croit vraiment en l'appartenance aux classes.

Pondérer les Salammbô

Chaque Salammbô peut aussi être associée à un degré de fiabilité utilisable pour pondérer ce que dit cette Salammbô. En l'absence de moyens cohérents pour distinguer la fiabilité des Salammbô les unes par rapport aux autres, il faut reconnaître qu'elles doivent être aussi fiables les unes que les aux autres, elles ont donc un degré de fiabilité égal à 1. Mais, il est possible de décider d'affiner cette fiabilité des Salammbô en considérant qu'elle doit dépendre aussi de la proportion des individus de chaque classe dans la base d'apprentissage. Une classe fortement représentée dans la base d'apprentissage peut donner une grande confiance dans la capacité de reconnaissance de la Salammbô qui est chargé de la reconnaître. Ainsi, par exemple, si une classe représente 90% de la base, il peut paraître intuitivement normal d'allouer une fiabilité plus grande à la Salammbô chargée de la reconnaître.

Une autre méthode de pondération possible, citée par (Bernhardt et al., 1997) dans le cadre des réseaux de neurones, est d'utiliser le taux d'erreur en apprentissage du classifieur comme degré de confiance.

5.5 Résultats

L'application générique Tanit a été implémentée en langages C/C++ et Lex sous le système d'exploitation UNIX. Son code est relativement court (environ 900 lignes). Quelques fonctions principales sont données en annexe E.

En fait, après avoir pris connaissance du nombre de classes dans la base d'apprentissage, Tanit se contente de lancer, à l'aide de la primitive `fork()`, autant de Salammbô que de classes. Chaque Salammbô reçoit en paramètre la classe qu'elle doit apprendre. Ensuite, Tanit attend de chaque Salammbô le résultat de la classification de la base de test et agrège les résultats donnés en utilisant différentes méthodes d'agrégation. Finalement, Tanit affiche le résultat de la classification obtenu après agrégation.

La version actuelle de Tanit a été testée, en validation croisée, sur plusieurs types de bases d'apprentissage (disponibles sur le site `ftp`[6] de l'université d'Irvine (Californie, USA). Les bases utilisées sont celles qui comprennent une forte proportion de données numériques afin de tester aussi les apports des arbres de décision flous.

Les résultats donnés pour toutes les méthodes d'agrégation sont basés sur l'utilisation d'un même arbre mais pour lequel la classification d'exemples a été considérée soit classiquement, soit d'une façon floue. Dans le cas flou, la méthode d'inférence pour classer un exemple dans l'arbre utilise, dans le cas des formes d'ondes, les opérateurs de Łukasiewicz, et dans le cas des Iris, les opérateurs de Zadeh, qui ont permis d'obtenir les meilleurs résultats, d'après des tests préliminaires.

Méthode	*Taux d'erreurs en %*	
	Non flou	*Méthode floue*
Arbre Unique	27,3	23,7
Agrégation par Vote, non normalisé	26,7	**20,1**
Agrégation par Vote, normalisé	26,7	25,6
Agrégation par Dempster	28,5	27,2
Agrégation par Évidence	26,7	**20,1**

Table 5.5 – *Résultats sur les formes d'ondes de Breiman*

Les premiers résultats sont ceux obtenus sur la base des formes d'ondes de Breiman (Breiman et al., 1984) (Table 5.5[7]). Dans ces données, les descriptions de chaque exemple comportent 21 attributs numériques et il y a trois classes à reconnaître.

Ces résultats montrent que les méthodes basées sur des arbres de décision flous sont toujours meilleures que celles basées sur des utilisations non floues des arbres. Dans ce

6. `ftp://ftp.ics.uci.edu/pub/machine-learning-databases/`

7. Comme résultats de techniques de construction d'arbres de décision classiques (ligne *Arbre Unique*, colonne *Non flou*), on cite les résultats donnés dans (Quinlan, 1996) pour l'algorithme C4.5 adaptés aux attributs numériques

domaine où l'on recherche à agréger des opinions de classifieurs, il est parfaitement naturel que l'agrégation d'opinions nuancées soit plus efficaces que l'agrégation d'opinions catégoriques. Les meilleurs taux de classification (*taux d'erreur minimal*) sont obtenus par deux type d'agrégation : la méthode du vote quand les degrés ne sont pas normalisés en sortie de chaque arbre, et la méthode de calcul basée sur la théorie de l'évidence (calcul des fonctions de croyance et de plausibilité et la décision est calculée en faisant la moyenne des deux valeurs). Bien que donnant le même taux d'erreur, ces deux méthodes apportent chacune leurs qualités et leurs défauts, ce qui est légèrement perceptible en observant leur écart type respectif : la méthode du vote (non normalisé) donne un écart type de $1,4$ alors que la méthode de la théorie de l'évidence donne un écart type de $1,3$. L'écart entre les taux d'erreur des deux méthodes basées sur le vote est aussi à noter. La méthode dans laquelle les degrés sont normalisés en sortie des Salammbô donne de plus mauvais résultats que la méthode qui ne les normalise pas. Comme il a été évoqué dans la partie sur les méthodes d'agrégation, la normalisation des degrés en sortie du classifieur leur fait perdre une information relative à la nuance de la décision prise par ledit classifieur. L'écart entre les deux classes est perdu, ce qui, dans le cas où les degrés vont être agrégés par la suite avec des degrés venant d'autres classifieurs, fait perdre la nuance de la décision retournée.

D'autres résultats sont donnés pour la base de reconnaissance des iris de Fischer (Table 5.6). Dans ces données, les descriptions de chaque exemple comportent 4 attributs numériques et il y a trois classes à reconnaître.

Méthode	*Taux d'erreurs en %*	
	Non flou	*Méthode floue*
Arbre Unique	4,8	4,7
Agrégation par Vote, non normalisé	4,7	**4,0**
Agrégation par Vote, normalisé	4,7	**4,0**
Agrégation par Dempster	5,3	4,7
Agrégation par Évidence	4,7	**4,0**

Table 5.6 – *Résultats sur la base des iris*

Dans cette base d'exemples, les deux types de méthodes basées sur le vote et la méthode basée sur la théorie de l'évidence donnent les meilleurs résultats avec la même valeur d'écart type. En fait, ce type de base peut être considéré comme trop petit pour arriver à bien différencier de telles méthodes.

En conclusion, on peut relever que, dans les deux cas, les meilleurs résultats en classification sont obtenus grâce à la méthode du vote appliquée aux degrés de classification fournis par les Salammbô utilisant une méthode de classification floue.

5.6 Complexité d'un système basé sur une forêt d'arbres de décision

Pour la constitution d'une forêt d'arbres de décision, la multiplication de la construction des arbres peut laisser supposer que la complexité du processus d'apprentissage augmentera d'autant.

Cependant, comme le souligne déjà (Friedman, 1977), si le nombre d'arbres à construire augmente, la taille de chaque arbre diminue. Un arbre capable de différencier deux classes est plus réduit – et donc plus rapide à construire – qu'un arbre capable de différencier plus de deux classes. Certes, cela ne suffit pas à limiter radicalement la complexité de ce nouveau processus d'apprentissage, mais, en fait, il faut aussi envisager la *parallélisation* – on dit aussi *distribution* – de la construction des arbres pour arriver à en profiter pleinement.

Ainsi, étant donné qu'avec les moyens informatiques actuels, les ordinateurs peuvent être connectés les uns aux autres, Tanit est conçue pour être une application distribuée sur plusieurs machines. Le but d'un tel système est alors de faire exécuter chaque Salammbô sur une machine distincte du réseau de machines disponibles. Dans un réseau de n machines, la construction de n arbres s'effectue ainsi en un temps équivalent au temps nécessaire à une seule machine pour construire un seul arbre. Le temps supplémentaire pris pour les échanges de données entre les machines est quant à lui compensé par la diminution de la taille des arbres à construire. Les échanges de données entre les Salammbô et l'application-maître Tanit peuvent s'effectuer par les services de protocoles usuels de la programmation distribuée, simples à mettre en œuvre comme par exemple, le protocole de communication RPC[8] disponible sur un réseau de machines UNIX (Rifflet, 1992).

En Intelligence artificielle, la programmation distribuée donne lieu à des travaux de recherche spécifiques. De tels programmes qui évoluent en parallèle et qui sont appelés à collaborer entre eux pour la résolution d'une tâche commune constituent les *agents* d'un système multi-agents (Guessoum, 1996).

Le concept de Tanit en tant qu'architecture distribuée est issue, à l'origine, d'une analogie avec de telles architectures multi-agents. Tanit et chaque Salammbô peuvent être considérées comme des agents informatiques évoluant dans un univers multi-agents qui concourent à la réalisation d'un même objectif qui est de classer au mieux de nouveaux exemples après avoir appris à reconnaître des classes. L'agent Tanit représente le superviseur général de cet univers, c'est elle qui est le lien entre tous les agents Salammbô et qui coordonne le système tout entier. Durant la phase d'apprentissage, Tanit crée autant d'agents Salammbô que de classes dans la base d'apprentissage. Durant la phase de classification, ces agents retournent à Tanit des informations concernant les exemples à classer et leur opinion personnelle sur chacun de ces exemples.

L'aspect rudimentaire de ce système peut être encore améliorer par l'utilisation de communications plus élaborées entre les Salammbô. De telles communications sont un moyen d'échanger des informations, par exemple durant la phase d'apprentissage, pour choisir un attribut dans des cas ambigus. De même, durant la phase de classification, il peut être alors possible d'autoriser une correction des arbres construits par les Salammbô. Chaque agent Salammbô apprend à mieux classer les exemples en utilisant les informations communiquées par les autres Salammbô, l'agent Tanit pouvant encore servir d'intermédiaire général pour centraliser toutes les informations du système.

Finalement, la forêt ne fait pas perdre le pouvoir explicatif obtenu par les arbres de

8. Remote Procedure Call

décision. Comme il a été dit, les arbres de la forêt sont souvent plus petits que l'arbre global, les règles qui en sont déduites sont donc plus réduites en nombre de prémisses. Quoi qu'il en soit, la forêt donne aussi, comme dans le cas d'un arbre unique, une base de règles floues.

5.7 Conclusion

Dans ce chapitre, une méthode d'apprentissage par une forêt d'arbres de décision flous a été présentée.

Cette méthode est basée sur la construction de plusieurs arbres de décision flous, chacun ayant pour tâche de reconnaître une classe particulière contre toutes les autres classes de la base d'apprentissage initiale. Il est intéressant de noter que la combinaison de tels arbres permet de faire ressortir le caractère discriminant d'un attribut pour une classe et non plus un ensemble de classes. Ainsi, les attributs racines des arbres obtenus sont souvent différents en fonction de la classe à reconnaître.

De plus, l'introduction de nouveaux types de classifieurs dans la forêt est aussi à considérer. En plus des Salammbô pour construire des arbres de décision flous, l'application générique Tanit a la possibilité de lancer d'autres types de classifieurs sur les mêmes bases d'apprentissage et de test. De plus, même au niveau des Salammbô, Tanit est capable d'introduire dans sa forêt des arbres de décision qui ne discriminent pas obligatoirement deux classes. Ainsi, la multiplication des classifieurs de tous types peut apporter une richesse dans les traitements de la base d'exemples et améliorer encore les résultats en classification.

Il reste encore de nombreux points pour améliorer cette technique. En particulier, la méthode d'agrégation des résultats est une étape importante qui mérite une étude approfondie afin de donner de meilleurs résultats.

Finalement, cette méthode permet, dans le cadre de l'étude de la meilleure mesure de discrimination à utiliser pour construire un arbre de décision, de ne se consacrer qu'à l'étude des propriétés d'une telle mesure nécessaires pour discriminer un ensemble à deux classes, sans avoir à étudier des problèmes où le nombre de classes est supérieur à deux.

Chapitre 6

Une nouvelle méthode d'étude des mesures de discrimination

Delenda Carthago!

Caton l'ancien – *Discours*

6.1 Introduction

Comme il a été vu au chapitre 2 sur la construction des arbres de décision flous, les différentes méthodes se différencient essentiellement par la mesure de discrimination qu'elles utilisent. Choisir une méthode revient donc à choisir une mesure de discrimination, c'est la partie prépondérante de l'algorithme de construction.

Les recherches usuelles sur les mesures de discrimination permettent souvent de déterminer si une mesure est une bonne mesure de discrimination mais ne permettent pas forcément de comparer deux mesures de discrimination entre elles.

De plus, les mesures de discrimination actuellement utilisées dans la construction des arbres de décision sont des mesures issues de la théorie de l'information, comme l'entropie de Shannon dans le système ID3, ou bien des mesures issues des statistiques, comme le critère de Gini dans le système CART. Il est indéniable que ces mesures ont leurs avantages, mais, souvent, essentiellement dans un cadre symbolique. En présence de données numériques, des limites apparaissent. L'entropie de Shannon favorise les attributs avec un grand nombre de valeurs, même si ces valeurs sont peu informatives, comme le sont généralement des valeurs numériques. Les mesures statistiques ne rendent pas compte de l'aspect informationnel des données.

En présence de données numériques-symboliques ou de données numériques associées à une partition floue, les mesures actuelles ne peuvent pas rendre compte de l'apport d'information qui est donnée par la partition floue sur l'univers des valeurs. L'entropie étoile est une tentative pour étendre l'entropie classique à la prise en compte de données numériques associées à une partition floue. Mais il est alors difficile de donner les raisons pour utiliser cette mesure plutôt qu'une autre. En règle générale, on ne sait pas justifier l'emploi de telle ou telle mesure de discrimination. L'entropie de Shannon est en cela une mesure très intéressante dans le sens où Shannon et d'autres théoriciens de l'information ont justifié son emploi et ses fondements en tant que mesure d'information. Dans le cas d'autres mesures, la justification de leur emploi est moins nette et se confond souvent avec des critères empiriques ou intuitifs. Dans le cadre flou, les mesures issues spécifiquement de la théorie de l'information ne rendent pas forcément compte de l'aspect flou des données.

Dans ce chapitre, nous étudions des méthodes de classification de mesures. À la différence des méthodes de validation présentées au chapitre 2, les méthodes de ce chapitre permettent de comparer les mesures en examinant leurs propriétés de façon plus sémantique.

Ce chapitre est composé comme suit : après une présentation de quelques travaux existants sur l'étude des mesures, appelées encore mesures d'information, nous étudions une méthode plus particulière qui est la théorie des modèles entropiques de (Bouchon, 1985). Ensuite, nous proposons une nouvelle méthode d'étude de mesures, appelées alors mesures de discrimination, basée sur l'élaboration d'une hiérarchie de fonctions. Cette méthode constructive, est appliquée à l'examen de deux mesures usuelles de discrimination, et son emploi pour la construction d'une nouvelle mesure est développé. Une extension de cette hiérarchie pour les mesures prenant en compte les données numériques-symboliques est donnée. À l'aide de cette hiérarchie, une comparaison de deux méthodes d'apprentissage inductif, les algorithmes de construction d'arbres de décision et les méthodes de construction de prototypes, est faite.

Finalement, la dernière partie de ce chapitre est consacrée à de nouvelles perspectives concernant des mesures de discrimination floues.

6.2 Méthode existante d'étude des mesures d'information

6.2.1 Les mesures d'information

Depuis (Shannon, 1948), les mesures d'information ont été énormément étudiées et elles ont même donné lieu au développement de la théorie de l'information. Une bonne introduction aux mesures d'information est peut-être l'article de (Csiszár, 1978) qui présente les mesures d'information les plus courantes.

Les mesures d'information, et en particulier la mesure d'entropie de Shannon, ont été connectées au domaine des inférences inductives et déductives très tôt dans l'ère de la théorie de l'information (Watanabe, 1960). Dans cet important article, on note déjà l'utilisation d'une mesure d'information pour mesurer le pouvoir prédictif[1].

On note Π un système d'événements, et H une mesure d'information sur ce système d'événements comme la mesure d'entropie de Shannon. Par sa définition initiale, $H(\Pi)$ est une mesure *a priori* sur le désordre potentiel du système Π.

Le terme *désordre* est très approprié à ce qu'il est censé refléter : le système n'est connu qu'éventuellement par les probabilités d'occurrences de chacun des événements possibles qui le composent. Plus il existe des événements dans Π sensés se produire, plus le système devient imprévisible, et par analogie avec la théorie physique des particules, plus le taux de désordre du système augmente. On comprend aisément pourquoi une

1. "*In this sense, $J(I)$ may be called the "predictive information content" or "predictive power" of hypothesis H_t*" (Watanabe, 1960). (Remarque: dans cet article, $J(I) = \log n - H_S(I)$.)

mesure comme l'entropie de Shannon est utilisée pour rendre compte de la prévisibilité d'un système d'événements.

En thermodynamique, l'entropie de Boltzman[2] rend compte du taux de désordre d'un système physique, c'est-à-dire de la façon dont il est possible de prévoir le devenir de ce système. De même, l'entropie de Shannon est une mesure qui est censée rendre compte du taux de désordre d'un système d'événements probabilistes, en termes de prévision d'occurrence de chaque événement élémentaire.

L'ouvrage de (Aczél et Daróczy, 1975) fait une étude complète de cette mesure d'entropie dans le domaine de la théorie de l'information. Les différents travaux sur la théorie de l'information ont aussi montré l'unicité de cette entropie pour mesurer le taux de désordre informationnel (voir par exemple le livre de (Guiaşu et Theodorescu, 1968)).

Plusieurs extensions des mesures d'information pour la prise en compte d'événements flous ont été proposées depuis celle de Zadeh (Zadeh, 1968) (*cf.* mesure 2.11). La plus connue est l'entropie de sous-ensembles flous de (De Luca et Termini, 1972; De Luca et Termini, 1979) qui remplace la mesure de probabilité par la fonction d'appartenance d'un sous-ensemble flou dans l'équation de Shannon.

L'entropie *floue* de (Ishikawa et Mieno, 1979) étend l'entropie de Shannon en introduisant la fonction caractéristique du sous-ensemble flou en plus des probabilités dans son expression. Elle reste d'une forme proche de l'entropie de (De Luca et Termini, 1972).

Dans leur article, (Xie et Bedrosian, 1984) proposent une mesure d'entropie de sous-ensembles flous prenant en compte à la fois l'entropie de Shannon et l'entropie de (De Luca et Termini, 1972).

Il faut aussi citer les travaux de Klir sur la généralisation de la théorie de l'information et ses propositions d'entropies basées sur des mesures de la théorie de l'évidence (Klir et Folger, 1988; Klir, 1991).

Les travaux sur l'extension de la notion d'entropie aux événements flous sont encore très actuels, on peut citer par exemple (Criado et Gachechiladze, 1997) qui montrent que les entropies de Zadeh et de (De Luca et Termini, 1972) sont des formes liées à l'entropie de Shannon.

La profusion des modèles d'extension de l'entropie de Shannon ont pour effet pervers de multiplier le nombre des mesures sans permettre pour autant d'en sélectionner une dans un cas particulier d'application. La théorie des *modèles entropiques* introduite en 1985 (Bouchon, 1985) est un modèle de telle classification des mesures. Cette théorie est d'autant plus intéressante qu'elle se base une distinction naturelle entre les mesures d'information et les mesures de flou. Par l'étude des propriétés requises dans chaque cas, des grandes familles de mesures se détachent, ainsi que leurs spécificités.

2. Pour la définition de cette entropie, je renvoie aux ouvrages de physique, thermodynamique ou autres. Dans le domaine de la théorie des sous-ensembles flous, on peut citer le livre de (Klir et Folger, 1988).

6.2.2 Les différents types d'information

Comme on l'a noté dans la présentation de ce chapitre, on ne peut pas mesurer n'importe quel type d'événement avec n'importe quelle mesure. Avec une mesure d'information, on est capable de mesurer l'information apportée par un événement. Mais, en l'occurrence, l'information apportée par un événement peut être, par nature, de deux ordres (Bouchon-Meunier, 1989) :

- l'information d'*observation*, qui est intrinsèque aux données appréhendées par un observateur. Elle est liée à leur représentation. La mesurer revient à *évaluer* la qualité des données. C'est la *forme* des données.
- l'information d'*exploitation*, qui permet de prendre une décision. C'est l'information classique, au sens d'Hartley et de Shannon. On veut arriver à connaître l'utilisation qui peut être faite des données disponibles, indépendamment de leur contenu. C'est le *fond* intrinsèque aux données.

Pour mesurer le premier type d'information, on utilise généralement une mesure de flou ou une mesure de spécificité et d'imprécision. Le second type d'information est mesuré par les mesures telles que l'information de Hartley, l'entropie de Shannon, la divergence de Kullback...

Ce besoin de discerner les deux types d'informations amenées par un événement a toujours été sous-jacent à l'utilisation des mesures d'information. Déjà dans l'ouvrage de (Yaglom et Yaglom, 1959), on peut noter ce besoin de différencier deux quantités distinctes : "*...Il convient cependant d'avoir en vue que la mesure de Shannon, comme celle d'Hartley, ne peut prétendre tenir compte de tous les facteurs qui déterminent* l'incertitude d'une expérience*, dans tous les sens que l'on peut rencontrer dans la nature. Par exemple, la grandeur*[3] $H(\Pi)$ *ne dépend que des probabilités* $p(A_1), p(A_2), ..., p(A_k)$ *des différentes issues de l'expérience, mais ne dépend nullement de ces issues, du fait qu'elles soient* voisines *ou* éloignées." [4]

Plus tard, après l'introduction de la théorie des sous-ensembles flous, (Kaufmann, 1977) relèvera l'existence de deux types d'incertitudes sur des événements[5] :

- l'*incertitude du* 1^{er} *type*, qui est de nature aléatoire et concerne la prévision du résultat. Elle est mesurable par l'entropie de Shannon.
- l'*incertitude du* 2^{nd} *type*, qui concerne l'interprétation du résultat en valeur 0 ou 1. Elle est mesurable par l'indice de flou de (De Luca et Termini, 1972).

Il est donc important d'utiliser une mesure d'information en adéquation avec le type d'information que l'on a à gérer. Pour cela, il nous faut donc exhiber les facteurs déterminant chaque type d'information et étudier les propriétés que cela entraîne pour la mesure d'information adéquate. Il faut établir une classification des mesures en fonction du type d'information que l'on souhaite mesurer, afin de pouvoir choisir, en toute connaissance de causes, la mesure la plus adaptée au problème à résoudre.

3. L'entropie de Shannon
4. (Yaglom et Yaglom, 1959), p. 44
5. (Kaufmann, 1977), p. 49

6.2.3 Les modèles entropiques

La théorie des modèles entropiques a pour origine le constat de l'existence de ces deux types d'information pour en tirer une méthode de classification des mesures existantes. De plus, elle veut aussi permettre la validation de mesures utilisées pour répondre à un problème donné. L'intérêt des modèles entropiques est donc de fournir un cadre général permettant de classer les mesures, d'identifier leur nature et d'en recenser les différents cas d'utilisation.

Deux types de modèles entropiques (Bouchon, 1985), (Bouchon, 1988) sont définis :

- les *modèles entropiques de type 1*, qui peuvent être considérés pour mesurer l'information contenue dans les résultats d'une expérience en dépit de l'incertitude intervenant durant l'observation. Ces modèles prennent en compte les choix des caractéristiques utilisées pour décrire une observation, leur *finesse* (en anglais : *sharpness*), leurs divisions possibles ou leurs regroupements. *Ils décrivent un point de vue informationnel.*

- les *modèles entropiques de type 2*, qui mesurent l'incertitude intervenant durant l'observation des résultats d'une expérience. Ces modèles mesurent la qualité, l'exactitude des caractéristiques données à une observation. *Ils expriment une étude de l'imprécision.*

La définition formelle d'un modèle entropique est rappelée dans l'Annexe C.

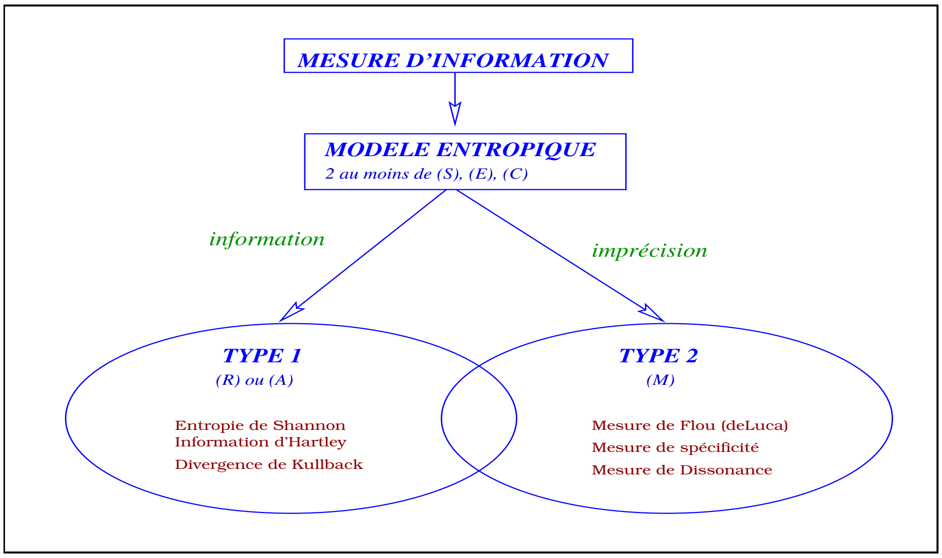


Figure 6.1 – *Les différents types de modèles entropiques*

Les trois propriétés (*cf.* Annexe C) de symétrie (S), d'expansiabilité (E) et de continuité (C) sont des propriétés qui peuvent être considérées comme élémentaires. Elles jugent de la pertinence du modèle choisi comme mesure d'information au sens général.

Les trois autres propriétés de récursivité (R), d'additivité (A) et de monotonie (M) sont plus sophistiquées et ce sont elles qui vont vraiment permettre de différencier les deux types de modèle entropique.

En fonction de ces propriétés, on en déduit les définitions formelles des différents types de modèles entropiques (Figure 6.1) :

- les *modèles entropiques de type 1* satisfont au moins 2 des propriétés de symétrie (S), d'expansiabilité (E) et de continuité (C) ainsi qu'une des propriétés de récursivité (R) ou d'additivité (A).

- les *modèles entropiques de type 2* satisfont au moins 2 des propriétés symétrie (S), d'expansiabilité (E) et de continuité (C) ainsi que la propriété de monotonie (M).

On peut de suite remarquer qu'une mesure peut très bien vérifier une des deux propriétés (R) ou (A) et, simultanément, la propriété (M). Ainsi, un modèle entropique peut appartenir aux deux types définis ci-dessus et il permettra alors de mesurer à la fois l'information d'observation et l'information d'exploitation contenue dans un événement. C'est typiquement ce que l'on attend de l'entropie utilisée pour construire des arbres de décision flous. Elle doit rendre compte du flou des données et de l'information en terme de classement qu'elles renferment, ce que réalise, par exemple, l'entropie étoile.

6.3 Une nouvelle modélisation des mesures de discrimination : une approche constructive

La vision de Kampé de Fériet

Dans ses travaux précurseurs, Joseph Kampé de Fériet a étudié longuement les mesures d'information et a introduit une optique complètement différente de la façon de voir habituelle. C'est son modèle de vision que nous adoptons ici.

En 1967, dans (Kampé de Fériet et Forte, 1967), Kampé de Fériet écrit "*...il nous paraît plus difficile d'analyser notre intuition de l'information pour $H(\Pi)$ que pour $J(A)$.*" [6] (*cf.* Figure 6.2). De cette constatation, il recommande alors d'étudier dans un premier temps la mesure $J(A)$, afin de remonter les résultats de cette étude pour en déduire une analyse de la mesure plus complexe $H(\Pi)$.

Dans la théorie de l'information, les méthodes d'étude des mesures d'information examinent toutes la forme $H(\Pi)$ de la mesure d'information. De la même manière, les méthodes de validation des mesures de discrimination à utiliser dans un processus de construction d'arbres de décision, s'appuient sur l'examen des propriétés requises pour ces mesures. Ainsi, les propriétés demandées par (Breiman et al., 1984), (Zighed et al., 1991) ou (Rabaséda-Loudcher, 1996) portent essentiellement sur la forme $H(\Pi)$. C'est peut-être la raison qui conduit à utiliser comme mesure de discrimination une mesure d'information, qui, par nature, possède les propriétés requises.

Mais, comme Kampé de Fériet le préconise, il peut être intéressant et instructif d'étudier les propriétés requises pour la forme $J(A)$ de la mesure d'information. C'est ce que nous nous proposons de faire dans ce chapitre. Sans vouloir remettre en cause les méthodologies d'études des mesures de discrimination existantes, un nouveau type

6. Les notations utilisées sont celles de (Kampé de Fériet et Forte, 1967) : $H(\Pi)$ est la mesure d'information *a priori* du système Π composé des événements A_i ; $J(A)$ est l'information *a posteriori* apportée par l'occurrence de l'événement A, *une fois que cet événement a eu lieu.*

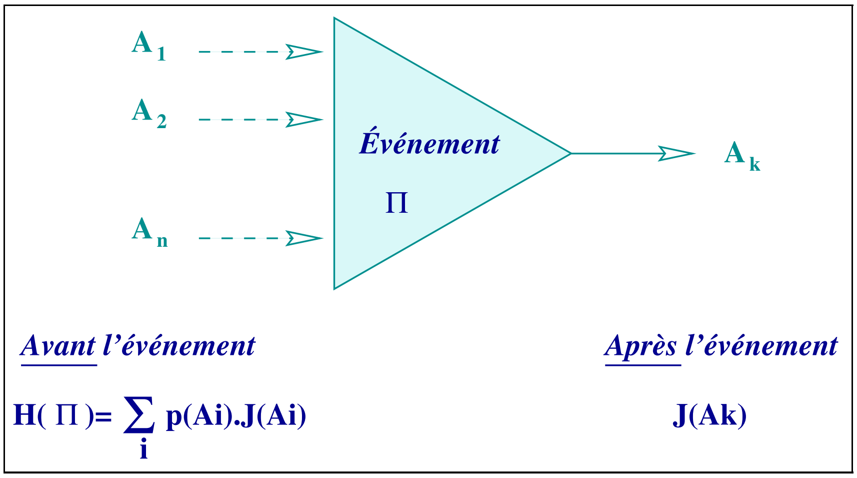


Figure 6.2 – *L'occurrence d'un événement et la mesure de son information*

d'analyse de ces mesures peut apporter une vision différente pour considérer les mesures d'information. Ainsi, au lieu de raisonner sur les bonnes propriétés que doit posséder, *a priori*, une bonne mesure de discrimination, nous préférons considérer le problème *a posteriori*, en partant de chaque ensemble d'exemples induit par les modalités d'un attribut, et en se donnant les bonnes propriétés que doit posséder une mesure qui doit rendre compte de l'adéquation de chaque ensemble vis-à-vis de la classe.

Ainsi, le raisonnement pour trouver une bonne mesure ne s'effectue pas *globalement* sur les occurrences potentielles de chaque événement [7], mais à chaque étape de la combinaison des différentes mesures obtenues niveau par niveau.

Le premier niveau de notre étude est le niveau de la mesure de l'information apportée par un événement, ou, dans le cas des arbres de décision, de la mesure du pouvoir discriminant d'une valeur d'attribut relativement à une valeur de la classe. Ensuite, les niveaux supérieurs sont des niveaux de combinaison des mesures calculées par les niveaux inférieurs. Ce sont ces niveaux qui sont présentés ici sous l'aspect d'une architecture de fonctions.

De plus, de par sa forme hiérarchique, cette architecture de fonctions permet de construire aisément des mesures de discrimination.

Application de cette vision à l'apprentissage inductif

L'apprentissage inductif veut généraliser à partir d'un ensemble, une *base d'apprentissage*, $\mathcal{E} = \{e_1, ..., e_n\}$ d'objets caractéristiques d'un concept, une loi générale régissant l'occurrence des formes du concept. Ces objets e_i représentent des *exemples* du concept C (la *classe*) à apprendre. Chaque exemple e_i de l'ensemble $\mathcal{E}$ est donné sous la forme d'une *description* associée à une valeur pour la classe C. Une description est un N-uplet de couples attribut - valeur $(A_j, e_i(A_j))$ où un attribut est une caractéristique mesurable et observable du concept. La valeur $e_i(C)$ de la classe C est la valeur du concept associé

7. Ici, le terme *événement* est synonyme de *réponse à une question sur la valeur d'un attribut*.

à la description en question. Par abus de langage, on appellera *classe de l'exemple* e_i la valeur $e_i(C)$.

Pour simplifier les notations dans cette partie, on note aussi A_j l'ensemble de toutes les modalités prises par les objets de $\mathcal{E}$ pour l'attribut A_j et on définit l'ensemble de tous les attributs existants pour décrire les objets de $\mathcal{E}$: $\mathcal{A} = \cup_j A_j$. On dénote plus particulièrement un attribut particulier $C \in \mathcal{A}$. C est l'ensemble de toutes les modalités c_k possibles prises par les objets de $\mathcal{E}$ pour la classe.

Pour tout objet $e \in \mathcal{E}$, on note e^j sa modalité pour l'attribut A_j et e^c sa modalité pour la classe C. Finalement, on définit $\forall v_{jl} \in A_j$, $\mathcal{E}_{v_{jl}} = \{e \in \mathcal{E} \mid e^j = v_{jl}\}$ ainsi que $\forall c_k \in C$, on note $\mathcal{E}_{c_k} = \{e \in \mathcal{E} \mid e^c = c_k\}$.

6.3.1 Modèle hiérarchique de fonctions

Étant donné un ensemble d'objets $\mathcal{S}$, on note $\mathbb{P}[\mathcal{S}]$ l'ensemble des sous-ensembles de $\mathcal{S}$, soit $\mathbb{P}[\mathcal{S}] = \{U \mid U \subseteq \mathcal{S}\}$. On définit trois types de fonctions sur $\mathbb{P}[\mathcal{S}] \times \mathbb{P}[\mathcal{S}]$ à valeurs dans $\mathbb{R}$.

Définition 6 ($\mathcal{F}$-fonction) *Une* $\mathcal{F}$-fonction *est une fonction* F *de* $\mathbb{P}[\mathcal{S}] \times \mathbb{P}[\mathcal{S}]$ *à valeurs dans* $\mathbb{R}$ *(c'est-à-dire telle que* $F : \mathbb{P}[\mathcal{S}] \times \mathbb{P}[\mathcal{S}] \longrightarrow \mathbb{R}$*), qui associe à tout couple* (U, V) *de* $\mathbb{P}[\mathcal{S}] \times \mathbb{P}[\mathcal{S}]$ *une valeur réelle, telle que :*

i) $F(U, V)$ **minimum** *quand* $U \subseteq V$.

ii) $F(U, V)$ **maximum** *quand* $U \cap V = \emptyset$.

iii) $F(U, V)$ *est* **strictement décroissante** *avec* $U \cap V$.
c'est-à-dire que si $U_1 \cap V \subset U_2 \cap V$ *alors* $F(U_1, V) > F(U_2, V)$ *et si* $U \cap V_1 \subset U \cap V_2$ *alors* $F(U, V_1) > F(U, V_2)$.

Soit une $\mathcal{F}$-fonction F ainsi qu'une suite de fonctions réelles continues $g_k : \mathbb{R}^k \longrightarrow \mathbb{R}^+$, $k \in \mathbb{N}$.

Définition 7 ($\mathcal{G}$-fonction) *Une* $\mathcal{G}$-fonction *est une fonction* G *de* $\mathbb{P}[\mathcal{S}] \times \mathbb{P}[\mathcal{S}]$ *à valeurs dans* $\mathbb{R}^+$*, qui, étant donné une partition* $V_1, ..., V_n$ *de* V*, associe à tout couple* (U, V) *de* $\mathbb{P}[\mathcal{S}] \times \mathbb{P}[\mathcal{S}]$ *une valeur réelle* $G(U, V)$ *définie par :*

$$G(U, V) = g_n(F(U, V_1), ..., F(U, V_n))$$

telle que :

i) $G(U, V)$ **minimum** *quand il existe* V_j $(1 \leq j \leq n)$ *tel que* $U \subseteq V_j$.

ii) $G(U, V)$ **maximum** *quand* $F(U, V_1) = ... = F(U, V_n)$.

Soit une $\mathcal{G}$-fonction G et une suite de fonctions réelles continues $h_k : \mathbb{R}^{+k} \longrightarrow \mathbb{R}^+$, $k \in \mathbb{N}$.

Définition 8 ($\mathcal{H}$-fonction) *Une* $\mathcal{H}$-fonction *est une fonction H de $\mathbb{P}[\mathcal{S}] \times \mathbb{P}[\mathcal{S}]$ à valeurs dans $\mathbb{R}^+$, qui associe à tout couple (U,V) de $\mathbb{P}[\mathcal{S}] \times \mathbb{P}[\mathcal{S}]$ une valeur réelle $H(U,V)$ qui, étant donnée une partition $U_1, ..., U_m$ de U, est définie par :*

$$H(U,V) = h_m(G(U_1,V), ..., G(U_m,V)).$$

6.3.2 Interprétation de ce modèle dans le cadre de l'apprentissage inductif

Dans le cadre de l'apprentissage inductif, l'ensemble d'objets $\mathcal{S}$ est la base des exemples $\mathcal{E}$, et les trois types de fonctions s'appliquent donc à des sous-ensembles d'exemples. Les fonctions ainsi définies permettent de mesurer différents niveaux d'adéquation entre différentes composantes de $\mathcal{E}$.

$\mathcal{F}$-fonction : inadéquation d'une modalité v_{jl} avec une modalité c_k de la classe

Le premier niveau de notre hiérarchie concerne les fonctions utiles pour mesurer l'inadéquation entre deux ensembles.

Une $\mathcal{F}$-fonction compare deux ensembles et retourne une valeur qui montre à quel point ces deux ensembles ne sont pas similaires. Deux sous-ensembles U et V de $\mathcal{E}$ sont d'autant moins similaires qu'ils ont peu d'éléments en communs.

Une distinction entre les deux ensembles est faite dans la définition des $\mathcal{F}$-fonctions, un des deux paramètres, le paramètre V, est considéré comme la référence à laquelle l'autre paramètre, U, est comparé. Une $\mathcal{F}$-fonction n'est donc pas obligatoirement symétrique. Dans la hiérarchie des mesures de comparaison de (Bouchon-Meunier et al., 1996; Rifqi, 1996), la famille des $\mathcal{F}$-fonctions s'apparente plutôt à la famille des mesures de satisfiabilité[8], mais de façon contraire : une $\mathcal{F}$-fonction mesure le degré avec lequel l'ensemble U *n'est pas inclus* dans l'ensemble V.

Avec une telle définition de fonction, dans le cadre de l'apprentissage inductif en général, la première propriété de ce type de fonctions est que, avec la $\mathcal{F}$-fonction F, $F(\mathcal{E}_{v_{jl}}, \mathcal{E}_{c_k})$ mesure l'inadéquation de la valeur v_{jl} avec la classe c_k.

En effet, $F(\mathcal{E}_{v_{jl}}, \mathcal{E}_{c_k})$ mesure le degré avec lequel $\mathcal{E}_{v_{jl}}$ n'est pas similaire à $\mathcal{E}_{c_k}$, en termes d'éléments communs. Si $F(\mathcal{E}_{v_{jl}}, \mathcal{E}_{c_k})$ est minimum, alors $\mathcal{E}_{v_{jl}}$ est inclus ou égal à $\mathcal{E}_{c_k}$ et tous les exemples appartenant à $\mathcal{E}_{v_{jl}}$ appartiennent aussi à $\mathcal{E}_{c_k}$. Par conséquent, d'après les définitions de $\mathcal{E}_{v_{jl}}$ et de $\mathcal{E}_{c_k}$, si $F(\mathcal{E}_{v_{jl}}, \mathcal{E}_{c_k})$ est minimum, alors tous les exemples de $\mathcal{E}$ qui possèdent la valeur v_{jl} pour l'attribut A_j, possèdent aussi la valeur c_k pour la classe C. Ce qui revient à dire que :

$$\mathcal{E}_{v_{jl}} \subseteq \mathcal{E}_{c_k} \quad \Longrightarrow \; (\forall e \in \mathcal{E},\; e \in \mathcal{E}_{v_{jl}} \;\Longrightarrow\; e \in \mathcal{E}_{c_k})$$

8. "Pour ce type de mesures, l'appellation usuelle de *mesures d'inclusion* serait alors peut être plus judicieuse." (Maria Rifqi)

soit donc :

$$\mathcal{E}_{v_{jl}} \subseteq \mathcal{E}_{c_k} \quad \Longrightarrow \ (\forall e \in \mathcal{E},\ e^j = v_{jl} \implies e^c = c_k)$$

De la même façon, si $F(\mathcal{E}_{v_{jl}}, \mathcal{E}_{c_k})$ est maximum, alors aucun exemple de $\mathcal{E}$ qui possède la valeur v_{jl} pour l'attribut A_j, ne possède aussi la valeur c_k pour la classe C.

$\mathcal{G}$-fonction : pouvoir discriminant d'une modalité v_{jl} relativement à la classe

Le deuxième niveau de la hiérarchie concerne des fonctions construites à l'aide d'une $\mathcal{F}$-fonction. Ainsi, à partir d'un certain nombre de valeurs retournées par cette $\mathcal{F}$-fonction, une $\mathcal{G}$-fonction permet de combiner ces valeurs pour en déduire une valeur unique. Cette agrégation doit respecter certaines contraintes qui constituent la définition des $\mathcal{G}$-fonctions.

Étant donné deux ensembles U et V ainsi qu'une partition de V en n sous-ensembles, une $\mathcal{G}$-fonction agrège les valeurs obtenues en mesurant $F(U, V_l)$, pour tout sous-ensemble V_l de la partition. Cela revient donc à mesurer l'adéquation de U avec chacun des éléments V_l et d'agréger ensuite ces degrés d'adéquation.

Dans le domaine de l'apprentissage inductif, il apparaît que $G(\mathcal{E}_{v_{jl}}, \mathcal{E}^C)$, pour la partition de $\mathcal{E}$ engendrée par les modalités c_k de la classe C (on note ici $\mathcal{E}^C$ cette association de l'ensemble $\mathcal{E}$ et de la partition par les modalités de C), mesure le pouvoir discriminant de la valeur v_{jl} pour l'ensemble des classes C. En effet, les K modalités c_k de la classe C engendrent la partition de la base d'apprentissage suivante :

$$\mathcal{E} = \bigcup_{k=1,\ldots,K} \mathcal{C}_k \quad \text{avec } \ \mathcal{C}_k = \{e \in \mathcal{E} \mid e^c = c_k\}$$

Chaque valeur de la $\mathcal{F}$-fonction, $F(\mathcal{E}_{v_{jl}}, \mathcal{E}_{c_k})$ mesure l'inadéquation de la valeur v_{jl} à la classe c_k, la $\mathcal{G}$-fonction G qui agrège ces degrés d'adéquation selon les deux propriétés de minimalité et de maximalité requises, retournera l'adéquation de la valeur à la partition de $\mathcal{E}$ engendrée par la classe C. S'il existe une valeur k telle que $\mathcal{E}_{v_{jl}}$ est inclus dans $\mathcal{E}_{c_k}$, $F(\mathcal{E}_{v_{jl}}, \mathcal{E}_{c_k})$ est minimum pour ce cas là, cela signifie que la valeur v_{jl} coïncide exactement avec la valeur de la classe c_k. Dans ce cas là, cette valeur d'attribut est totalement en adéquation avec la classe et elle discrimine totalement cette classe.

D'un autre côté, on peut remarquer aussi que $G(\mathcal{E}, \mathcal{E}^C)$, pour la partition engendrée par les c_k, mesure le degré d'impureté de l'ensemble d'objets $\mathcal{E}$ relativement à la classe C.

On peut remarquer qu'une $\mathcal{G}$-fonction est généralement continue, de plus, elle peut être symétrique (si les fonctions g_i sont symétriques) ou bien, il lui est aussi possible de posséder la propriété d'expansiabilité $(g_n(x_1, ..., x_n) = g_{n+1}(x_1, ..., x_n, \emptyset))$. Il est alors clair, qu'avec ces propriétés, une $\mathcal{G}$-fonction est un modèle entropique tel qu'il a été défini dans la partie précédente.

$\mathcal{H}$-fonction : pouvoir discriminant d'un attribut A_j relativement à la classe

Le dernier niveau de cette hiérarchie de fonctions concerne des fonctions construites à l'aide d'une $\mathcal{G}$-fonction. De la même manière qu'une $\mathcal{G}$-fonction, une $\mathcal{H}$-fonction permet de combiner des valeurs issues d'une $\mathcal{G}$-fonction, pour en déduire une valeur unique. Mais, à la différence des $\mathcal{G}$-fonctions, aucune contrainte n'est fixée pour cette agrégation.

Cette dernière définition est très générale, elle ne mérite pas de contraintes supplémentaires sur les valeurs aux limites ou des propriétés spécifiques. Les conditions aux limites ne sont pas utiles car, par exemple dans le cadre de la construction d'arbres de décision, elles seront fixées dans l'algorithme de construction. Tel algorithme choisira l'attribut qui *minimise* une $\mathcal{H}$-fonction croissante, tel autre choisira l'attribut qui *maximise* une $\mathcal{H}$-fonction décroissante. La symétrie n'est pas imposée non plus, afin d'autoriser des pondérations différentes associées aux valeurs des $\mathcal{G}$-fonctions, selon l'importance qu'il est souhaitable de lui donner.

Dans le cadre de l'apprentissage inductif, $H(\mathcal{E}, \mathcal{E}^C)$, associé aux partitions engendrées par A_j et par C, mesure le pouvoir discriminant de l'attribut A_j pour l'ensemble des classes C.

6.3.3 Application du modèle hiérarchique aux mesures de discrimination

La hiérarchie de fonctions proposée est construite à partir des fonctions de base que sont les $\mathcal{F}$-fonctions (*cf.* Figure 6.3).

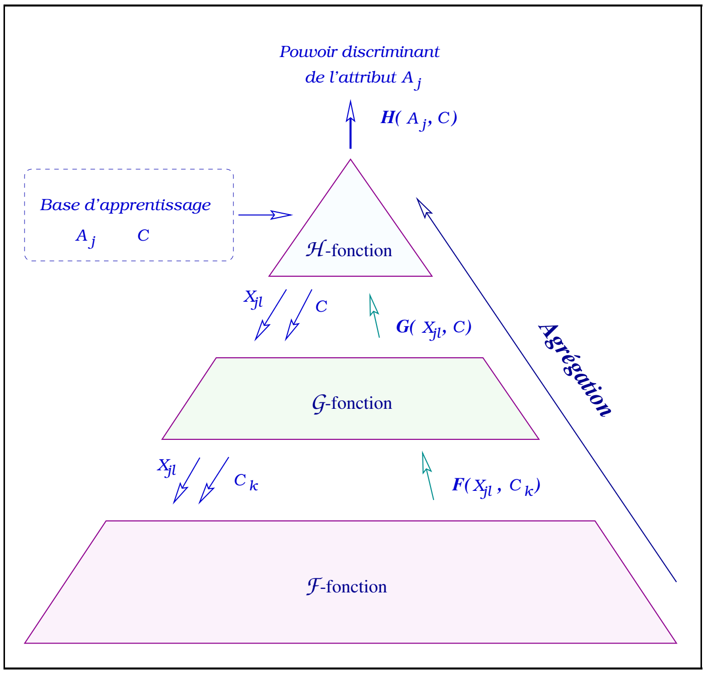


Figure 6.3 – *Hiérarchie des fonctions*

Une $\mathcal{F}$-fonction permet de connaître l'adéquation de l'ensemble des exemples induits par une modalité v_{jl} d'un attribut A_j et l'ensemble des exemples induit par la modalité d'une classe c_k. En combinant les valeurs des $\mathcal{F}$-fonctions obtenues pour chaque modalité c_k de la classe C, pour une modalité v_{jl} donnée, une $\mathcal{G}$-fonction permet de connaître le pouvoir discriminant de cette modalité relativement à l'ensemble des modalités de la classe C. Finalement, les valeurs des $\mathcal{G}$-fonctions obtenues pour chaque modalité v_{jl} de l'attribut A_j sont agrégées à l'aide d'une $\mathcal{H}$-fonction pour obtenir une mesure du pouvoir discriminant de l'attribut A_j relativement à la classe C.

Ainsi, avec les contraintes fixées pour la construction des différents type de fonctions présentées, une $\mathcal{H}$-fonction est une bonne mesure de discrimination utilisable pour construire un arbre de décision. Pour trouver un attribut afin de partitionner la base d'apprentissage, il faut comparer chaque attribut et choisir celui qui possède le meilleur pouvoir discriminant relativement à la classe. De par sa définition, une $\mathcal{H}$-fonction est adaptée à la mesure d'un tel pouvoir discriminant. Par conséquent, une $\mathcal{H}$-fonction peut être utilisée comme mesure de discrimination pour construire un arbre de décision.

Comme il a été précisé, une $\mathcal{G}$-fonction est un cas général de modèle entropique. Un point important relevé à partir de cette hiérarchie de fonctions est donc la confirmation qu'elle apporte au fait qu'une mesure d'information est une bonne mesure de discrimination pour construire un arbre de décision. Une opinion contraire aurait eu de quoi soulever, naturellement, le scepticisme de tout adepte de la théorie de l'information. Il est clair depuis très longtemps qu'un arbre de décision, ou un questionnaire, est la source d'un message et que l'information apportée par ce message se mesure par l'entropie de Shannon.

La différence apportée par la hiérarchie de fonctions proposée réside dans la forme d'agrégation finale, la $\mathcal{H}$-fonction. Cette fonction est généralement considérée comme un conditionnement de la mesure d'information utilisée. Or, il apparaît que toute fonction de combinaison de valeurs peut être utilisée. Rien n'oblige à pondérer les mesures résultantes d'une $\mathcal{G}$-fonction par une probabilité comme dans le cas de l'agrégation usuelle des mesures d'information.

Ainsi, par exemple, la symétrie d'une telle forme de combinaison, n'est envisageable qu'en l'absence de connaissances particulières sur chaque valeur d'un attribut. Il est possible de pondérer ces valeurs par d'autres poids que leur probabilité d'occurrence dans l'ensemble d'apprentissage, comme, par exemple, le coût d'obtention d'une valeur, ou l'importance de l'obtention d'une telle valeur dans le résultat de la décision.

6.4 Application de ce modèle à des mesures existantes

Dans cette partie, la hiérarchie proposée est appliquée aux deux mesures de discrimination les plus utilisées pour construire des arbres de décision : l'entropie de Shannon (équation 2.3) et la mesure basée sur le test de Gini (équation 2.5). L'intérêt de cette étude est de montrer que ces mesures rentrent bien dans le cadre de la hiérarchie des fonctions $\mathcal{F}$, $\mathcal{G}$ et $\mathcal{H}$. Appliquer cette hiérarchie à une mesure revient, en fait, à vérifier que la mesure en question est une $\mathcal{H}$-fonction. Et pour vérifier qu'une mesure est une $\mathcal{H}$-fonction, il faut la décomposer en $\mathcal{G}$-fonction et en $\mathcal{F}$-fonction.

Les mesures présentées ici se décomposent assez trivialement en $\mathcal{F}$, $\mathcal{G}$ et $\mathcal{H}$-fonctions, et elles entrent donc bien dans le cadre de cette hiérarchie.

Il faut remarquer que la $\mathcal{F}$-fonction de chacune ces deux mesures est basée sur une considération particulière qui est faite implicitement dans leur calcul en apprentissage inductif. À savoir, en règle générale, les probabilités qui entrent dans le calcul des mesures de discrimination dans le domaine de l'apprentissage inductif, sont des probabilités fréquentielles, estimées à partir de la base d'apprentissage. Les mesures de probabilités s'expriment donc toutes en fonction d'une mesure de cardinalité. C'est-à-dire que si $\mathcal{U}$ est un ensemble d'événements, la probabilité de tout sous-ensemble de $\mathcal{U}$ se calcule à partir de sa fréquence : $\forall U \subseteq \mathcal{U},\ p(U) = \frac{|U|}{|\mathcal{U}|}$

6.4.1 Application à la mesure de discrimination basée sur l'entropie de Shannon

L'entropie de Shannon a déjà été donnée dans le chapitre 2 (équation 2.3) :

$$H_S(C|A_j) \quad = \quad -\sum_{l=1}^{m_j} P(v_{jl}) \sum_{k=1}^{K} P(c_k|v_{jl}) \log(P(c_k|v_{jl}))$$

$\mathcal{F}$-fonction associée

La $\mathcal{F}$-fonction associée à la mesure d'entropie de Shannon est la fonction $F : \mathbb{P}[\mathcal{E}] \times \mathbb{P}[\mathcal{E}] \longrightarrow \mathbb{R}^+$ définie par $F(U,V) = -\log\ p(V|U)$.

Comme il a été précisé en introduction de cette section, dans le cadre de l'utilisation habituelle de l'entropie d'un ensemble, la probabilité se calcule à l'aide du nombre d'exemples : $p(V|U) = \frac{|U \cap V|}{|U|}$.

Avec cela, on vérifie que la fonction F possède les propriétés requises pour être une $\mathcal{F}$-fonction :

- $F(U,V) = 0$ quand $U \cap V = U$,
- $F(U,V) = +\infty$ quand $U \cap V = \emptyset$,
- et F est continue, décroissante en $|U \cap V|$.

La fonction F est donc bien une $\mathcal{F}$-fonction.

$\mathcal{G}$-fonction associée

À partir de la fonction F définie précédemment et d'une partition de V en n sous-ensembles V_k, on définit la fonction $G : \mathbb{P}[\mathcal{E}] \times \mathbb{P}[\mathcal{E}] \longrightarrow \mathbb{R}^+$ par :

$$G(U,V) = \sum_{k=1}^{n} F(U,V_k)\, e^{-F(U,V_k)}$$

On vérifie que la fonction G ainsi définie est bien une $\mathcal{G}$-fonction :

- $G(U,V)$ est minimum quand il existe V_j tel que $U \subseteq V_j$. En effet, dans ce cas $F(U,V_j) = 0$ et $\forall k \neq j,\ F(U,V_k) = +\infty$, car F est une $\mathcal{F}$-fonction. De plus, on sait que $\lim\limits_{x \longrightarrow +\infty} xe^{-x} = 0$, on en déduit donc que $G(U,V) = 0$ quand il existe V_j tel que $U \subseteq V_j$. La fonction F est à valeurs positives ou nulles et $G(U,V)$ la somme de valeurs positives ou nulles, donc 0 est bien un minimum pour G.

- $G(U,V)$ est maximum quand $F(U,V_1) = ... = F(U,V_n)$. En effet, dans ce cas $G(U,V) = nF(U,V_1)\ e^{-F(U,V_1)} = n(-\log\frac{1}{n})(\frac{1}{n}) = \log n$. La maximalité de G est alors démontrée en utilisant le fait que $G(U,V) = H_S(P(V_1|U),...,P(V_n|U))$, et que, dans ce cas, comme l'ont démontré (Aczél et Daróczy, 1975) (théorème (1.3.7) page 34 de leur livre), $\forall (p_1,...,p_n)$ on a :

$$-\sum_{k=1}^{n} p_k \log p_k \ \leq\ \log n$$

La fonction G est donc bien une $\mathcal{G}$-fonction.

$\mathcal{H}$-fonction associée

À partir de la fonction G définie précédemment et d'une partition de U en m sous-ensembles, on définit la fonction $H : \mathbb{P}[\mathcal{E}] \times \mathbb{P}[\mathcal{E}] \longrightarrow \mathbb{R}^+$ par

$$H(U,V) = \sum_{j=1}^{m} p(U_j) G(U_j, V)$$

La fonction H ainsi définie coïncide avec la mesure d'entropie de Shannon. On a donc montré, en trouvant les trois formes de l'entropie de Shannon, que cette mesure rentre bien dans le cadre de notre hiérarchie de fonctions $\mathcal{FGH}$.

6.4.2 Application à la mesure d'impureté de Gini

La mesure d'impureté, basée sur le test de Gini, qui est utilisée pour construire des arbres de décision par la méthode CART de (Breiman et al., 1984) a déjà été donnée dans le chapitre 2 (équation 2.5) :

$$H_G(C|A_j) = \sum_{l=1}^{m_j} P(v_{jl}) \sum_{k_1 \neq k_2} P(c_{k_1}|v_{jl}) P(c_{k_2}|v_{jl})$$

Comme pour l'entropie de Shannon, cette mesure se décompose assez trivialement en fonctions $\mathcal{F}$, $\mathcal{G}$ et $\mathcal{H}$.

$\mathcal{F}$-fonction associée

La $\mathcal{F}$-fonction associée à la mesure de Gini est la fonction $F : \mathbb{P}[\mathcal{E}] \times \mathbb{P}[\mathcal{E}] \longrightarrow \mathbb{R}$ définie par $F(U,V) = -p(V|U)$.

Comme dans le cas de l'entropie, $p(V|U)$ est approximée par $\frac{|U \cap V|}{|U|}$. Dans ce cas, on a $F(U,V) = -\frac{|U \cap V|}{|U|}$ et les conditions aux limites et de décroissance requises pour être une fonctions $\mathcal{F}$ se vérifient aisément.

$\mathcal{G}$-fonction associée

À partir de la fonction F définie précédemment et d'une partition de V en n sous-ensembles V_k, on définit la fonction $G : \mathbb{P}[\mathcal{E}] \times \mathbb{P}[\mathcal{E}] \longrightarrow \mathbb{R}^+$ par :

$$G(U,V) = \sum_{i \neq j} F(U,V_i).F(U,V_j)$$

Avec cette définition, G est équivalente à la mesure de diversité de Gini donnée par (Breiman et al., 1984) :

$$G(U,V) = \sum_{i \neq j} P(V_i|U)P(V_j|U)$$

On vérifie alors que la fonction G ainsi définie est bien une $\mathcal{G}$-fonction. Les propriétés de minimalité et de maximalité découlent naturellement de la définition d'une mesure d'impureté donnée par Breiman que vérifie le test de Gini.

$\mathcal{H}$-fonction associée

À partir de la fonction G définie précédemment et d'une partition de U en m sous-ensembles U_j, on définit la fonction $H : \mathbb{P}[\mathcal{E}] \times \mathbb{P}[\mathcal{E}] \longrightarrow \mathbb{R}^+$ par

$$H(U,V) = \sum_{j=1}^{m} p(U_j)G(U_j,V)$$

La fonction H ainsi définie coïncide avec la mesure de discrimination basée sur le test de Gini proposé par (Breiman et al., 1984). On a donc montré, en trouvant les trois formes de cette mesure, qu'elle rentre bien, elle aussi, dans le cadre de notre hiérarchie de fonctions $\mathcal{FGH}$.

6.4.3 Extension à des mesures existantes

L'étude qui est faite ici pour la mesure d'entropie de Shannon et le test de Gini se doit d'être étendue aux autres mesures existantes pour vérifier leur concordance à cette hiérarchie.

En fait, on peut déjà remarquer ce qui avait déjà été noté à propos des $\mathcal{G}$-fonctions. Les $\mathcal{G}$-fonctions sont des extensions des modèles entropiques. Tous les modèles entropiques existants sont des $\mathcal{G}$-fonctions qui possèdent la propriété de symétrie en supplément des propriétés requises. Par conséquent, toutes les mesures de discrimination classiquement utilisées en apprentissage possèdent une forme de $\mathcal{G}$-fonction, et donc, par suite, une forme de $\mathcal{H}$-fonction. L'aspect supplémentaire de la hiérarchie proposée est la forme de $\mathcal{F}$-fonction qui compose les $\mathcal{G}$-fonctions.

Les $\mathcal{F}$-fonctions sont d'un niveau encore plus "bas" que les $\mathcal{G}$-fonctions, et elles rappellent que toute mesure de discrimination se doit d'être fondée sur une mesure de dissimilarité des sous-ensembles induits par les modalités d'un attribut envers les sous-ensembles induits par les modalités de la classe.

6.5 Construction d'une mesure à l'aide du modèle hiérarchique

L'intérêt principal de la hiérarchie de fonctions $\mathcal{FGH}$ proposée réside dans la construction de nouvelles mesures de discrimination.

L'étude de la hiérarchie montre bien que l'étape fondamentale de la construction d'une mesure de discrimination est la détermination de la $\mathcal{F}$-fonction qui sert de base aux autres fonctions. Les deux autres fonctions, les fonctions $\mathcal{G}$ et $\mathcal{H}$, sont des fonctions d'agrégation. La $\mathcal{G}$-fonction doit respecter un certain nombre de propriétés, la $\mathcal{H}$-fonction peut, elle, être choisie sous une forme générique, comme par exemple, celle prise dans le cas de la mesure d'entropie de Shannon ou de la mesure basée sur le test de Gini.

Conformément à ce qui a été développé dans le chapitre 5 sur les forêts d'arbres de décision flous, nous nous intéressons maintenant au cas où il n'existe que deux valeurs pour la classe à reconnaître. Comme dans le cas de la règle du *twoing* de (Breiman et al., 1984), tout problème à plus de deux classes sera transformé en problème à deux classes.

Construction d'une $\mathcal{F}$-fonction

La première fonction à choisir est la $\mathcal{F}$-fonction. Une forme simple, simplement décroissante en $U \cap V$, peut être prise comme $\mathcal{F}$-fonction. Par exemple :

$$F(U, V) = \frac{1}{|U \cap V|}$$

Cette mesure d'information [9] proposée par Kampé de Fériet dans ces travaux (Kampé de Fériet et Forte, 1967).

---

9. Une mesure d'information au sens de Kampé de Fériet est une mesure J, définie sur un ensemble d'éléments Ω, à valeurs dans $\mathbb{R}^+$ telle que :

i $J(\Omega) = 0$ et $J(\emptyset) = +\infty$

ii $\forall B, A \in \Omega,\ B \subset A \Longrightarrow J(B) \geq J(A)$

Cette définition d'une mesure d'information est donnée pour la première fois dans l'article (Kampé de Fériet et Forte, 1967).

Il est intéressant de remarquer au passage la dualité entre une mesure d'information définie ainsi par Kampé de Fériet et la définition d'une mesure de Sugeno (Sugeno, 1977). Une mesure S, définie sur un ensemble d'éléments Ω, à valeurs dans $\mathbb{R}^+$ est une mesure de Sugeno si :

i $S(\Omega) = 1$ et $S(\emptyset) = 0$

ii $\forall B, A \in \Omega,\ B \subset A \Longrightarrow S(B) \leq S(A)$

Comme (Shafer, 1987) le démontre dans son approche hiérarchique, les probabilités sont des mesures de Sugeno particulières.

À partir de là, la cohérence entre les deux définitions de mesures est conservée : Une mesure d'information est définissable à partir d'une mesure de Sugeno par : $\forall A \in \Omega,\ J(A) = f(S(A))$ avec f une fonction décroissante de $\mathbb{R}^+$ dans $\mathbb{R}^+$. Par exemple, on peut choisir $f(x) = -\log(x)$.

Construction d'une $\mathcal{G}$-fonction associée

À partir de la fonction F définie précédemment et d'une partition de V en n sous-ensembles V_k, nous présentons[10] une fonction $G : \mathbb{P}[\mathcal{E}] \times \mathbb{P}[\mathcal{E}] \longrightarrow \mathbb{R}^+$ possible :

$$G(U,V) = \frac{1}{\sum_{i \neq j}(F(U,V_i) - F(U,V_j))^2}$$

On vérifie que la fonction G ainsi définie est bien une $\mathcal{G}$-fonction :

- $G(U,V)$ est minimum quand il existe V_k tel que $U \subseteq V_k$.
  Dans ce cas $F(U,V_k) = \frac{1}{|U|}$, et $\forall i \neq j$, $(F(U,V_i) - F(U,V_j))^2 = +\infty$. Par conséquent, $G(U,V) = 0$ et $G(U,V)$ est donc bien minimum (On vérifie aisément que $G(U,V)$ est toujours positif).

- $G(U,V)$ est maximum quand $F(U,V_1) = ... = F(U,V_n)$.
  En effet, dans ce cas $\forall i \neq j$ $F(U,V_i) - F(U,V_j))^2 = 0$, et par conséquent on a alors que $G(U,V) = +\infty$, et $G(U,V)$ est donc maximum.

Construction d'une $\mathcal{H}$-fonction associée

À partir de la fonction G définie précédemment et d'une partition de U en m sous-ensembles U_j, la fonction $H : \mathbb{P}[\mathcal{E}] \times \mathbb{P}[\mathcal{E}] \longrightarrow \mathbb{R}^+$ est définie, comme dans le cas de la mesure de Shannon ou le test de Gini, par :

$$H(U,V) = \sum_{j=1}^{m} p(U_j)G(U_j,V)$$

La fonction $\mathcal{H}$-fonction ainsi construite peut servir de mesure de discrimination dans un processus de construction d'arbres de décision.

6.6 Mesures de discrimination en présence de données numériques-symboliques

La hiérarchie de fonctions $\mathcal{FGH}$ proposée s'applique dans le cas de mesures de discrimination devant prendre en compte des données numériques-symboliques. Les fonctions à adapter dans le cas numérique-symbolique, sont les fonctions devant prendre en compte cet aspect différent. Dans notre hiérarchie, cette prise en compte s'effectue à deux niveaux.

10. L'information hyperbolique, qui est une formule de combinaison d'informations citée par Kampé de Fériet (Kampé de Fériet, 1971; Kampé de Fériet, 1975), peut aussi être utilisée comme $\mathcal{G}$-fonction :

$$G(U,V) = \frac{1}{F(U,V_1) + F(U,V_2)}$$

Au niveau des $\mathcal{F}$-fonctions, cette prise en compte est fondamentale. Les $\mathcal{F}$-fonctions sont en liaison directe avec les données et elles se doivent donc de dépendre toutes leurs composantes. Il faut donc étendre les $\mathcal{F}$-fonctions classiques pour la prise en compte de données numériques-symboliques. Par exemple, la cardinalité des deux mesures construites peut être considérée comme une cardinalité d'ensembles flous, ce qui généralise alors les $\mathcal{F}$-fonctions qui l'utilisent en présence de sous-ensembles flous.

Au niveau des $\mathcal{G}$-fonctions, cette prise en compte est implicite. Elle ne se fait pas directement dans le calcul de la valeur retournée par la fonction, mais elle se fait dans la contrainte de minimalité de la fonction. Cette contrainte est définie relativement à une inclusion qui peut être considérée comme une inclusion floue.

L'autre type de fonctions, les $\mathcal{H}$-fonctions, n'a pas de contact direct avec l'aspect numérique-symbolique présent dans la base d'apprentissage et n'a donc pas à être obligatoirement adapté au cas flou. Il est quand même possible d'utiliser une composante numérique-symbolique dans son expression, comme, par exemple, une probabilité d'événements flous.

Exemple d'extension d'une $\mathcal{F}$-fonction pour la prise en compte de données numériques-symboliques

La mesure donnée dans le paragraphe 6.5 est étendue pour prendre en compte les données numériques-symboliques en utilisant la mesure de cardinalité d'ensembles flous de Zadeh (aussi nommée σ-count). Si U est un sous-ensemble flou d'un ensemble $\mathcal{X}$, de fonction d'appartenance μ définie sur un univers de valeurs X, sa cardinalité $|U|$ est définie par :

$$|U| = \int_X \mu(x)\,dx$$

si X est continu. Si l'ensemble des valeurs X est discret, on définit de même :

$$|U| = \sum_X \mu(x)\,dx$$

C'est cette mesure de cardinalité d'ensembles flous qui permet ensuite d'étendre les probabilités classiques en probabilités d'événements flous, pour la prise en compte des données numériques-symboliques.

Il existe d'autres méthodes de construction de mesure de probabilités d'événements flous qui peuvent être utilisées pour construire une $\mathcal{F}$-fonction puis une $\mathcal{G}$ et une $\mathcal{H}$-fonction. Par exemple, on peut aussi utiliser les travaux de (Bouchon, 1980) relatifs au calcul de l'information transmise par un système d'événements flous. Dans ces travaux, un nouveau concept de probabilité d'ensemble flous est proposée. Cette probabilité se calcule par niveau d'α-coupe ce qui autorise la construction d'une mesure d'information par α-coupes.

6.7 Mesures de discrimination et apprentissage par prototypes

La hiérarchie des fonctions $\mathcal{FGH}$ présentées dans la section précédente permet de se rendre compte d'un des fondements d'une mesure de discrimination. La $\mathcal{F}$-fonction qui sert de base à une mesure de discrimination, peut être rattachée au concept de mesure de comparaison. Ainsi, il est possible de rapprocher l'apprentissage inductif par arbre de décision de l'apprentissage inductif par construction de prototypes.

Dans cette partie, après un court rappel sur l'apprentissage par prototypes[11], les points communs de la phase de classification et de la phase d'apprentissage des deux méthodes d'apprentissage inductif, sont développés.

6.7.1 Rappels sur la théorie des prototypes

L'apprentissage par prototypes est une méthode d'apprentissage inductif. Elle est fondée sur la détermination des caractéristiques typiques d'un ensemble d'éléments.

Comme dans le cas de la construction des arbres de décision, on suppose donnée une base d'apprentissage $\mathcal{E}$ composée d'exemples d'un concept, décrits par un ensemble de couples $(A_k, e_i(A_k))$ et associés chacun à une modalité c_k de la classe C.

Pour (Rifqi, 1996), l'apprentissage par prototypes consiste à déterminer un ensemble de modalités d'attributs typiques pour chaque modalité c_k de la classe C. Un exemple virtuel, prototype de cette modalité pour la base d'apprentissage, peut alors être construit.

La *typicalité* de chaque valeur d'attribut relativement à chaque modalité de la classe est mesurée afin de sélectionner les modalités les plus typiques qui composeront le prototype de cette modalité de la classe.

La typicalité d'une modalité v_{jl} d'un attribut A_j, relativement à la modalité c_k de la classe, se mesure en regardant la ressemblance R de v_{jl} avec toutes les autres modalités de A_j associées aux exemples possédant la même modalité c_k pour la classe. Cette ressemblance est agrégée à la dissimilarité D de cette modalité v_{jl} avec toutes les modalités de A_j associées aux exemples possédant une modalité différente de c_k pour la classe. Un prototype est la conjonction des modalités d'attributs les plus typiques.

Un prototype d'une modalité c_k de la classe peut s'exprimer sous la forme d'une règle *si - alors*. La prémisse de cette règle est une conjonction ou une disjonction des tests sur les attributs relatifs aux modalités les plus typiques de la modalité c_k de la classe C. La conclusion est la modalité c_k elle même.

L'apprentissage par prototypes permet de construire une base de règles *si - alors*, dans cette base, il existe autant de règles que de modalités de la classe présentes dans la base d'apprentissage.

11. La vision de l'apprentissage par prototypes développée ici est celle de (Rifqi, 1996).

6.7.2 Analogies entre un arbre de décision et un groupe de prototypes

Équivalence en classification

Comme il est remarqué dans (Bouchon-Meunier et al., 1997a), un arbre de décision et un groupe de prototypes de classes sont tous deux équivalents à une base de règles.

Au chapitre 2, l'équivalence entre un arbre de décision et une base de règles a été développée. Chaque chemin de l'arbre est associé à une règle *si - alors*. La conjonction des tests rencontrés sur les nœudsdu chemin constituent la prémisse de la règle, et la modalité de la classe libellant la feuille du chemin constitue la conclusion de la règle. Par conséquent, un arbre de décision est équivalent à une base avec autant de règles que de chemins dans l'arbre. Les règles comportant la même conclusion, c'est-à-dire la même modalité c_k, peuvent être fusionnées en utilisant une disjonction sur les prémisses, pour n'obtenir qu'une unique règle possédant pour conclusion c_k.

Ainsi, un arbre de décision s'interprète lui aussi comme un système de prototypes.

Analogie méthodologique en apprentissage

En supplément de cette équivalence durant la phase de classification, l'examen de la hiérarchie des fonctions $\mathcal{FGH}$ permet de déceler une équivalence de méthodologie entre ces deux méthodes de construction de base de règles.

Comme il a été rappelé, pour mesurer la typicalité d'une modalité d'attribut, la méthode des prototypes calcule la ressemblance de cette modalité avec toutes les modalités des éléments possédant la même classe. Cette ressemblance est agrégée avec la dissimilarité de cette modalité avec toutes les modalités des éléments possédant une classe différente.

De son côté, dans la méthode de construction d'arbres de décision, le pouvoir discriminant de chaque attribut est mesuré pour choisir le meilleur attribut qui partitionne la base d'apprentissage. Or, cette mesure de discrimination est une $\mathcal{H}$-fonction qui s'appuie sur l'agrégation de $\mathcal{G}$-fonctions, elles-mêmes formées par l'agrégation de $\mathcal{F}$-fonctions. Comme il a été remarqué dans le paragraphe 6.3.2, une $\mathcal{F}$-fonction est un cas plus général d'une mesure de comparaison au sens de (Rifqi, 1996). La fonction $F(\mathcal{E}_{v_{jl}}, \mathcal{E}_{c_k})$ mesure l'inadéquation de la modalité v_{jl} avec la modalité c_k de la classe qui correspond à une mesure de non-inclusion de l'ensemble $\mathcal{E}_{v_{jl}}$ des exemples possédant la modalité v_{jl} dans l'ensemble $\mathcal{E}_{c_k}$ des exemples possédant la modalité c_k. Ainsi, on retrouve le même type de mesure que dans le cas des prototypes pour mesurer le pouvoir discriminant d'une valeur.

La valeur de ce pouvoir discriminant se ramène toujours à la mesure de la ressemblance d'une modalité d'un attribut vis-à-vis d'une modalité de la classe.

Ce type de constatation est à rapprocher des travaux de formalisation des mesures de sous-ensembles flous de (Xuecheng, 1992). Dans son point de vue, il relie l'entropie à une mesure de similarité. Une mesure d'entropie est une mesure de similarité entre un ensemble et son complémentaire.

Dans le même esprit, on peut citer l'entropie de (Kosko, 1997), construite à partir d'un degré d'inclusion, qui est une instance de mesure de similarité au sens de Tverski[12].

12. Pour en savoir plus sur les mesures de comparaison : (Rifqi, 1996).

Avec notre approche, ce formalisme peut donc s'étendre pour les mesures de discrimination. L'entropie d'un ensemble relativement à un autre ensemble, est une mesure de son inclusion dans cet ensemble.

6.8 Vers une mesure de discrimination floue?

Les mesures de discrimination usuelles sont des fonctions réelles qui retournent une valeur représentant le pouvoir de discrimination d'un attribut relativement à une classe. Or, la notion de *pouvoir discriminant* est une notion qui peut être considérée comme floue, comme peut l'être aussi la notion de probabilités, ou la notion d'information apportée par un événement.

Ce qui est le plus intéressant dans une mesure de discrimination ne réside pas dans la valeur précise qu'elle retourne pour un attribut, mais dans la comparaison du pouvoir discriminant de plusieurs attributs. C'est en accord avec l'opinion de Lotfi Zadeh concernant le "Computing with words" : "*... First, computing with words is a necessity when the available information is not precise enough to justify the use of numbers. And second, computing with words is advantageous when there is a tolerance for imprecision, uncertainty and partial truth that can be exploited to achieve tractability, robustness, low solution cost and better rapport with reality.*" (Zadeh, 1997). Ces arguments sont pleinement valables dans le cadre des mesures de discrimination.

Les probabilités classiques ont déjà été étendues en probabilités floues[13] par (Yager, 1977; Yager, 1984),(Ralescu, 1994; Ralescu, 1995). Les exemples de probabilités floues cités basent tous deux leur probabilité floue sur la définition d'une cardinalité floue, qui retourne une valeur floue comme cardinalité d'un sous-ensemble flou.

Il est donc alors envisageable d'étendre une mesure de discrimination en mesure floue et non plus seulement en mesure prenant en compte seulement les données numériques-symboliques. Dans ce cas, il faut alors adapter une série d'outils pour traiter et gérer ce type de mesures floues dans la comparaison des attributs. Pour ce faire, il faut posséder un moyen de comparer deux valeurs floues associées à des mesures de discrimination, comme, par exemple, les solutions basées sur une mesure de "crispness" (Bouchon, 1982a) ou bien en utilisant des mesures de comparaison de sous-ensembles flous.

6.9 Conclusion

Dans ce chapitre, une hiérarchie constructive de mesures de discrimination a été proposée.

Cette hiérarchie est fondée d'après les travaux de Kampé de Fériet relatifs aux mesures d'information. Elle donne lieu à une approche du même ordre, mais adaptée aux mesures de discrimination. Au lieu de raisonner sur les bonnes propriétés que doit posséder, *a priori*, une bonne mesure de discrimination, le problème est considéré *a*

13. Il faut bien différencier la notion de *probabilité d'événements flous*, qui est souvent abusivement appelée probabilité floue, et la notion de *probabilité floue*. Dans le premier cas, on associe à un événement floue une probabilité qui est une valeur réelle. Dans le second cas, on associe à un événement, flou ou non, une probabilité qui est une valeur floue.

posteriori, en partant de chaque ensemble d'exemples induit par les modalités d'un attribut, et en se donnant les bonnes propriétés que doit posséder une mesure qui doit rendre compte de l'adéquation de chaque ensemble vis-à-vis de la classe.

Ainsi, le premier niveau de cette hiérarchie, les $\mathcal{F}$-fonctions, rend compte de l'inadéquation entre une modalité d'attribut et une modalité de la classe.

Le deuxième niveau de la hiérarchie, les $\mathcal{G}$-fonctions, est composé de fonctions d'agrégation de $\mathcal{F}$-fonctions. Ces $\mathcal{G}$-fonctions sont des cas généraux des modèles entropiques définis dans la classification de (Bouchon, 1985). Des mesures d'impureté telles l'entropie de Shannon ou le test de Gini sont des $\mathcal{G}$-fonctions.

Le dernier niveau de la hiérarchie proposée, les $\mathcal{H}$-fonctions, est composée par des fonctions qui permettent d'agréger des $\mathcal{G}$-fonctions. Ces fonctions sont des mesures de discrimination qui peuvent être utilisées dans un processus de construction d'arbres de décision.

Grâce à ces types de fonctions, l'extension d'une mesure de discrimination classique pour la prise en compte de données numériques-symboliques, s'effectue de façon structurée : l'aspect flou des données se intervient à chaque niveau de la hiérarchie de fonctions $\mathcal{FGH}$.

De plus, cette structure hiérarchique de fonctions permet la construction de mesures de discrimination.

Finalement, l'étude des propriétés des $\mathcal{F}$-fonctions permet de rapprocher encore la méthode de construction d'arbres de décision d'une autre méthode d'apprentissage inductif qui est la construction de prototypes. Le rapprochement était déjà explicite dans le fait que ces deux méthodes construisent une structure de connaissance similaire : une base de règles. L'analogie méthodologique de construction des règles laissent entrevoir des perspectives de formalisation communes de ces deux méthodes d'apprentissage inductif.

Chapitre 7

Résultats en classification

La faculté de reconnaissance et de classification constitue donc l'une des bases fondamentales du comportement animal.

J. Ruffié – *Traité du vivant*

7.1 Présentation

Comme il a été précisé dans les chapitres correspondants (chapitre 2 et chapitre 5), les applications Salammbô et Tanit ont été implantées en Langage C sur station Sparc-Sun, sous le système d'exploitation UNIX.

Dans le chapitre 5, des résultats concernant les forêts d'arbres de décision flous ont été donnés. Dans ce chapitre-ci, nous ne présentons que les résultats concernant l'application Salammbô sur différents types de bases.

L'évaluation des résultats donnés est délicate et difficile à mettre en œuvre. En effet, il existe plusieurs critères pour juger et comparer des méthodes. Nous préférons donner ici les résultats de notre méthode sur diverses bases d'exemples, sans but de comparaison, afin de montrer la capacité de prise en compte de tout type de problèmes par notre système.

Pour chaque type de méthode, des résultats sont donnés avec différentes façons d'utiliser les arbres flous construits. Les deux principales méthodes de construction utilisées génèrent des arbres de décision flous.

La première méthode est décrite dans (Marsala, 1994). Elle met en œuvre une discrétisation basée sur la minimalisation du critère de discrimination (l'entropie de Shannon ou autre). Cette discrétisation permet de trouver deux valeurs successives de la base d'apprentissage qui servent à partitionner l'attribut numérique en deux modalités.

La deuxième méthode de construction est la méthode de construction d'arbres de décision flous basée sur l'utilisation de l'entropie étoile ainsi que la méthode de classification floue présentées dans le chapitre 2, avec la méthode de construction de partitions floues présentées au chapitre 4 (l'application *Salammbô*).

Pour chaque méthode, nous présentons les résultats obtenus en utilisant les arbres flous de différentes façons pour le classement d'un nouvel exemple.

La première façon d'utiliser un arbre flou est de respecter les tests des nœuds strictement (méthode dite *stricte* par la suite). À partir d'un test flou sur deux modalités floues, il est possible de déduire un test strict sur une valeur numérique en choisissant la valeur qui appartient de façon égale aux deux modalités floues (*ie.* de degré d'appartenance égal à $\frac{1}{2}$).

Une autre façon d'utiliser un arbre de flou est de mettre en œuvre la méthode d'inférence présenté au chapitre 2, et d'utiliser différentes t-normes et t-conormes pour agréger les résultats.

Dans ce chapitre, seuls les résultats sont donnés. Quelques exemples d'arbres construits pour chaque type de données sont donnés en annexe F

7.2 Approximation de fonctions

Les premiers résultats concernent un domaine artificiel sur lequel nous proposons de tester la construction d'arbres de décision flous.

Il est connu qu'une fonction $f : \mathbb{R} \longrightarrow \mathbb{R}$ partage le plan en deux parties : l'ensemble E_0 des points (x, y) tels que $y < f(x)$ et l'ensemble E_1 des points (x, y) tels que $y \geq f(x)$. Un arbre de décision peut alors être construit pour reconnaître la position d'un point quelconque par rapport à cette fonction.

Il est possible ainsi de générer, en utilisant un générateur de nombres aléatoires, un grand nombre de points (x, y) et d'associer la classe 0 aux points qui appartiennent à E_0 et la classe 1 aux points qui appartiennent à E_1. On obtient alors aisément une base d'apprentissage et une base de test pour tester un algorithme d'apprentissage.

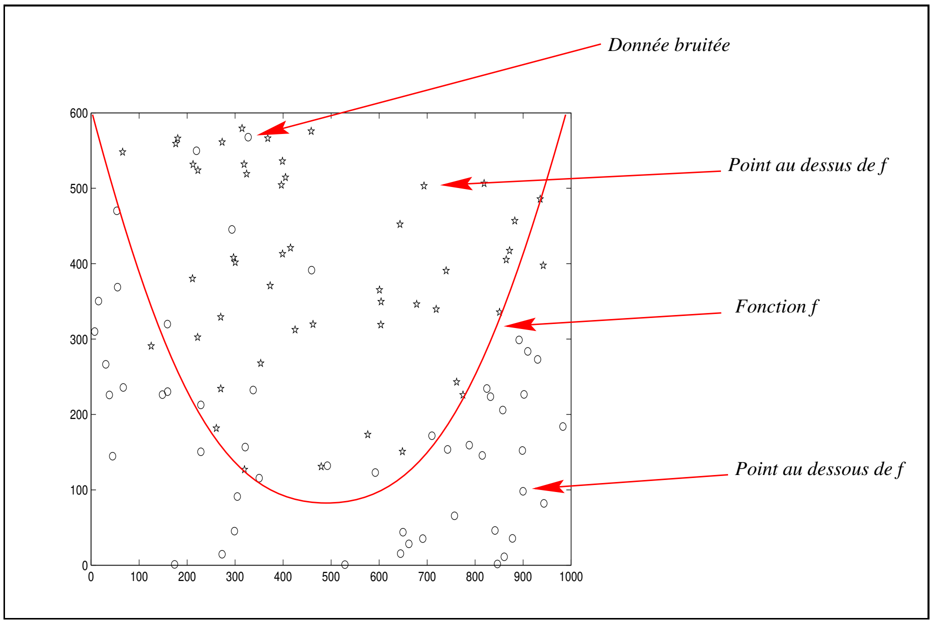


Figure 7.1 – *Exemple de base d'apprentissage bruitée pour la fonction f*

La fonction que nous avons choisi d'utiliser est un polynôme du second degré :

$$f(x) = \frac{(x-500)^2}{500} + 100$$

Pour restreindre l'espace de recherche, les valeurs de x sont choisies aléatoirement entre 0 et 1000. Par conséquent, les points de f peuvent varier entre 100 et 600. Pour chaque x, une valeur y, générée entre 0 et 600, lui est associée. Cette valeur est ensuite comparée à $f(x)$ pour déterminer si le point (x, y) est au-dessous ou au-dessus du point $(x, f(x))$. Du bruit est rajouté dans l'ensemble des points générés en modifiant la classe de certains points (Figure 7.1). Un tel bruit est intéressant à intégrer dans une telle base afin de mieux rendre compte des avantages des méthodes floues.

À partir d'un générateur de points écrit en langage C, 2000 exemples, autant d'exemples de E_0 que d'exemples de E_1, ont été générés. Des tests en validation croisée ont ensuite été réalisés. Les 2000 exemples ont été répartis en 10 bases d'apprentissage de 200 exemples, en respectant la proportion de chaque classe. Pour chaque base d'apprentissage un arbre de décision a été construit et validé sur les exemples des autres bases.

7.2.1 Arbres à seuils flous et mesure de Shannon

Les premiers tests ont été réalisés en utilisant une méthode de discrétisation classique, basée sur le choix d'un point de coupure minimisant la mesure de discrimination (l'entropie de Shannon).

Les arbres ainsi construits obtiennent les taux d'erreur en classification donnés dans le tableau 7.1. Dans ce tableau, des résultats sont donnés pour plusieurs méthodes d'utilisation des arbres de décision flous. La moyenne et l'écart type des taux d'erreur en classification obtenus pour l'ensemble des bases d'apprentissage sont donnés.

Méthode	*Taux d'erreur*	*Écart type*
Stricte	15,4%	2,1
Opérateur de Zadeh ou Probabilistes	13,1%	2,2
Opérateurs de Lukasiewicz	12,4%	2,5

Table 7.1 – *Approximation de fonction : arbres à seuils flous et mesure de Shannon*

Sans mécanisme d'élagage, les arbres construits par cette méthode comportent en moyenne 40,4 branches, ils ont une profondeur moyenne de 7,1 nœuds et une profondeur maximale de 12,1 nœuds. Le temps moyen de construction d'un arbre à partir d'une des bases d'apprentissage, comportant 200 exemples, est inférieur à une seconde, sur une station Sun UltraSparc.

7.2.2 Arbres à seuils flous et mesure de discrimination

Les deuxièmes résultats concernent la méthode de construction d'arbres de décision classique, utilisant une méthode de discrétisation générant des points de coupure flous comme précédemment. Mais, ici, la mesure d'entropie de Shannon a été remplacé par la mesure de discrimination construite dans le chapitre 6 pour classer les attributs lors de la construction de l'arbre de décision. Les arbres ainsi construits obtiennent les taux d'erreur en classification donnés dans le tableau 7.2.

Méthode	*Taux d'erreur*	*Écart type*
Stricte	15,7%	2,2
Opérateur de Zadeh ou Probabilistes	14,0%	2,1
Opérateurs de Lukasiewicz	12,1%	2,4

Table 7.2 – *Approximation de fonction : arbres à seuils flous et mesure de discrimination*

Les arbres construits par cette méthode comportent en moyenne 49,8 branches, ils ont une profondeur moyenne de 6,8 nœuds et une profondeur maximale de 11,5 nœuds. Le temps moyen de construction d'un arbre à partir d'une des bases d'apprentissage est inférieur à une seconde sur une station Sun UltraSparc.

7.2.3 Arbres construits avec l'entropie étoile

Les derniers tests portent sur la méthode de construction d'arbres de décision flous, basée sur l'utilisation de la mesure d'entropie étoile, et la méthode de construction de partitions floues présentée au chapitre 4. Les arbres de décision flous construits obtiennent les taux d'erreur en classification donnés dans le tableau 7.3.

Méthode	*Taux d'erreur*	*Écart type*
Stricte	11,6%	2,1
Opérateurs de Zadeh	12,3%	2,3
Opérateurs de Lukasiewicz	12,5%	2,6
Opérateurs probabilistes	12,2%	2,3

Table 7.3 – *Approximation de fonction : arbres construits avec l'entropie étoile*

Les arbres construits par cette méthode comportent en moyenne 11,2 branches, ils ont une profondeur moyenne de 3,7 nœuds et une profondeur maximale de 4,6 nœuds. Le temps moyen de construction d'un arbre à partir d'une des bases d'apprentissage est inférieur à une seconde sur une station Sun UltraSparc.

7.3 Données des Iris

Les résultats suivants sont donnés pour la base de reconnaissance des iris de Fischer disponible sur le site `ftp`[1] de l'université d'Irvine (Californie, USA). Dans ces données, les descriptions de chaque exemple comportent 4 attributs numériques et il y a trois classes à reconnaître. Les tests ont été encore effectués en validation croisée, dix bases d'apprentissage comportant chacune 135 exemples sont utilisées pour construire un arbre qui est ensuite testé sur une base de test comportant 15 exemples.

1. `ftp://ftp.ics.uci.edu/pub/machine-learning-databases/`

De nombreuses méthodes existantes ont été validées sur cette base. Comme méthode de construction d'arbres de décision classiques, on peut citer les résultats obtenus par la version récente de l'algorithme ID3, l'algorithme C4.5 de (Quinlan, 1996). Pour cette base, cet algorithme produit des arbres de 8,5 branches en moyenne qui obtiennent un taux d'erreur moyen de 4,8%.

7.3.1 Arbres à seuils flous et mesure de Shannon

Les arbres construits par cette méthode obtiennent les taux d'erreur en classification donnés dans le tableau 7.4.

Méthode	*Taux d'erreur*	*Écart type*
Toutes	4,7%	4,5

Table 7.4 – *Iris : arbres à seuils flous et mesure de Shannon*

Sans mécanisme d'élagage, les arbres construits par cette méthode comportent en moyenne 8,7 branches, ils ont une profondeur moyenne de 3,8 nœuds et une profondeur maximale de 5,1 nœuds. Le temps moyen de construction d'un arbre à partir d'une des bases d'apprentissage, comportant 135 exemples, est inférieur à une seconde, sur une station Sun UltraSparc.

7.3.2 Arbres à seuils flous et mesure de discrimination

Les arbres ainsi construits obtiennent les taux d'erreur en classification donnés dans le tableau 7.5.

Méthode	*Taux d'erreur*	*Écart type*
Stricte, Zadeh ou probabilistes	4,7%	5,5
Opérateurs de Lukasiewicz	5,3%	5,3

Table 7.5 – *Iris : arbres à seuils flous et mesure de discrimination*

Les arbres construits par cette méthode comportent en moyenne 14,9 branches, ils ont une profondeur moyenne de 4,7 nœuds et une profondeur maximale de 6,5 nœuds. Le temps moyen de construction d'un arbre à partir d'une des bases d'apprentissage est inférieur à une seconde sur une station Sun UltraSparc.

7.3.3 Arbres construits avec l'entropie étoile

Les arbres ainsi construits obtiennent les taux d'erreur en classification donnés dans le tableau 7.6.

Les arbres construits par cette méthode comportent en moyenne 4 branches, ils ont une profondeur moyenne de 2,2 nœuds et une profondeur maximale de 2,8 nœuds. Le temps moyen de construction d'un arbre à partir d'une des bases d'apprentissage est inférieur à une seconde sur une station Sun UltraSparc.

Méthode	*Taux d'erreur*	*Écart type*
Stricte, Zadeh ou probabilistes	4,0%	4,7
Opérateurs de Lukasiewicz	4,7%	4,5

Table 7.6 – *Iris : arbres construits avec l'entropie étoile*

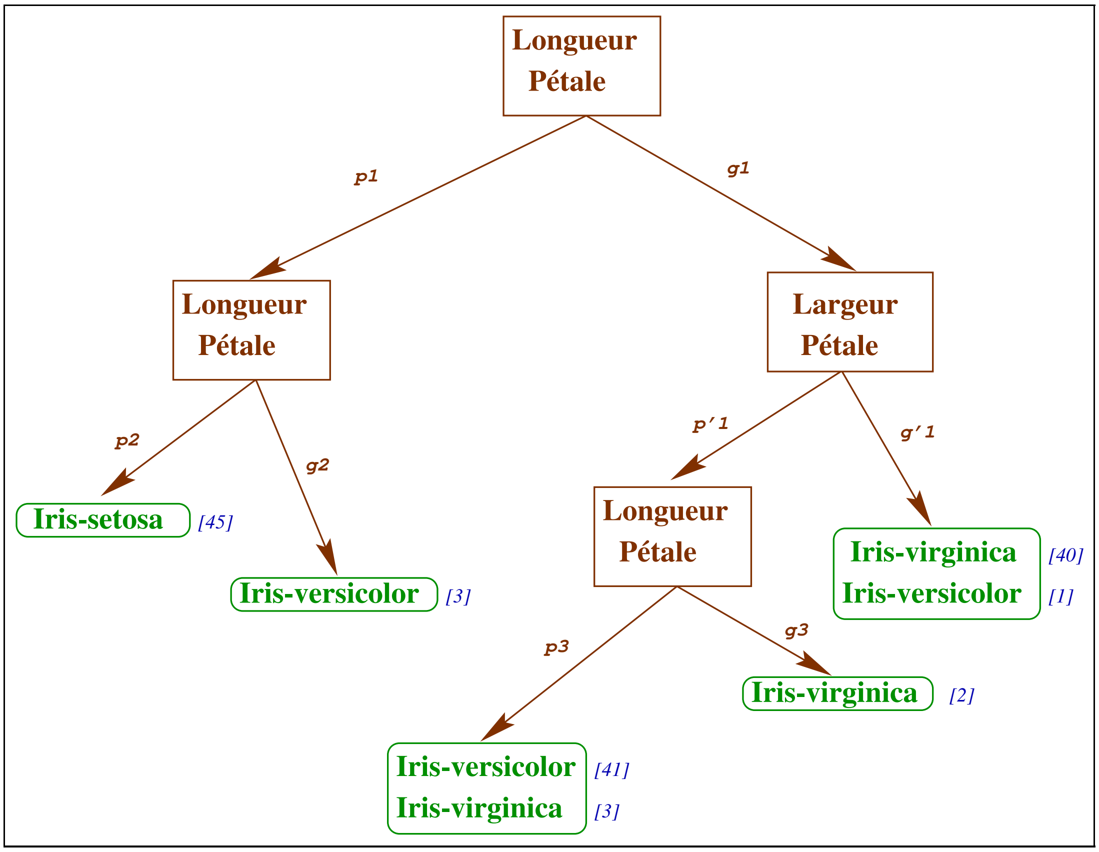


Figure 7.2 – *Données des Iris : arbre de décision flou construit*

Un exemple des arbres de décision flous construits pour cette base (*cf.* Annexe F) est donné dans la Figure 7.2.

Cet arbre se transcrit par la base de règles floues suivante :

```
R1  si la longueur du pétale est p1 et p2 alors c'est un Iris-setosa
R2  si la longueur du pétale est p1 et g2 alors c'est un Iris-versicolor
R3  si la longueur du pétale est g1 et p3 et la largeur du pétale est p'1
    alors c'est un Iris-versicolor (93%) ou un Iris-virginica (7%)
R4  si la longueur du pétale est g1 et g3 et la largeur du pétale est p'1
    alors c'est un Iris-virginica
R5  si la longueur du pétale est g1 et la largeur du pétale est g'1
    alors c'est un Iris-virginica (98%) ou un Iris-versicolor (2%)
```

Les modalités floues utilisées ont été déterminées par le processus décrit au chapitre 4. Par exemple, les modalités construites pour les attribut *longueur* et *largeur* du pétale sont données dans la Figure 7.3.

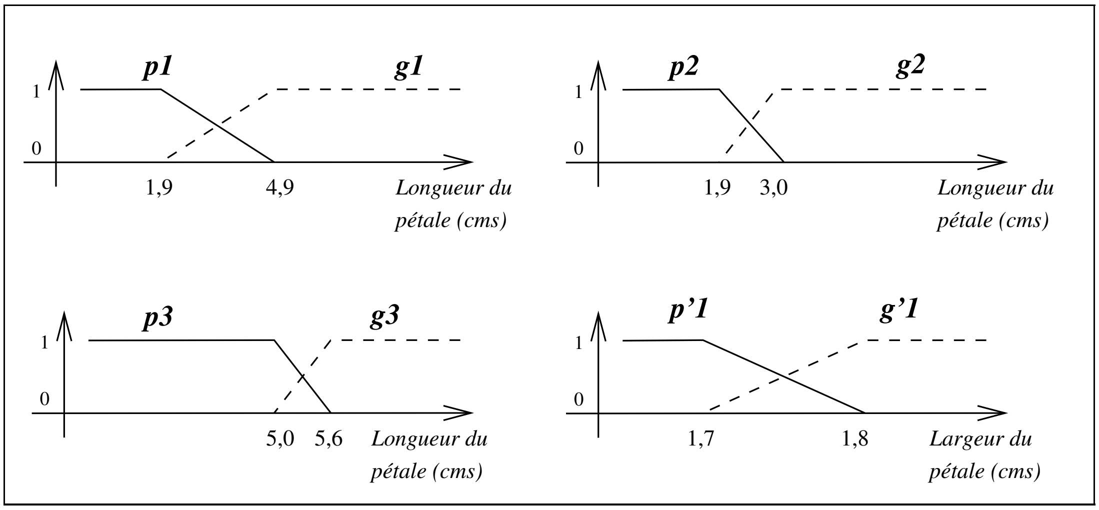


Figure 7.3 – *Données des Iris : modalités floues construites sur les attributs* longueur *et* largeur *du pétale*

7.4 Données des formes d'ondes

Les résultats suivants sont ceux obtenus sur la base des formes d'ondes de Breiman (Breiman et al., 1984). Dans ces données, les descriptions de chaque exemple comportent 21 attributs numériques et il y a trois classes à reconnaître. Comme méthode de construction d'arbres de décision classiques, on cite encore les résultats obtenus par l'algorithme C4.5 de (Quinlan, 1996). Pour cette base, cet algorithme produit des arbres de 44,6 branches en moyenne qui obtiennent un taux d'erreur moyen de 27,3%.

7.4.1 Arbres à seuils flous et mesure de Shannon

Les arbres ainsi construits obtiennent les taux d'erreur en classification donnés dans le tableau 7.7.

Méthode	*Taux d'erreur*	*Écart type*
Stricte	24,1%	2,3
Opérateurs de Zadeh, de Lukasiewicz ou probabilistes	24,2%	2,1

Table 7.7 – *Formes d'ondes : arbres à seuils flous et mesure de Shannon*

Sans mécanisme d'élagage, les arbres construits par cette méthode comportent en moyenne 74,8 branches, ils ont une profondeur moyenne de 6,9 nœuds et une profondeur maximale de 10,8 nœuds. Le temps moyen de construction d'un arbre à partir d'une des bases d'apprentissage, comportant 4500 exemples, est de 12 secondes, sur une station Sun UltraSparc.

7.4.2 Arbres à seuils flous et mesure de discrimination

Les arbres ainsi construits obtiennent les taux d'erreur en classification donnés dans le tableau 7.8.

Méthode	*Taux d'erreur*	*Écart type*
Stricte	24,8%	1,6
Opérateurs de Zadeh, de Lukasiewicz ou probabilistes	24,8%	1,9

Table 7.8 – *Formes d'ondes : arbres à seuils flous et mesure de discrimination*

Les arbres construits par cette méthode comportent en moyenne 70,2 branches, ils ont une profondeur moyenne de 7,2 nœuds et une profondeur maximale de 12,9 nœuds. Le temps moyen de construction d'un arbre à partir d'une des bases d'apprentissage est inférieur à une seconde sur une station Sun UltraSparc.

7.4.3 Arbres construits avec l'entropie étoile

Les arbres ainsi construits obtiennent les taux d'erreur en classification donnés dans le tableau 7.9.

Méthode	*Taux d'erreur*	*Écart type*
Stricte	26,7%	1,6
Opérateurs de Zadeh	23,3%	2,3
Opérateurs probabilistes	21,8%	2,5

Table 7.9 – *Formes d'ondes : arbres construits avec l'entropie étoile*

Les arbres construits par cette méthode comportent en moyenne 66,9 branches, ils ont une profondeur moyenne de 6,2 nœuds et une profondeur maximale de 7,9 nœuds. Le temps moyen de construction d'un arbre à partir d'une des bases d'apprentissage est de 2 minutes et 23 secondes sur une station Sun UltraSparc.

7.5 Données électriques

Les derniers résultats que nous présentons concernent des données issues d'une application réelle du domaine des réseaux électriques. Ces données nous ont été fournies par M. Wehenkel, et une description complète peut en être trouvée dans (Boyen et Wehenkel, 1996; Wehenkel et Pavella, 1996). Chaque exemple est décrit par 17 attributs numériques à valeurs réelles. La classe à trouver est, à l'origine, une valeur réelle qui a été discrétisée en deux classes symboliques.

Des premiers tests ont été réalisés en prenant une base d'apprentissage de 2000 exemples et une base de test de 1000 exemples. Comme valeur de discrétisation de la classe, la valeur $2,4$, introduite par (Boyen et Wehenkel, 1996; Wehenkel et Pavella, 1996) dans leurs travaux, a d'abord été utilisée. Avec cette valeur, les arbres à seuils flous obtiennent de meilleurs résultats en classification que les arbres construits avec l'entropie étoile. Mais, il faut alors noter que la discrétisation de la classe sur la valeur $2,4$ ne donne pas une répartition équitable des données.

Nous avons donc choisi de discrétiser la classe à l'aide de la valeur $3,5$ qui permet une répartition en proportion équitable des exemples des bases d'apprentissage et de test de chaque côté de ce seuil. Là encore, les résultats des arbres à seuils flous étaient encore légèrement meilleurs que ceux des arbres construits avec l'entropie étoile.

Du bruit a alors été introduit pour essayer de mettre en évidence les avantages de la méthode de construction basée sur l'entropie étoile, mais cela ne l'a pas avantagé non plus.

Finalement, c'est en choisissant le fichier des 1000 exemples comme base d'apprentissage et le fichier des 2000 exemples comme base de test, que la méthode basée sur l'entropie étoile s'est distinguée de la méthode à seuils flous.

La déduction immédiate de ces essais et que, dans ce type de données, la méthode basée sur l'entropie étoile est déterminante quand les données d'apprentissage ne couvrent pas complètement l'espace de représentation des données. Quand cette base est trop représentative, les méthodes moins floues sont quasiment optimales car les exemples d'apprentissage rendent compte entièrement du phénomène à apprendre. Par contre, dans ce cas, une méthode complètement floue à beaucoup plus de difficultés à rendre compte exactement de la base, car sa nature la pousse à généraliser de façon excessive les données d'apprentissage dont elle dispose.

Les résultats que nous donnons ici ont été obtenus en discrétisant la classe en utilisant la valeur $3,5$ comme valeur de coupure sur l'ensemble des valeurs de la classe. Cette valeur permet de séparer en deux sous-ensembles d'exemples de même taille. La base d'apprentissage est donc composée de 1000 exemples et la base de test de 2000 exemples.

7.5.1 Arbres à seuils flous et mesure de Shannon

Les arbres ainsi construits obtiennent les taux d'erreur en classification donnés dans le tableau 7.10.

Méthode	*Taux d'erreur*
Stricte	8,25%
Opérateurs de Zadeh ou probabilistes	8,35%
Opérateurs de Łukasiewicz	8,1%

Table 7.10 – *Données électriques : arbres à seuils flous et mesure de Shannon*

Les arbres construits par cette méthode comportent en moyenne 38 branches, ils ont une profondeur moyenne de 6,13 nœuds et une profondeur maximale de 10 nœuds. Le temps moyen de construction d'un arbre à partir d'une des bases d'apprentissage est d'environ deux secondes sur une station Sun UltraSparc.

7.5.2 Arbres construits avec l'entropie étoile

Les arbres ainsi construits obtiennent les taux d'erreur en classification donnés dans le tableau 7.11.

Méthode	*Taux d'erreur*
Stricte	8,4%
Opérateurs de Zadeh	7,7%
Opérateurs de Lukasiewicz	7,55%
Opérateurs probabilistes	7,1%

Table 7.11 – *Données électriques : arbres construits avec l'entropie étoile*

Les arbres construits par cette méthode comportent en moyenne 19 branches, ils ont une profondeur moyenne de 4,63 nœuds et une profondeur maximale de 7 nœuds. Le temps moyen de construction d'un arbre à partir d'une des bases d'apprentissage est d'environ quatre secondes sur une station Sun UltraSparc.

Par exemple, l'arbre de décision flou construit (*cf.* Annexe F) donne, entres autres, les deux règles floues suivantes :

```
R1 : si l'attribut A3 est p1 et p2 et l'attribut A5 est p'1
     alors la classe est 0 (inférieure à 3,5)
R2 : si l'attribut A3 est p1 et p2 et
     l'attribut A5 est g'1 et l'attribut A6 est g"1
     alors la classe est 1 (supérieure à 3,5)
```

Les modalités floues utilisées ont été déterminées par le processus décrit au chapitre 4. Par exemple, les modalités construites pour les attributs des deux règles **R1** et **R2** sont données dans la Figure 7.4.

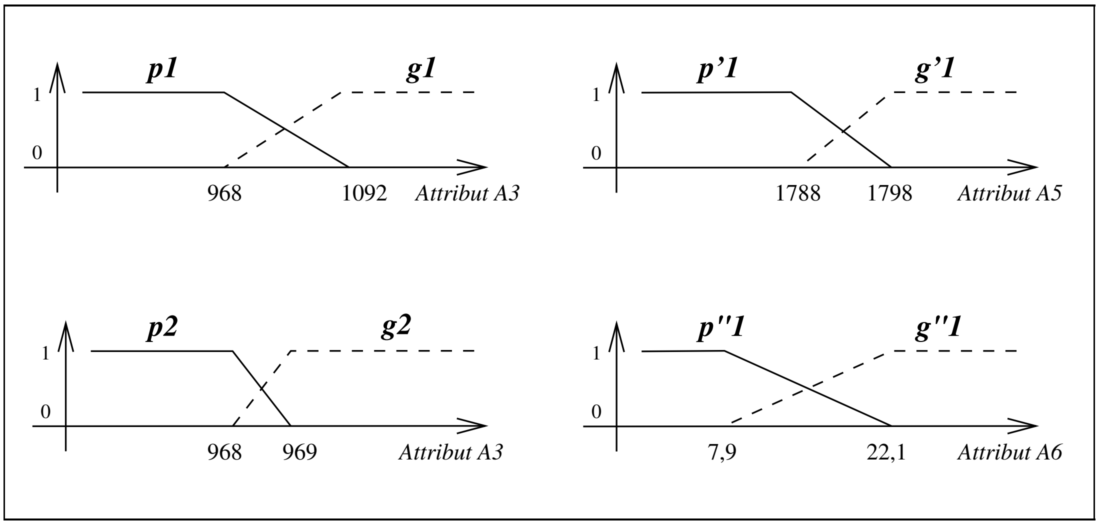


Figure 7.4 – *Données électriques : modalités floues pour les règles présentées*

7.6 Conclusion

Les résultats présentés permettent de dégager quelques éléments de conclusion sur les différentes méthodes testées. Il apparaît que les arbres construits dépendent de la mesure de discrimination utilisée sur certains aspects. Les arbres de décision flous, construits en utilisant la mesure d'entropie étoile, sont d'une taille souvent beaucoup plus réduite que les arbres de décision construits dans une mesure de discrimination classique. Cette différence en taille peut s'expliquer par l'élagage *naturel* que produit la méthode de construction d'arbres flous. Ainsi, les arbres obtenus sont plus explicites et compréhensibles.

En terme de taux d'erreur en classification, les résultats obtenus par les méthodes floues montrent l'efficacité de telles méthodes. Par contre, il semblerait ne pas exister de méthode d'agrégation optimale pour tous les types de problèmes. Le choix des opérateurs à utiliser apparaît spécifique au problème considéré.

Conclusion et perspectives

On a beau colmater, c'est toujours une surprise, l'étendue de notre ignorance.

D. Pennac – *Monsieur Malaussène*

Dans cette thèse, nous avons présenté une méthode d'apprentissage inductif en présence de données imprécises. L'apprentissage inductif est réalisé par la construction d'arbres de décision flous qui permet de construire une base de règles. Des améliorations ont été apportées pour une meilleure prise en compte de l'imprécision et pour l'apport de techniques automatiques. Un système informatique, composé de deux applications génériques *Salammbô* et *Tanit*, a été réalisé et testé sur différentes bases d'apprentissage.

Les travaux développés dans cette thèse laissent pourtant encore entrevoir de nombreuses perspectives de recherches qui demandent à être conduites afin d'atteindre notre objectif principal de conception d'un système entièrement autonome.

Notre contribution

Dans cette thèse, à partir de la méthode d'apprentissage inductif par arbres de décision, nous avons étudié les meilleurs apports possibles de la théorie des sous-ensembles flous pour la prise en compte de l'imprécision.

À l'aide d'une nouvelle formalisation des algorithmes de construction d'arbres de décision, il a été possible de remarquer que les systèmes actuels de construction d'arbres de décision ne se différencient essentiellement que sur le choix de la mesure de discrimination qu'ils utilisent pour sélectionner les attributs entre eux. Le formalisme que nous avons introduit exhibe les paramètres fondamentaux des algorithmes de construction d'arbres de décision. Ce sont ces paramètres qui doivent être modifiés pour la prise en considération de l'imprécision des données, permettant ainsi, la construction d'un arbre de décision flou.

Ensuite, nous avons formalisé notre méthode d'utilisation d'un arbre de décision flou pour le classement d'un nouvel objet inconnu. Cette méthode est basée sur l'utilisation d'une mesure de satisfiabilité qui permet de rendre compte de la ressemblance des valeurs du nouvel objet et des modalités présentes dans les tests des nœuds de l'arbre. Les degrés sont combinés entre eux, selon une méthode d'agrégation spécifique à la mesure de satisfiabilité choisie, pour donner un degré d'appartenance à chaque classe à reconnaître.

Une analogie entre la méthode que nous proposons et la méthode d'inférence a été effectuée afin de remarquer la similarité des deux processus d'inférence.

De plus, l'étude de la stabilité d'une telle méthode d'inférence à partir de données floues a été effectuée afin de faire ressortir la robustesse en terme de décision d'un tel système.

Ces méthodes de construction et d'utilisation d'arbres de décision flous ont donné lieu à la réalisation du programme informatique *Salammbô*, et ont donc pu être ainsi testées sur différentes bases d'apprentissage.

Dans notre optique première de conférer une plus grande autonomie de déroulement pour un tel système d'apprentissage, et afin d'améliorer encore les performances des arbres de décision flous, nous avons ensuite proposé des améliorations à différents niveaux du processus d'apprentissage.

Tout d'abord, la construction d'un arbre de décision flou utilisant une partition floue sur l'ensemble des valeurs des attributs numériques-symboliques, nous proposons une méthode automatique de construction d'une telle partition.

Cette méthode est basée sur l'utilisation des opérateurs de la théorie de la morphologie mathématique formalisés à l'aide de la théorie des langages formels. Elle permet à Salammbô d'être entièrement autonome dans le traitement de tout type de base d'apprentissage.

La prise en compte de bases d'apprentissage possédant plus de deux classes, est un problème qui dégrade souvent les performances des arbres de décision en général. Pour pallier ce type de problèmes, nous avons été amenés à proposer une architecture multi-classifieurs fondée sur la multiplication des arbres de décision flous. L'application *Tanit* pour construire une forêt d'arbres de décision flous a alors été introduite. Tanit est basée sur l'utilisation de plusieurs Salammbô, exécutées de façon parallèle, chacune devant apprendre à ne reconnaître que deux classes. À la différence des arbres de décision classiques, les arbres flous construits par les Salammbô retournent des degrés d'appartenance à chaque classe, Tanit profite alors de la gradualité des décisions des Salammbô pour déduire la classe de tout nouvel objet.

Ce type d'architecture multi-classifieurs nous a demandé d'étudier le problème de l'agrégation de données multi-sources, et, en particulier, de nous interroger sur la notion d'indépendance entre les différents classifieurs que sont les Salammbô. Finalement, nous avons pu vérifier l'indépendance de nos Salammbô en adoptant la définition de l'indépendance introduite par Kampé de Fériet, ce qui nous a permis d'étendre les choix possibles des méthodes d'agrégation des résultats donnés par les Salammbô.

Des résultats de l'application de cette stratégie de construction de forêt d'arbres de décision flous, implémentée dans le système Tanit, sont donnés sur différentes bases d'apprentissage.

Le dernier type d'améliorations que nous avons décidé d'apporter concerne la mesure de discrimination utilisée dans le processus de construction d'un arbre de décision. La classification des mesures habituellement utilisée nous paraissant trop fortement imprégnée de la théorie de l'information, nous avons suivi, encore une fois, les travaux de Kampé de Fériet pour proposer une nouvelle méthode d'étude des mesures à utiliser dans la construction d'un arbre de décision. Cette méthode permet de valider une mesure pour vérifier son adéquation afin de mesurer le pouvoir discriminant d'un attribut et de construire de nouvelles mesures de discrimination. La hiérarchie de fonctions que nous proposons exhibe différents niveaux de mesures, ce qui facilite l'introduction des éléments de la théorie des sous-ensembles flous pour prendre en compte l'imprécision des données.

Perspectives

Certes, nos travaux devront être poursuivis pour construire un système complètement autonome. Tout d'abord, la formalisation du processus de construction d'arbres de décision demande à être poursuivie et menée à terme, tous les aspects de l'algorithme de construction doivent être formalisés afin d'être plus facilement étudiés. De plus, une telle formalisation doit permettre de faire ressortir tous les éléments fondamentaux, à tous niveaux, d'un tel algorithme d'apprentissage.

Dans le cadre de l'utilisation de l'arbre de décision flou en classification, l'étude qui a été entamée sur les liens entre la mesure de satisfiabilité et les opérateurs d'inférence et d'agrégation, demande à être poursuivie et analysée plus en détail. Des liens existent entre la mesure de satisfiabilité et les opérateurs à utiliser, ces liens seront étudiés pour arriver à paramétrer complètement les choix de mesures à faire pour la méthode de classification.

Dans le cadre de l'inférence de partitions floues, il reste encore des travaux à compléter. Les fonctions utilisées pour filtrer les partitions floues peuvent être étudiées plus en détail. Des comparaisons de différentes fonctions de filtrage peuvent être conduites pour clarifier le choix d'une telle fonction.

De même, l'agrégation utilisée dans les forêts d'arbres de décision flous demande à être complétée et affinée. D'autres méthodes d'agrégation doivent être introduites dans Tanit afin de comparer leurs avantages respectifs.

Finalement, l'étude des mesures de discrimination doit être complétée par l'application du modèle hiérarchique de fonctions à d'autres mesures de discrimination existantes. Ce modèle doit aussi être appliqué aux mesures de discrimination et la prise en compte de données numériques-symboliques.

Pour améliorer encore la prise en compte des données imprécises d'une part, et l'autonomie de notre système d'apprentissage d'autre part, nous prévoyons de poursuivre nos recherches après cette thèse dans les directions suivantes :

Sur les mesures de discrimination

Dans le cadre de la construction des arbres de décision flous, l'étude des mesures de discrimination doit encore être poursuivie. Une qualification plus poussée des caractéristiques des mesures de discrimination doit être menée de façon plus approfondie. La hiérarchie des fonctions $\mathcal{FGH}$ est une étape vers l'étude du comportement des mesures de discrimination. La $\mathcal{F}$-fonction associée à une mesure conditionne grandement sa manière de réagir aux variations des données. C'est cette façon de réagir qui doit permettre de distinguer les mesures entres elles et de déterminer quel type de mesures doit être adapté à un type de base particulier. Cette étude doit conduire à une plus grande autonomie du système de construction qui aura alors les moyens de choisir, par lui-même, la mesure de discrimination la plus adaptée à la base d'apprentissage qu'il est en train de traiter.

D'un autre côté, la remarque formulée au chapitre 6, sur la possibilité d'utiliser une mesure de discrimination floue est, à notre avis, une voie de recherche très intéressante à poursuivre. Une mesure floue du calcul du pouvoir discriminant doit, en effet, pouvoir assurer une méthode de comparaison plus graduelle des attributs entre eux.

Sur la robustesse de l'apprentissage par arbre flou

Après l'étude de la stabilité en classification que nous avons menée, il reste à étudier la stabilité des arbres de décision flous lors de leur construction. La robustesse apportée par la méthode de discrétisation floue et par la construction des arbres en présence de données imprécises, se doit d'être examinée en détail pour comparer les avantages d'une telle méthode vis-à-vis des méthodes classiques.

Sur le système des forêts d'arbres de décision flous

Des perspectives de recherches apparaissent aussi dans le cadre des forêts d'arbres de décision flous. En plus de l'étude des méthodes d'agrégation qui se doit d'être approfondie, les aspects multi-agents esquissés dans la section 5.6 du chapitre 5 sont encore à développer. Les rapports entre Tanit et les Salammbô qu'elle crée peuvent donner lieu à des échanges plus complexes qu'ils ne le sont actuellement. Au lieu de s'échanger des degrés d'appartenance aux classes, les Salammbô et Tanit devraient avoir la possibilité de s'échanger des informations plus évoluées qui leur permettraient d'entamer un vrai dialogue pour classer un nouvel objet, ce qui, de plus, les autoriseraient à améliorer chacune leur arbre de décision en puisant des conseils auprès de leurs consœurs.

Sur la validation du système et la découverte de connaissances

Les résultats généralement donnés pour les arbres de décision ne sont que des résultats en pourcentage de bonnes classifications ou de taille des arbres construits. Mais, souvent, avec ce type de validation, les arbres de décision flous ont beaucoup de mal à faire valoir leurs points forts qui les démarquent réellement des méthodes classiques d'apprentissage inductif. C'est pourquoi, nous pensons qu'un nouveau type de validation de système doit être défini.

De tels systèmes d'apprentissage en présence de données imprécises doivent être validés dans un environnement existant. Comme nous l'avons précisé en introduction de cette thèse, un environnement informatique est de différents ordres : le monde réel dans le cadre d'un système robotique, une base de données, un système distribué comme un réseau de machines... L'intérêt d'un système d'apprentissage automatique susceptible d'apprendre à partir de tous types de connaissances, c'est d'être testé dans tous types d'environnements.

C'est dans un environnement d'un tel type que le véritable travail de découverte de connaissances ou la fouille de données peut s'effectuer. Les arbres de décision sont un outil prépondérant pour la restructuration de connaissances à partir d'un ensemble de données. Les arbres de décision flous leur apportent une dimension supplémentaire qui les autorise à traiter tous les types de données existants et qui leur confère une plus grande robustesse dans leur décision.

La masse de données disponibles dans le monde croît, sans cesse, de jour en jour, à une vitesse telle que l'être humain n'est plus capable de l'appréhender. L'ordinateur est le seul recours pour transformer cette masse de données en information exploitable, c'est en le rendant autonome et en l'autorisant à traiter tous types de connaissances qu'il sera alors capable de mener à bien une telle tâche.

Annexes

Annexe A

Éléments de la théorie des langages formels

La partie sur la mise en évidence automatique de partitions floues *(chapitre 4)* utilise quelques éléments de la théorie des langages formels. Les notions de base nécessaires à la bonne compréhension de cette partie sont présentées dans cette annexe. De plus amples développements sur cette théorie pourront être trouvés dans les ouvrages de (Ginsburg, 1966) ou, surtout, de (Autebert, 1987).

La théorie des langages formels formalise le langage pour le manipuler plus aisément, ainsi, les notions usuelles de la langue se retrouvent dans les fondements de cette théorie.

Définition 9 *Un* alphabet *est un ensemble fini, non vide, d'éléments.*

Les éléments d'un alphabet prennent le nom de *lettres*. Soit A un alphabet, un *mot* de A est une suite finie, ou séquence, $(x_1, x_2, ..., x_n)$ de lettres de A. $(x_1, x_2, ..., x_n)$ est noté plus simplement $x_1x_2...x_n$. Un mot particulier est le mot *vide* (), composé d'aucune lettre et noté ε. L'ensemble des mots de A est noté A^*, et, par définition, $\varepsilon \in A^*$.

Soient v et w deux mots de A^*.

Définition 10 *La* concaténation *de v et w est le mot $v.w \in A^*$ tel que $v = x_1x_2...x_n$, $w = y_1y_2...y_p$ et $v.w = x_1x_2...x_ny_1y_2...y_p$*

La concaténation est une *opération interne* qui admet le mot ε comme élément neutre. Généralement, $v.w$ est noté vw.

On appelle x^n une suite de n lettres identiques x dans un mot : $\underbrace{x...x}_{n} = x^n$. Une séquence x^n dans un mot v est dite *complète* si elle ne peut pas être augmentée dans v. Par exemple, dans $v = abb^3ac^4a$, la suite b^3 n'est pas une séquence complète, mais c^4 en est une.

Pour travailler sur les mots d'un langage, on définit des fonctions de transformations sur ces mots. Ces fonctions portent le nom de système de réécriture.

Définition 11 *Un* système de réécriture *est un ensemble fini de couples ordonnés de mots sur un alphabet donné.*

Par exemple, sur $A = \{a, b, c\}$, $\{(aa, a), (b, bb), (ac, b), (a, \varepsilon)\}$ définit un système de réécriture. Chaque couple (u, v) est une *règle de réécriture* (ou de *production*). Un système de réécriture est généralement noté de la façon suivante :

Exemple de système de réécriture:

$$\begin{array}{lcl@{\qquad\qquad}lcl} aa & \longrightarrow & a & b & \longrightarrow & bb \\ ac & \longrightarrow & b & a & \longrightarrow & \varepsilon \end{array}$$

On dit que v se *réécrit* en w s'il existe une suite de règles de réécriture qui permette de passer de v à w. On note alors $v \overset{*}{\longrightarrow} w$. La notation * sur la flèche $\longrightarrow$ autorise l'utilisation de 0 règle ou plus. w est *irréductible* s'il ne peut pas se réécrire en un autre mot. v possède un irréductible s'il existe un irréductible w tel que $v \overset{*}{\longrightarrow} w$.

Les définitions et théorèmes suivants permettent d'étudier le comportement d'un système de réécriture. Les preuves des théorèmes peuvent être facilement trouvées (Autebert, 1987).

Définition 12 *Un système de réécriture est* nœthérien *s'il n'existe pas de séquence infinie de règles de réécriture.*

Par exemple, le système donné en exemple n'est pas nœthérien car à partir du mot b, un nombre infini de règles peut être appliqué.

Définition 13 *Un système de réécriture est* confluent *si, pour tout triplet de mots u, v et w tels que $u \overset{*}{\longrightarrow} v$ et $u \overset{*}{\longrightarrow} w$, il existe un mot t tel que $v \overset{*}{\longrightarrow} t$ et $w \overset{*}{\longrightarrow} t$.*

Définition 14 *Un système de réécriture est dit* local confluent *si, pour tout triplet de mots u, v et w tels que $u \longrightarrow v$ et $u \longrightarrow w$, il existe un mot t tel que $v \overset{*}{\longrightarrow} t$ et $w \overset{*}{\longrightarrow} t$.*

Théorème A.1 *Avec un système nœthérien, tout mot possède un irréductible.*

Théorème A.2 *Avec un système confluent, aucun mot ne possède plus d'un irréductible.*

Théorème A.3 *Un système de réécriture qui est nœthérien et local confluent est confluent.*

Les transductions rationnelles et les grammaires sont des formes particulières de systèmes de réécriture. Une transduction rationnelle, qui est un automate avec sortie, permet d'exprimer des systèmes de réécriture dont les règles de production sont simples. Une grammaire permet d'exprimer des systèmes de réécriture plus complexes.

Les transductions rationnelles

Une *transduction rationnelle*, ou plus simplement *transduction*, est un sextuplet $< \mathcal{A}, \mathcal{B}, S, I, E, \delta >$ où $\mathcal{A}$ est l'alphabet d'entrée, $\mathcal{B}$ est l'alphabet de sortie, S est un ensemble (fini) d'états, $I \subset S$ est l'ensemble des états de départ de la transduction,

$E \subset S$ est l'ensemble des états terminaux de la transduction et $\delta \subset S \times \mathcal{A}^* \times \mathcal{B}^* \times S$ est une fonction de transition.

Une transduction lit un mot d'entrée $v \in \mathcal{A}^*$ et le réécrit en un mot correspondant $w \in \mathcal{B}^*$. La réécriture s'effectue séquentiellement, de la première lettre de v jusqu'à la dernière, par le déplacement d'une tête de lecture lettre par lettre. Les règles de réécriture pour générer w se basent sur δ. Soit $(s_i, z, t, s_j) \in \delta$, avec $s_i, s_j \in S, z \subset \mathcal{A}^*$ et $t \subset \mathcal{B}^*$, (s_i, z, t, s_j) est appelée *transition* de la transduction. Si s_i est l'état courant et que l'on peut lire z dans v (*ie.* z est composé par les lettres successives qui viennent juste après la tête de lecture), on le remplace par t et l'état courant devient s_j. Par convention, on utilise \$ pour marquer la fin du mot d'entrée, et ε (le *mot vide*) est utilisé lorsque l'on ne veut rien écrire.

Par exemple, soient $\mathcal{A} = \{a, b\}, \mathcal{B} = \{0, 1\}$, $S = \{\mathrm{S1}, \mathrm{S2}, \mathrm{S3}\}$, $I = \{\mathrm{S1}\}$, $E = \{\mathrm{S3}\}$ et $\delta = \{(\mathrm{S1}, a, 0, \mathrm{S1}),\ (\mathrm{S1}, b, \varepsilon, \mathrm{S2}),\ (\mathrm{S2}, a, 0, \mathrm{S1}),\ (\mathrm{S2}, b, 1, \mathrm{S2}),\ (\mathrm{S1}, \$, \$, \mathrm{S3}),\ (\mathrm{S2}, \$, \$, \mathrm{S3})\}$

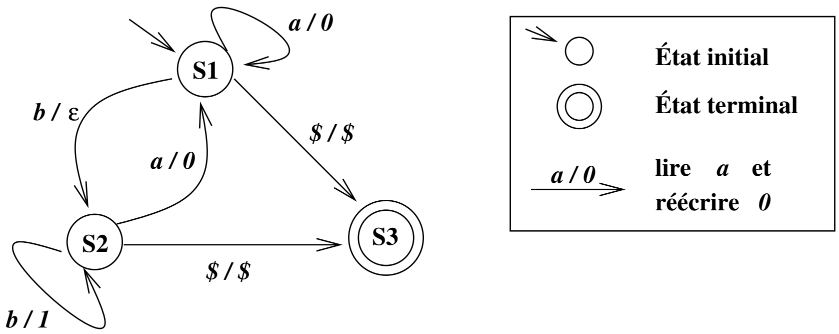


Figure A.1 – *Exemple de transduction rationnelle*

Cette transduction, comme tout automate, se représente graphiquement de façon simple et visuelle (Figure A.1). Avec cette transduction, le mot $v = abbaabaab$ se réécrit comme suit : après la séquence d'états (S1, S1, S2, S2, S1, S1, S2, S1, S1, S2, S3), on obtient $w = 010000$ et, étant donné que l'état S3 est un état terminal et que l'on n'a plus rien à lire dans v, la réécriture est terminée.

Les grammaires contextuelles

Une grammaire est un système de réécriture particulier. Une grammaire est un quadruplet $G =< \mathcal{A}, \mathcal{B}, \mathcal{S}, \mathcal{P} >$ où $\mathcal{A}$ est un alphabet, $\mathcal{B} \subseteq \mathcal{A}$ est l'alphabet terminal, $\mathcal{S} \in \mathcal{A} - \mathcal{B} = \mathcal{N}$ est l'ensemble des symboles de départ, ou variables, et $\mathcal{P}$ est un ensemble fini de règles de production de la forme : $\alpha \longrightarrow \beta$ avec $\alpha \in \mathcal{A}^* N \mathcal{A}^*$ et $\beta \in \mathcal{A}^*$.

Il existe différents types de grammaires, recensés par la hiérarchie de Chomsky (Harrison, 1978), dont celui des grammaires contextuelles [1].

Une grammaire est dite *contextuelle* si ses règles de réécriture sont de la forme $\alpha V \gamma \longrightarrow \alpha\beta\gamma$ où $V \in \mathcal{N}$, $\alpha, \beta, \gamma \in \mathcal{A}^*$, ou de la forme $\mathcal{S} \longrightarrow \varepsilon$.

1. *Context-sensitive grammar*

Cela revient à dire que la réécriture de V en β dépend du contexte $(\alpha,\ \gamma)$ qui l'entoure. Par exemple, dans la grammaire contextuelle suivante, la lettre V se réécrit différemment en fonction des lettres qui l'entourent :

Exemple de grammaire contextuelle:

$$\begin{array}{lcl@{\qquad\qquad}lcl} aVa & \longrightarrow & aaa & bVb & \longrightarrow & b\varepsilon b \\ bVa & \longrightarrow & bca & aVb & \longrightarrow & acb \end{array}$$

Annexe B

Éléments de la théorie de l'évidence

Les notions de base de la théorie de l'évidence sont présentés dans cette annexe. À partir d'une répartition de masses de croyance sur des événements d'un univers donné, cette théorie autorise la construction de deux fonctions duales, la *croyance* et la *plausibilité* de sous-ensembles d'événements. Suivant le type de la répartition des masses de croyance, la théorie de l'évidence se généralise soit la théorie des probabilités, ou bien en théorie des possibilités. Les fondements de cette théorie de l'évidence sont donnés dans l'ouvrage (Shafer, 1976). Une vision globale et synthétique en est aussi donnée dans le livre (Bouchon-Meunier, 1995).

Fonction de masse

Les deux fonctions de base de la théorie de l'évidence se définissent sur un univers d'événements particulier appelé *cadre de discernement*. Un cadre de discernement (ou tout simplement *cadre*) est un ensemble dont les éléments peuvent être interprétés comme des réponses possibles à une question donnée et dont on sait ou on croit qu'une et une seule de ces réponses est correcte (Shafer, 1976; Shafer, 1987).

Sur un cadre donné, il est possible d'associer à chaque élément, ou groupe d'éléments, de ce cadre, un degré avec lequel il peut être raisonnable de croire en cet événement. Il est intéressant de respecter deux contraintes pour affecter ces degrés afin qu'ils forment une *fonction de masse*[1].

Définition 15 (Fonctions de masse) *Étant donné un univers fini* Ω*. Une* fonction de masse *est une fonction* $m : 2^{\Omega} \longrightarrow [0,1]$ *telle que :*

i) $\sum_{a \in 2^{\Omega}} m(a) = 1$

ii) $m(\emptyset) = 0$

Un cadre de discernement $\mathcal{F}$ associé à une fonction de masse m sur les éléments de ce cadre constitue un *corps d'évidence* (noté $(\mathcal{F}, m)$).

1. En anglais, Shafer utilise l'appellation plus explicite de "Basic Probability Assignment"

Fonctions de croyance et de plausibilité

À partir d'un corps d'évidence $(\mathcal{F}, m)$ donné, il est possible de définir deux fonctions particulières. La première fonction, la *fonction de croyance* Bel, *résume dans quelle mesure on peut croire en la réalisation de chaque événement, étant donné l'incertitude exprimée* (Bouchon-Meunier et Nguyen, 1996) :

Définition 16 (Fonction de croyance) *Étant donné un corps d'évidence $(\mathcal{F}, m)$. La fonction Bel définie par*

$$\forall a \in 2^{\mathcal{F}}, \quad Bel(a) = \sum_{b \in 2^{\mathcal{F}},\ b \subseteq a} m(b)$$

est appelée fonction de croyance *associée au corps d'évidence $(\mathcal{F}, m)$.*

Cette fonction admet une fonction duale, la *fonction de plausibilité* Pl :

Définition 17 (Fonction de plausibilité) *Étant donné un corps d'évidence $(\mathcal{F}, m)$. La fonction Pl définie par*

$$\forall a \in 2^{\mathcal{F}}, \quad Pl(a) = \sum_{b \in 2^{\mathcal{F}},\ b \cap a \neq \emptyset} m(b)$$

est appelée fonction de plausibilité *associée au corps d'évidence $(\mathcal{F}, m)$.*

Ces deux fonctions sont duales dans le sens où elles sont exprimables l'une par rapport à l'autre :

Propriété B.1 *Pour tout $A \subset \Omega$, et en notant $\bar{A}$, le complément de A (ie. $\bar{A} \subset \Omega$ tel que $A \cup \bar{A} = \Omega$ et $A \cap \bar{A} = \emptyset$) :*

$$Bel(A) = 1 - Pl(\bar{A})$$

La théorie de l'évidence admet comme cas particulier la théorie des probabilités dans le cas où les masses de croyances portent sur des éléments disjoints et élémentaires. Dans ce cas, la fonction de croyance et la fonction de plausibilité coïncident avec une mesure de probabilité. Dans le cas où les masses de croyances portent sur des éléments emboîtés, la fonction de croyance est une mesure de possibilité et la fonction de plausibilité est une mesure de nécessité, de la théorie des possibilités.

La règle de Dempster

Afin d'agréger des croyances portant sur un même cadre de discernement, Shafer (Shafer, 1976) propose l'utilisation de la règle de combinaison de Dempster, qui est dérivée de la règle de combinaison de témoignages de Hooper.

Connaissant deux fonctions de masse indépendantes m_1 et m_2, la règle de combinaison de Dempster permet de calculer une fonction de masse combinée m_{12} :

$$\forall a \subset 2^{\Omega}, \ m_{12}(a) = K \sum_{\forall b,c \subset 2^{\Omega},\ b \cap c = a} m_1(b)\, m_2(c)$$

(Où K est un facteur de normalisation).

Annexe C

Les modèles entropiques

Un modèle entropique est défini comme suit (Bouchon, 1988) :

Soit $\mathcal{U}$ un univers fini donné et soit $\mathcal{C}$ une famille de sous-ensembles flous caractéristiques de $\mathcal{U}$.

$\forall n \in \mathbb{N}$, on considère les familles :

- $P_n = \{(p_1, ..., p_n) | p_i \geq 0 \; \forall i, \text{ et } \sum_i p_i = 1\}$ de distribution de probabilités sur $\mathcal{U}$ à travers une loi de probabilité p définie sur $\mathcal{U}$.
- $F_n = \{(x_1, ..., x_n) | x_i \in \mathcal{C} \; \forall i\}$ de n-tuples de $\mathcal{C}$.

Définition 18 *Un modèle entropique est une suite de fonctions définies sur $F_m \times P_m \times F_n \times P_n$, à valeurs dans $\mathbb{R}$, ($\forall m, n \in \mathbb{N}$) qui satisfait à plusieurs "requis" de base, et est notée :*

$$J_{m\,n}(X, Q, Y, P) = \left(\begin{pmatrix} x_1, ..., x_m \\ q_1, ..., q_m \end{pmatrix} \begin{pmatrix} y_1, ..., y_n \\ p_1, ..., p_n \end{pmatrix} \right)$$

Les propriétés d'un modèle entropique vont permettre son classement dans l'un ou l'autre des deux types définis plus haut.

On suppose que Q et P dépendent respectivement de X et Y par l'intermédiaire d'une application R telle que $Q = R(X)$ et $P = R(Y)$ et que, de plus, X soit obtenu par l'intermédiaire d'une transformation T à partir de Y ($X = T(Y)$). Les propriétés considérées sont :

La symétrie (S)

Le remplacement de $Y = (y_1, ..., y_n)$ par $Y' = (y_{\rho(1)}, ..., y_{\rho(n)})$ pour toute permutation ρ de $\{1, ..., n\}$ et de $P = R(Y)$ par $P' = R(Y')$ ne modifie pas la valeur du modèle entropique.

L'expansiabilité (E)

Le remplacement de $Y = (y_1, ..., y_n)$ par $Y' = (y'_1, ..., y'_n, V)$ pour un élément "vide" V de $\mathcal{C}$ ne change pas la valeur du modèle entropique .

La continuité (C)

La valeur du modèle entropique est continue par rapport à la pondération des composants de Y.

La récursivité (R)

Le remplacement de $Y = (y_1, ..., y_n)$ par $Y' = (y_1 \cup y_2, y_3, ..., y_n)$ et de $P = (p_1, ..., p_n)$ par $P' = (p_1 + p_2, p_3, ..., p_n)$ réduit la valeur du modèle entropique proportionnellement au modèle entropique correspondant au 4-uplet $(T(Y''), R(T(Y'')), Y'', P'')$ avec $Y'' = (y_1, y_2)$ et $P'' = (\frac{p_1}{p_1+p_2}, \frac{p_2}{p_1+p_2})$.

Le modèle entropique a une plus grande valeur quand la description des faits est plus détaillée.

L'additivité (A)

Si on considère $Y = (y_1, ..., y_n)$ associé à $P = (p_1, ..., p_n)$ et $Y' = (y'_1, y'_2, ..., y'_n)$ associé à $P' = (p'_1, ..., p'_n)$ qui sont non-interactifs en regard d'une opération $\cap$ telle que $Y * Y' = (y_1 \cap y'_1, ..., y_j \cap y'_j, ...)$ soit associé à $P * P' = (p_1 p'_1, ..., p_i p'_j, ..., p_n p'_n)$. Alors l'élément combiné $Y * Y'$ fournit une valeur du modèle entropique égale à la somme des 2 valeurs fournies par Y et Y'.

Cette propriété indique que la valeur du modèle entropique correspondant à une expérience à l'intersection des informations fournies par 2 expériences non-interactives, est exactement égale à la somme des valeurs qui seraient obtenues par le biais de ces 2 expériences.

La monotonie (M)

Si un ordre partiel $\prec$ est défini sur $F_n, \forall n$ et deux éléments $Y = (y_1, ..., y_n)$ et $Y' = (y'_1, y'_2, ..., y'_n)$ sont tels que $Y \prec Y'$ et $P = R(Y) = R(Y')$, alors la valeur du modèle entropique correspondant à Y n'est pas plus grande que celle correspondant à Y' (elle lui est inférieure ou égale).

Cette propriété concerne un changement possible dans les indications obtenues pendant 2 expériences différentes, les distributions de probabilités restant identiques. Cela peut s'interpréter comme une amélioration dans la description des composants y_i observés.

Annexe D

Extraits du code de Salammbô

D.1 Les données

Structure de données interne pour les exemples

```
typedef struct nombre_valeur  /* Une valeur d'attribut */
{ int nombre;
  struct valeur_attribut *nomClasse;
  struct nombre_valeur *suiv;
} ptrNombreValeur;

typedef struct type_intervall  /* Intervalle de valeurs */
{ double valeurInf, valeurSup;
  double lookaheadSup, lookaheadInf;
  char leTestInf, leTestSup;
  int nombreDelement;
  struct valeur_attribut *borneInf, *borneSup;
  struct type_intervall *suiv;
} t_intervalle;

typedef struct valeur_attribut /* Champs de la valeur d'un attribut */
{ char valeurContinue;  /* VRAI ou FAUX */
  char typeValeur;/*ENTIER, REEL, INTERVALLE, POINTEUR, SEF, INCONNU*/
  int ordre;  /* pour numéroter les classes */
  char lettreAssociee; /* pour les morpho-maths */
  union u_val
  { int iValeur;
    double fValeur;
    t_intervalle *pIntervalle;
    char *pValeur;
  } uValeur;
  brique* sefValeur; /* sefs comme valeur */
  int occurrence;
  brique *sefAssocie;
  struct valeur_attribut *suiv;
  struct liste_attribut *attribut;
  ptrNombreValeur *nombreParClasse;
```

```
  struct pile_exemples *exemple, *lastExemple;
} ptrValeurAttribut;

typedef struct liste_attribut /* Un attribut */
{ unsigned int numero;       /* No d'ordre dans la saisie */
  int cardinal;  /* Nombre de valeurs de l'attribut */
  ptrValeurAttribut *valeur;
  struct liste_attribut *suiv;
} ptrAttribut;

typedef struct champ_exemple  /* Champs d'un exemple */
{ ptrValeurAttribut *valeur;
  struct champ_exemple *suiv;
} ptrChampExemple;

typedef struct pile_exemples  /* Un exemple */
{ unsigned int numero;      /* No d'ordre dans la saisie */
  double degreApp; /* son degré d'appartenance */
  double valeur; /* pour s'y retrouver durant la duplication */
  ptrChampExemple *champ;
  struct pile_exemples *suiv;
} ptrExemple;
```

Les sous-ensembles flous :

```
typedef struct str_brique  /* Un sef est une list de points (x, f(x)) */
        { point p;
          struct str_brique *suiv;
        } brique;
```

où le type `point` est simplement défini par :

```
typedef struct str_point
        { float x, y;
        } point;
```

D.2 Choix de la mesure de discrimination à utiliser

```
#include "decl.h"
#include "decl_define.h"
#include "sef.h"
#include "entropieStar.h"
#include "infoPersoFloue.h"
#include "LeMain.h"

double (*info_Mesure_H_Floue[NOMBRE_DE_MESURES])
              (EltPile*,brique *, brique *, brique *);
double (*info_Mesure_G_Floue[NOMBRE_DE_MESURES])(EltPile*, brique *);

void Initialise_Les_Tableaux_De_Mesures_Floues( void )
{
```

```
   info_Mesure_H_Floue[ENTROPIE_STAR]= Entropie_Star;
   info_Mesure_G_Floue[ENTROPIE_STAR]= Entropie_Sef_Star;

   info_Mesure_H_Floue[MESURE_FLOUE_PERSO]= Fonction_H_Floue;
   info_Mesure_G_Floue[MESURE_FLOUE_PERSO]= Fonction_G_Floue;
}
/* ------------------------------------------------------ */
/*     Calcul de l'information apportee par un sef        */
/* ------------------------------------------------------ */
double  Mesure_Information_Pour_Une_Modalite_Floue
                   (EltPile *listeValeur, brique *unSef)
     /*  p(v).somme(p(ci|v).log2(p(ci|v))) */
{ double informationMesuree= 0.0;
  informationMesuree=
      info_Mesure_G_Floue[MesureFloueChoisie](listeValeur, unSef);
  return( informationMesuree );
}
/* ------------------------------------------------------ */
/*  Calcul de l'information apportee par un ensemble de   */
/*    valeurs associé à une partition floue en 3 sefs     */
/* ------------------------------------------------------ */
double Mesure_Information_Pour_Un_Attribut_Flou(EltPile *listeValeur,
                brique *sefInf, brique *sefSup, brique *sefMid)
     /* Cette fonction renvoie la mesure de l'info apportee par */
     /* une question posee sur une valeur floue d'un attribut */
     /* elle renvoie une valeur a MINIMISER */
     /* pour le moment la seule mesure est l'entropie */
{ double informationMesuree= 0.0;
  informationMesuree=
    info_Mesure_H_Floue[MesureFloueChoisie]
            (listeValeur,sefInf,sefSup,sefMid);
  return( informationMesuree );
}
```

D.3 Exemple de mesure de discrimination

```
/* ------------------------------------------------------ */
/*                  entropieStar.c                        */
/* ------------------------------------------------------ */
#include <math.h>
#include "decl.h"
#include "sef.h"
#include "LeMain.h"
extern int NombreTotalDeLettres;
/* ------------------------------------------------------ */
/*      Degré d'appartenance d'une valeur à un sef        */
/* ------------------------------------------------------ */
double Degre_Appartenance(EltPile *tete, brique *unSef)
{ double ordonnee= 0.0;
  for ( ; (unSef->suiv!=NULL)&&(tete->val >= unSef->p.x)
```

```
        &&(tete->val >= unSef->suiv->p.x);
      unSef = unSef->suiv);
  if ((tete->val <= unSef->p.x) || (unSef->suiv == NULL))
     ordonnee= unSef->p.y;
  else
     ordonnee= Appartenance(Equation_dr(unSef->p,unSef->suiv->p),tete->val);
  return(ordonnee);
}
/* ------------------------------------------------------- */
/*               Probabilité pour un sef                   */
/* ------------------------------------------------------- */
double proba_star(EltPile *tete, brique *unSef)
     /* Renvoie la proba d'un évènement flou */
{ double total= 0.0, ordonnee= 0.0;
  int occurrence= 0;

  if (unSef == NULL)
     return(total);
  for ( ; tete !=NULL; tete = tete->suiv)
     { ordonnee= Degre_Appartenance(tete,unSef);
       if (ordonnee > 0)
          { ordonnee= ordonnee * tete->nbr;
            total += ordonnee;
         }
       else
          { ordonnee= 0.0;
          }
       occurrence += tete->nbr;
    }
  if (occurrence != 0)
     total = total / occurrence;
  return(total);
}
double proba_star_par_classe(EltPile *tete, brique *unSef, char classe)
     /* Renvoie la proba d'un évènement flou relativement à une classe */
{ double total = 0.0, ordonnee = 0.0;
  int occurrence = 0;
  if (unSef == NULL)
     return(total);
  for ( ; (tete !=NULL); tete = tete->suiv)
     { ordonnee= Degre_Appartenance(tete,unSef);
       if (tete->classe == classe)
          { total += ordonnee * tete->nbr;
          }
       if (ordonnee > 0)
          occurrence += tete->nbr;
    }
  if (occurrence != 0)
     total = total / occurrence;
  return(total);
}
```

```
/* ------------------------------------------------------- */
/*           Calcule l'entropie Étoile d'un sef            */
/* ------------------------------------------------------- */
double Entropie_Sef_Star(EltPile *tete, brique *unSef)
     /*  p(v).somme(p(ci|v).log2(p(ci|v))) */
{ register int i;
  double valeurFinale = 0.0, valeurTemp =0.0;
  char classe;
  for (i=0; i<NombreTotalDeLettres; i++)
     { classe = LETTRE_DEPART+i;
       valeurTemp = proba_star_par_classe(tete,unSef, classe);
       if (valeurTemp > 0.0)
          valeurTemp = valeurTemp*log2(valeurTemp);
       else
          valeurTemp = 0.0;
       valeurFinale += valeurTemp;
    }
  valeurTemp = proba_star(tete,unSef);
  valeurFinale = -valeurFinale*valeurTemp;
  return(valeurFinale);
}
double Entropie_Star(EltPile *tete, brique *sefInf, brique *sefSup,
                                    brique *sefMid)
{ double gauche, droite, centre, total;
  gauche= proba_star(tete,sefInf) * Entropie_Sef_Star(tete, sefInf);
  centre= proba_star(tete,sefMid) * Entropie_Sef_Star(tete, sefMid);
  droite= proba_star(tete,sefSup) * Entropie_Sef_Star(tete, sefSup);
  total = gauche + droite + centre;
  return(total);
}
```

Annexe E

Extraits du code de Tanit

E.1 Lancement des processus Salammbô

```
switch ( fork() )
{ case -1 :              /* Erreur de fork */
     perror("Tanit: Problème de fork");
     exit( FIN_ERREUR );
   case 0 :              /* On est dans le fils */
     printf("\tOn est dans le lecteur\n");
     /* Redirection de l'entrée standard : */
     close(LECTURE);
     dup(tube[LECTURE]);
     /* Fermeture des descripteurs inutilisés : */
     close(tube[ECRITURE]);
     close(tube[LECTURE]);
     /* Lecture des résultats : */
     printf("Tanit: Lecture des résultats dans le tube.\n");
     yylex();
     printf("\nTanit: Fin de la lecture dans le tube.\n");
     if (afficheTout)
        Affiche_Tous_Les_Resultats();
     Remplis_Classement();
     Affiche_Classement();
     /* ---------------------------------- */
     exit(FIN_NORMALE);
     break;
 default :
   for (pClass= lesClasses; pClass!=NULL; pClass= pClass->suiv)
    { arguments[3]= (char*)malloc(pClass->taille*sizeof(char));
      strcpy((char*)arguments[3],pClass->enChaine);
      if (nbProcess >= MAXPROCESS)
        { printf("Attente\n");
          pause();
          printf("FIN Attente\n");
       }
      /* Création d'un fils pour la classe arguments[3] */
```

```
      nbProcess++;
      switch ( fork() )
        { case -1 :         /* Erreur de fork */
            perror("Tanit: Problème de fork");
            exit( FIN_ERREUR );
          case 0 :         /* On est dans le fils */
            printf("\tLe fils de Tanit se transforme en Salammbô\n");
            /* Redirection de la sortie standart : */
            close(ECRITURE);
            dup(tube[ECRITURE]);
            /* Fermeture des descripteurs inutilisés : */
            close(tube[ECRITURE]);
            close(tube[LECTURE]);
            /* Recouvrement : */
            if (execv("Salammbo",arguments)==-1)
              { perror("Fils de Tanit: ERREUR de création de processus");
                exit( FIN_ERREUR );
             }
            exit( FIN_NORMALE );
        }
   }
   /* Fermeture des descripteurs inutilisés : */
   close(tube[ECRITURE]);
   close(tube[LECTURE]);
   while(nbProcess >= 1)
     pause();
}
```

E.2 Traitement des résultats

```
void listeDeResultats::MetAjour( ptrClasse* uneClasse, float unDegre )
{ if (classe->Egale(uneClasse))
   { switch (methodeAgregation) {
       case DEMPSTER :
         degre= degre*unDegre;
         break;
        case VOTE :
          degre += unDegre;
          break;
        case EVIDENCE :
          degre += unDegre;
          break;
        default :
          cerr << "Méthode d'agrégation inconnue\n";
      }
    }
  else
      if (suiv != NULL)
         suiv->MetAjour(uneClasse, unDegre);
}
```

```
void listeDeResultats::MetAjourAutre( ptrClasse* uneClasse, float unDegre )
{ if (!(classe->Egale(uneClasse)))
     {
        if (methodeAgregation == DEMPSTER)
           degre= degre*unDegre;
        else
           degre += unDegre;
     }
  if (suiv != NULL)
     suiv->MetAjourAutre(uneClasse, unDegre);
}
void listeDeResultats::MetAjour( char* chaineClasse,
           ptrClasse* laClasse, float unDegre )
{ if (laClasse == NULL)
    { switch (methodeAgregation) {
       case DEMPSTER :
         degre= degre*unDegre;
         break;
       case VOTE :
         degre += unDegre;
         break;
       case EVIDENCE :
         degre += unDegre;
         break;
       default :
         cerr << "Méthode d'agrégation inconnue\n";
      }
      if (suiv != NULL)
        suiv->MetAjour(chaineClasse,laClasse, unDegre);
      return;
     }
  if (laClasse->Egale(chaineClasse))
    { MetAjour(laClasse,unDegre);
      switch (methodeAgregation) {
       case DEMPSTER :
         if (!PlusieursClasses)
           MetAjour(laClasse,0.0);
       break;
       case VOTE :
         MetAjourAutre(laClasse,unDegre/((numeroClasse-1)*1.0));
       break;
       case EVIDENCE :
          MetAjourAutre(laClasse,unDegre);
         break;
        default :
           cerr << "Méthode d'agrégation inconnue\n";
       }
     }
  else
     { switch (methodeAgregation) {
        case DEMPSTER :
```

```
           MetAjourAutre(laClasse,unDegre);
           if (!PlusieursClasses)
              MetAjour(laClasse,0.0);
           break;
         case VOTE :
            MetAjourAutre(laClasse,unDegre/((numeroClasse-1)*1.0));
           break;
         case EVIDENCE :
            MetAjourAutre(laClasse,unDegre);
           break;
         default :
            cerr << "Méthode d'agrégation inconnue\n";
        }
    }
}
```

Annexe F

Résultats d'exécution

Description des résultats présentés

Dans cette annexe, des sorties d'exécution de l'application Salammbô sur les différents types de données sont présentées. Une sortie d'exécution est composée d'une entête :

```
                ----------------------------------
                -            Salammbô            -
                ----------------------------------
Paramètres de base :
        Découpage en 2
        Sans Ratio
        Sans optimisation (que MM)
        Classification floue demandée
        Seuil de continuité = 5
        Seuil d'entropie = 0.00
        Nombre minimum pour découper = 2
        Indice de tolérance pour le flou = 0.00
        Taille de la fenêtre de base = 50000
        Nombre de mauvaises classifications pour reconstruire = 50000
        Mesure choisie = SHANNON
        Mesure Floue choisie = ENTROPIE_STAR
        Nombre de filtrage = 0.05
        La base d'apprentissage est: Bas_Appr_xaa
```

Chaque ligne de cette entête donne la valeur d'un des paramètres d'exécution de Salammbô. En particulier, la mesure de discrimination utilisée est donnée par la ligne **Mesure choisie =**, dans le cas de la construction sans entropie étoile, et par la ligne **Mesure Floue choisie =**. Dans ce dernier cas, les attributs numériques sont discrétisés à l'aide de la méthode présentée dans le chapitre 4. La ligne **Nombre de filtrage =** donne alors le nombre maximum de filtrage appliqués (en pourcentage de la taille de la base d'apprentissage).

Ensuite, l'arbre construit par la Salammbô pour la base d'apprentissage correspondante est donné de façon très sommaire :

```
        ++ ARBRE CONSTRUIT ++
[Racine]: Attribut 3
A 3 $1.10$: [-Inf,3.40](<<3.50)(-Inf, 1.00)(1.90, 1.00)(4.90, 0.00)      {135}
   A 3 $0.23$: [-Inf,2.45](<<3.00)(-Inf, 1.00)(1.90, 1.00)(3.00, 0.00)   {48}
                                                  =>Iris-setosa {45} $0.00$
   A 3 $0.23$: [2.45,Inf](1.90>>)(1.90, 0.00)(3.00, 1.00)(Inf, 1.00)     {48}
                                                  =>Iris-versicolor {3} $0.00$
A 3 $1.10$: [3.40,Inf](3.30>>)(1.90, 0.00)(4.90, 1.00)(Inf, 1.00)        {135}
   A 4 $0.69$: [-Inf,1.75](<<1.80)(-Inf, 1.00)(1.70, 1.00)(1.80, 0.00)   {87}
      A 3 $0.34$: [-Inf,5.35](<<5.60)(-Inf, 1.00)(5.00, 1.00)(5.60, 0.00) {46}
                                                  =>Iris-versicolor {41}
                                                  =>Iris-virginica {3} $0.25$
      A 3 $0.34$: [5.35,Inf](5.10>>)(5.00, 0.00)(5.60, 1.00)(Inf, 1.00)  {46}
                                                  =>Iris-virginica {2} $0.00$
   A 4 $0.69$: [1.75,Inf](1.70>>)(1.70, 0.00)(1.80, 1.00)(Inf, 1.00)     {87}
                                                  =>Iris-virginica {40}
                                                  =>Iris-versicolor {1} $0.11$
```

Dans cette représentation, l'arbre se lit en profondeur d'abord, branche par branche, de la branche la plus à gauche, à celle la plus à droite[1]. Un arc intérieur de l'arbre est donné par une ligne, avec la signification suivante :

- **A 3** est l'attribut choisit pour libeller le nœud de l'arbre.
- **1.10** est la valeur de la mesure de discrimination au niveau de ce nœud.
- **[-Inf,3.40]** donne l'intervalle des valeurs de la base d'apprentissage pour le cas strict.
- **($\ll$ 3.50)** donne la valeur de test pour le cas strict.
- **(-Inf, 1.00)(1.90, 1.00)(4.90, 0.00)** donne le sous-ensemble flou libellant la branche issue du nœud considéré. Chaque couple de point (x, y) correspond à une valeur de l'attribut x associée à son degré d'appartenance y. Les points sont reliés entre eux pour donner la modalité floue. Par exemple, le sous-ensemble flou donné ici est représenté par la modalité **p1** de la Figure 7.3.
- **{135}** donne le nombre d'exemples de la base d'apprentissage au niveau de ce nœud.

Un arc se terminant sur une feuille contient les informations supplémentaires :

- => **Iris-versicolor** donne la classe libellant la feuille. Il peut exister plusieurs classes pour une même feuille, chacune associée avec le pourcentage d'exemples d'apprentissage ayant permis de la construire.

1. L'arbre présenté ici est représenté graphiquement et transcrit en règles de production dans la section 7.3.3 du chapitre 7

– **\$0.0\$** donne la valeur de la mesure de discrimination pour cette feuille.

La dernière partie de la sortie d'exécution présente les résultats obtenus lors de la phase de classification du fichier de test. Les résultats des 4 méthodes d'utilisation de l'arbre sont données.

Approximation de fonctions

Arbres à seuils flous et mesure de Shannon

```
                ----------------------------------
                -            Salammbô            -
                ----------------------------------
Paramètres de base :
        Découpage en 2
        Sans Ratio
        Sans optimisation (que MM)
        Classification floue demandée
        Seuil de continuité = 5
        Seuil d'entropie = 0.00
        Nombre minimum pour découper = 2
        Indice de tolérance pour le flou = 0.00
        Taille de la fenêtre de base = 50000
        Nombre de mauvaises classifications pour reconstruire = 50000
        Mesure choisie = SHANNON    ** Version sans Entropie Star **
        La base d'apprentissage est: BasAppr_xac
[Chargement de la base d'apprentissage] [0"]
[Apprentissage] [0"]
        ++ ARBRE CONSTRUIT ++
[Racine]: Attribut 2
A 2 $0.69$: [-Inf,23.17](<<23.37) {200}
   A 1 $0.55$: [-Inf,76.40](<<76.50) {91}
      A 2 $0.62$: [-Inf,12.46](<<12.61) {70}
         A 2 $0.35$: [-Inf,1.97](<<2.39) {36}
            A 1 $0.64$: [-Inf,27.67](<<27.90) {6}
               A 1 $0.50$: [-Inf,12.55](<<23.73) {5} => 1 {1} => 0 {1} $0.69$
               A 1 $0.50$: [12.55,Inf](5.09>>) {5} => 0 {3} $0.00$
            A 1 $0.64$: [27.67,Inf](26.49>>) {6} => 1 {1} $0.00$
         A 2 $0.35$: [1.97,Inf](1.88>>) {36}
            A 2 $0.24$: [-Inf,9.84](<<9.87) {30} => 0 {20} $0.00$
            A 2 $0.24$: [9.84,Inf](9.77>>) {30}
               A 2 $0.50$: [-Inf,10.82](<<10.93) {10}
                  A 1 $0.64$: [-Inf,63.54](<<69.97) {3} => 1 {2} $0.00$
                  A 1 $0.64$: [63.54,Inf](50.69>>) {3} => 0 {1} $0.00$
               A 2 $0.50$: [10.82,Inf](10.77>>) {10} => 0 {7} $0.00$
      A 2 $0.62$: [12.46,Inf](12.30>>) {70}
         A 1 $0.69$: [-Inf,16.03](<<16.12) {34} => 0 {11} $0.00$
         A 1 $0.69$: [16.03,Inf](15.99>>) {34}
            A 2 $0.52$: [-Inf,15.59](<<15.71) {23}
               A 1 $0.29$: [-Inf,23.41](<<38.26) {12}
```

```
                  A 1 $0.64$: [-Inf,17.97](<<18.46) {3} => 1 {2} $0.00$
                  A 1 $0.64$: [17.97,Inf](17.00>>) {3} => 0 {1} $0.00$
               A 1 $0.29$: [23.41,Inf](18.46>>) {12} => 1 {9} $0.00$
            A 2 $0.52$: [15.59,Inf](15.45>>) {23}
               A 2 $0.66$: [-Inf,15.87](<<16.43) {11} => 0 {2} $0.00$
               A 2 $0.66$: [15.87,Inf](15.75>>) {11}
                  A 1 $0.53$: [-Inf,54.48](<<55.01) {9} => 1 {5} $0.00$
                  A 1 $0.53$: [54.48,Inf](53.80>>) {9}
                     A 1 $0.69$: [-Inf,60.15](<<61.04) {4} => 0 {2} $0.00$
                     A 1 $0.69$: [60.15,Inf](59.27>>) {4} => 1 {2} $0.00$
   A 1 $0.55$: [76.40,Inf](76.07>>) {91} => 0 {21} $0.00$
A 2 $0.69$: [23.17,Inf](23.01>>) {200}
   A 1 $0.60$: [-Inf,84.16](<<84.65) {109}
      A 1 $0.46$: [-Inf,11.01](<<11.27) {87}
         A 1 $0.45$: [-Inf,5.94](<<6.08) {12} => 0 {5} $0.00$
         A 1 $0.45$: [5.94,Inf](5.84>>) {12}
            A 1 $0.60$: [-Inf,6.13](<<6.42) {7} => 1 {1} $0.00$
            A 1 $0.60$: [6.13,Inf](6.08>>) {7}
               A 2 $0.45$: [-Inf,29.83](<<31.20) {6} => 1 {1} $0.00$
               A 2 $0.45$: [29.83,Inf](29.55>>) {6} => 0 {5} $0.00$
      A 1 $0.46$: [11.01,Inf](10.97>>) {87}
         A 1 $0.24$: [-Inf,22.88](<<23.75) {75}
            A 1 $0.56$: [-Inf,22.53](<<22.77) {8}
               A 2 $0.41$: [-Inf,29.48](<<30.52) {7} => 1 {1} => 0 {1} $0.69$
               A 2 $0.41$: [29.48,Inf](29.07>>) {7} => 1 {5} $0.00$
            A 1 $0.56$: [22.53,Inf](20.84>>) {8} => 0 {1} $0.00$
         A 1 $0.24$: [22.88,Inf](22.77>>) {75}
            A 2 $0.18$: [-Inf,48.18](<<48.21) {67} => 1 {49} $0.00$
            A 2 $0.18$: [48.18,Inf](48.12>>) {67}
               A 2 $0.45$: [-Inf,55.10](<<55.14) {18}
                  A 2 $0.68$: [-Inf,54.48](<<54.62) {7}
                     A 2 $0.50$: [-Inf,49.18](<<53.04) {5} => 0 {1} $0.00$
                     A 2 $0.50$: [49.18,Inf](48.21>>) {5} => 1 {4} $0.00$
                  A 2 $0.68$: [54.48,Inf](54.14>>) {7} => 0 {2} $0.00$
               A 2 $0.45$: [55.10,Inf](55.07>>) {18} => 1 {11} $0.00$
   A 1 $0.60$: [84.16,Inf](82.25>>) {109}
      A 2 $0.59$: [-Inf,52.46](<<52.69) {22} => 0 {14} $0.00$
      A 2 $0.59$: [52.46,Inf](52.06>>) {22}
         A 1 $0.56$: [-Inf,94.83](<<95.38) {8}
            A 1 $0.69$: [-Inf,93.38](<<93.99) {4} => 1 {2} $0.00$
            A 1 $0.69$: [93.38,Inf](92.76>>) {4} => 0 {2} $0.00$
         A 1 $0.56$: [94.83,Inf](94.29>>) {8} => 1 {4} $0.00$
Nombre de branches: 32 Profondeur: Maximum = 8  Moyenne = 5.81
[Classification] ++ Fichier 1: BasTest_xac       [0"]
Stricte       :[0"] [0.015"]
        --> 1480 bons pour 1800 lus. soit 82.22% (E.R.: 17.78%)
        classe  0: Bons=  688 sur  900 soit 76.44
        classe  1: Bons=  792 sur  900 soit 88.00
de Zadeh        :[0"] [0.070"]
        --> 1505 bons pour 1800 lus. soit 83.61% (E.R.: 16.39%)
        classe  0: Bons=  707 sur  900 soit 78.56
```

```
        classe  1: Bons=  798 sur  900 soit 88.67
de Lukasiewicz  :[0"] [0.066"]
        --> 1512 bons pour 1800 lus. soit 84.00% (E.R.: 16.00%)
        classe  0: Bons=  704 sur  900 soit 78.22
        classe  1: Bons=  808 sur  900 soit 89.78
Probabiliste    :[0"] [0.062"]
        --> 1506 bons pour 1800 lus. soit 83.67% (E.R.: 16.33%)
        classe  0: Bons=  707 sur  900 soit 78.56
        classe  1: Bons=  799 sur  900 soit 88.78
                ----------------------------------
```

Arbres construits avec l'entropie étoile

```
                ----------------------------------
                -            Salammbô            -
                ----------------------------------
Paramètres de base :
        Découpage en 2
        Sans Ratio
        Sans optimisation (que MM)
        Classification floue demandée
        Seuil de continuité = 5
        Seuil d'entropie = 0.00
        Nombre minimum pour découper = 2
        Indice de tolérance pour le flou = 0.00
        Taille de la fenêtre de base = 50000
        Nombre de mauvaises classifications pour reconstruire = 50000
        Mesure choisie = SHANNON
        Mesure Floue choisie = ENTROPIE_STAR
        Nombre de filtrage = 0.05
        La base d'apprentissage est: BasAppr_xac
[Chargement de la base d'apprentissage] [0"]
[Apprentissage] [0"]
        ++ ARBRE CONSTRUIT ++
[Racine]: Attribut 2
A 2 $0.69$: [-Inf,17.50](<<17.97)(-Inf, 1.00)(12.30, 1.00)(23.37, 0.00)  {200}
  A 2 $0.54$: [-Inf,12.80](<<12.98)(-Inf, 1.00)(12.30, 1.00)(13.50, 0.00) {73}
              => 0 {42} => 1 {5} $0.34$
  A 2 $0.54$: [12.80,Inf](12.61>>)(12.30, 0.00)(13.50, 1.00)(Inf, 1.00)   {73}
    A 1 $0.69$: [-Inf,69.40](<<72.73)(-Inf, 1.00)(66.08, 1.00)(76.07,0.0){26}
      A 1 $0.69$: [-Inf,28.38](<<38.26)(-Inf,1.00)(16.12,1.00)(57.12,0.0){18}
                  => 0 {7} => 1 {2} $0.53$
      A 1 $0.69$: [28.38,Inf](18.50>>)(16.12, 0.00)(57.12, 1.00)(Inf, 1.00){18}
                  => 1 {8} => 0 {1} $0.35$
    A 1 $0.69$: [69.40,Inf](66.08>>)(66.08, 0.00)(76.07, 1.00)(Inf, 1.00) {26}
                => 0 {6} => 1 {2} $0.56$
A 2 $0.69$: [17.50,Inf](17.04>>)(12.30, 0.00)(23.37, 1.00)(Inf, 1.00)    {200}
  A 1 $0.65$: [-Inf,80.68](<<81.49)(-Inf, 1.00)(77.61, 1.00)(84.65, 0.00) {127}
    A 1 $0.57$: [-Inf,12.98](<<14.19)(-Inf, 1.00)(11.77,1.00)(14.19,0.00){100}
                => 0 {15} => 1 {3} $0.45$
    A 1 $0.57$: [12.98,Inf](11.77>>)(11.77, 0.00)(14.19, 1.00)(Inf, 1.00) {100}
```

```
      A 1 $0.39$: [-Inf,77.89](<<78.17)(-Inf,1.00)(77.61 1.00)(78.17,0.00){82}
                  => 1 {69} => 0 {8} $0.33$
      A 1 $0.39$: [77.89,Inf](77.61>>)(77.61, 0.00)(78.17,1.00)(Inf,1.00){82}
                  => 0 {3} => 1 {2} $0.67$
  A 1 $0.65$: [80.68,Inf](79.87>>)(77.61, 0.00)(84.65, 1.00)(Inf, 1.00) {127}
    A 2 $0.64$: [-Inf,53.73](<<54.14)(-Inf, 1.00)(52.69, 1.00)(54.14, 0.00){27}
                => 0 {17} => 1 {3} $0.42$
    A 2 $0.64$: [53.73,Inf](53.33>>)(52.69, 0.00)(54.14, 1.00)(Inf, 1.00) {27}
                => 1 {6} => 0 {1} $0.41$
Nombre de branches: 9 Profondeur: Maximum = 4  Moyenne = 3.33
[Classification] ++ Fichier 1: BasTest_xac      [1"]
Stricte :[1"] [0.015"]
        --> 1568 bons pour 1800 lus. soit 87.11% (E.R.: 12.89%)
        classe  0: Bons=  810 sur  900 soit 90.00
        classe  1: Bons=  758 sur  900 soit 84.22
de Zadeh        :[1"] [0.071"]
        --> 1540 bons pour 1800 lus. soit 85.56% (E.R.: 14.44%)
        classe  0: Bons=  777 sur  900 soit 86.33
        classe  1: Bons=  763 sur  900 soit 84.78
de Lukasiewicz  :[1"] [0.064"]
        --> 1537 bons pour 1800 lus. soit 85.39% (E.R.: 14.61%)
        classe  0: Bons=  762 sur  900 soit 84.67
        classe  1: Bons=  775 sur  900 soit 86.11
Probabiliste    :[1"] [0.064"]
        --> 1541 bons pour 1800 lus. soit 85.61% (E.R.: 14.39%)
        classe  0: Bons=  779 sur  900 soit 86.56
        classe  1: Bons=  762 sur  900 soit 84.67
                ----------------------------------
```

Données des iris

Arbres à seuils flous et mesure de Shannon

```
                ----------------------------------
                -           Salammbô             -
                ----------------------------------
Paramètres de base :
        Découpage en 2
        Sans Ratio
        Sans optimisation (que MM)
        Classification floue demandée
        Seuil de continuité = 5
        Seuil d'entropie = 0.00
        Nombre minimum pour découper = 2
        Indice de tolérance pour le flou = 0.00
        Taille de la fenêtre de base = 50000
        Nombre de mauvaises classifications pour reconstruire = 50000
        Mesure choisie = SHANNON     ** Version sans Entropie Star **
        La base d'apprentissage est: Bas_Appr_xaa
[Chargement de la base d'apprentissage] [0"]
```

```
[Apprentissage] [0"]
        ++ ARBRE CONSTRUIT ++
[Racine]: Attribut 3
A 3 $1.10$: [-Inf,2.27](<<3.00) {135} =>Iris-setosa {45} $0.00$
A 3 $1.10$: [2.27,Inf](1.90>>) {135}
   A 3 $0.69$: [-Inf,4.75](<<4.80) {90}
      A 4 $0.11$: [-Inf,1.70](<<1.70) {41} =>Iris-versicolor {40} $0.00$
      A 4 $0.11$: [1.70,Inf](1.60>>) {41} =>Iris-virginica {1} $0.00$
   A 3 $0.69$: [4.75,Inf](4.70>>) {90}
      A 4 $0.33$: [-Inf,1.72](<<1.80) {49}
         A 3 $0.69$: [-Inf,5.47](<<5.60) {8}
            A 2 $0.64$: [-Inf,2.25](<<2.50) {6} =>Iris-virginica {1} $0.00$
            A 2 $0.64$: [2.25,Inf](2.20>>) {6}
               A 3 $0.50$: [-Inf,5.06](<<5.10) {5} =>Iris-versicolor {3} $0.00$
               A 3 $0.50$: [5.06,Inf](5.00>>) {5} =>Iris-virginica {1}
                                                   =>Iris-versicolor {1} $0.69$
         A 3 $0.69$: [5.47,Inf](5.10>>) {8} =>Iris-virginica {2} $0.00$
      A 4 $0.33$: [1.72,Inf](1.70>>) {49}
         A 3 $0.11$: [-Inf,4.81](<<4.90) {41}
            A 1 $0.64$: [-Inf,5.93](<<6.00) {3} =>Iris-versicolor {1} $0.00$
            A 1 $0.64$: [5.93,Inf](5.90>>) {3} =>Iris-virginica {2} $0.00$
         A 3 $0.11$: [4.81,Inf](4.80>>) {41} =>Iris-virginica {38} $0.00$
Nombre de branches: 10 Profondeur: Maximum = 6  Moyenne = 4.20
[Classification] ++ Fichier 1: Bas_Test_xaa      [0"]
Stricte       :[0"] [0.000"]
        --> 15 bons pour 15 lus. soit 100.00% (E.R.: 0.00%)
        classe Iris-setosa: Bons=    5 sur    5 soit 100.00
        classe Iris-versicolor: Bons=    5 sur    5 soit 100.00
        classe Iris-virginica: Bons=    5 sur    5 soit 100.00
de Zadeh        :[0"] [0.000"]
        --> 15 bons pour 15 lus. soit 100.00% (E.R.: 0.00%)
        classe Iris-setosa: Bons=    5 sur    5 soit 100.00
        classe Iris-versicolor: Bons=    5 sur    5 soit 100.00
        classe Iris-virginica: Bons=    5 sur    5 soit 100.00
de Lukasiewicz  :[0"] [0.000"]
        --> 15 bons pour 15 lus. soit 100.00% (E.R.: 0.00%)
        classe Iris-setosa: Bons=    5 sur    5 soit 100.00
        classe Iris-versicolor: Bons=    5 sur    5 soit 100.00
        classe Iris-virginica: Bons=    5 sur    5 soit 100.00
Probabiliste    :[0"] [0.000"]
        --> 15 bons pour 15 lus. soit 100.00% (E.R.: 0.00%)
        classe Iris-setosa: Bons=    5 sur    5 soit 100.00
        classe Iris-versicolor: Bons=    5 sur    5 soit 100.00
        classe Iris-virginica: Bons=    5 sur    5 soit 100.00
                ----------------------------------
```

Arbres à seuils flous et mesure de discrimination

```
                ----------------------------------
                -           Salammbô             -
                ----------------------------------
```

```
Paramètres de base :
        Découpage en 2
        Sans Ratio
        Sans optimisation (que MM)
        Classification floue demandée
        Seuil de continuité = 5
        Seuil d'entropie = 0.00
        Nombre minimum pour découper = 2
        Indice de tolérance pour le flou = 0.00
        Taille de la fenêtre de base = 50000
        Nombre de mauvaises classifications pour reconstruire = 50000
        Mesure choisie = PERSO       ** Version sans Entropie Star **
        La base d'apprentissage est: Bas_Appr_xaa
[Chargement de la base d'apprentissage] [0"]
[Apprentissage] [0"]
        ++ ARBRE CONSTRUIT ++
[Racine]: Attribut 4
A 4 $6075.00$: [-Inf,1.77](<<1.80) {135}
   A 3 $0.00$: [-Inf,2.43](<<3.00) {94} =>Iris-setosa {45} $0.00$
   A 3 $0.00$: [2.43,Inf](1.90>>) {94}
      A 2 $0.00$: [-Inf,2.64](<<2.70) {49}
         A 2 $0.00$: [-Inf,2.45](<<2.50) {18}
            A 2 $0.01$: [-Inf,2.24](<<2.30) {9}
               A 3 $0.07$: [-Inf,4.88](<<5.00) {4} =>Iris-versicolor {3} $0.06$
               A 3 $0.07$: [4.88,Inf](4.50>>) {4} =>Iris-virginica {1} $0.50$
            A 2 $0.01$: [2.24,Inf](2.20>>) {9} =>Iris-versicolor {5} $0.02$
         A 2 $0.00$: [2.45,Inf](2.40>>) {18}
            A 2 $0.01$: [-Inf,2.56](<<2.60) {9}
               A 4 $0.04$: [-Inf,1.66](<<1.70) {5} =>Iris-versicolor {4} $0.03$
               A 4 $0.04$: [1.66,Inf](1.50>>) {5} =>Iris-virginica {1} $0.50$
            A 2 $0.01$: [2.56,Inf](2.50>>) {9}
               A 1 $0.07$: [-Inf,6.02](<<6.10) {4} =>Iris-versicolor {3} $0.06$
               A 1 $0.07$: [6.02,Inf](5.80>>) {4} =>Iris-virginica {1} $0.50$
      A 2 $0.00$: [2.64,Inf](2.60>>) {49}
         A 2 $0.00$: [-Inf,2.84](<<2.90) {31}
            A 2 $0.01$: [-Inf,2.75](<<2.80) {11} =>Iris-versicolor {5} $0.02$
            A 2 $0.01$: [2.75,Inf](2.70>>) {11}
               A 3 $0.02$: [-Inf,5.05](<<5.10) {6} =>Iris-versicolor {5} $0.02$
               A 3 $0.02$: [5.05,Inf](4.80>>) {6} =>Iris-virginica {1} $0.50$
         A 2 $0.00$: [2.84,Inf](2.80>>) {31}
            A 2 $0.00$: [-Inf,2.93](<<3.00) {20} =>Iris-versicolor {7} $0.01$
            A 2 $0.00$: [2.93,Inf](2.90>>) {20}
               A 2 $0.00$: [-Inf,3.07](<<3.10) {13}
                  A 1 $0.01$: [-Inf,7.14](<<7.20){9} =>Iris-versicolor{8}$0.01$
                  A 1 $0.01$: [7.14,Inf](6.70>>){9} =>Iris-virginica{1}$0.50$
               A 2 $0.00$: [3.07,Inf](3.00>>) {13} =>Iris-versicolor {4} $0.03$
A 4 $6075.00$: [1.77,Inf](1.70>>) {135}
   A 4 $0.00$: [-Inf,1.83](<<1.90) {41}
      A 3 $0.01$: [-Inf,4.83](<<4.90) {11}
         A 2 $0.17$: [-Inf,3.13](<<3.20) {3} =>Iris-virginica {2} $0.12$
         A 2 $0.17$: [3.13,Inf](3.00>>) {3} =>Iris-versicolor {1} $0.50$
```

```
      A 3 $0.01$: [4.83,Inf](4.80>>) {11} =>Iris-virginica {8} $0.01$
   A 4 $0.00$: [1.83,Inf](1.80>>) {41} =>Iris-virginica {30} $0.00$
Nombre de branches: 19 Profondeur: Maximum = 7  Moyenne = 5.16
[Classification] ++ Fichier 1: Bas_Test_xaa      [0"]
Stricte       :[0"] [0.000"]
        --> 15 bons pour 15 lus. soit 100.00% (E.R.: 0.00%)
        classe Iris-setosa: Bons=    5 sur    5 soit 100.00
        classe Iris-versicolor: Bons=    5 sur    5 soit 100.00
        classe Iris-virginica: Bons=    5 sur    5 soit 100.00
de Zadeh        :[0"] [0.000"]
        --> 15 bons pour 15 lus. soit 100.00% (E.R.: 0.00%)
        classe Iris-setosa: Bons=    5 sur    5 soit 100.00
        classe Iris-versicolor: Bons=    5 sur    5 soit 100.00
        classe Iris-virginica: Bons=    5 sur    5 soit 100.00
de Lukasiewicz  :[0"] [0.000"]
        --> 15 bons pour 15 lus. soit 100.00% (E.R.: 0.00%)
        classe Iris-setosa: Bons=    5 sur    5 soit 100.00
        classe Iris-versicolor: Bons=    5 sur    5 soit 100.00
        classe Iris-virginica: Bons=    5 sur    5 soit 100.00
Probabiliste    :[0"] [0.000"]
        --> 15 bons pour 15 lus. soit 100.00% (E.R.: 0.00%)
        classe Iris-setosa: Bons=    5 sur    5 soit 100.00
        classe Iris-versicolor: Bons=    5 sur    5 soit 100.00
        classe Iris-virginica: Bons=    5 sur    5 soit 100.00
                ----------------------------------
```

Arbres construits avec l'entropie étoile

```
                ----------------------------------
                -            Salammbô            -
                ----------------------------------
Paramètres de base :
        Découpage en 2
        Sans Ratio
        Sans optimisation (que MM)
        Classification floue demandée
        Seuil de continuité = 5
        Seuil d'entropie = 0.00
        Nombre minimum pour découper = 2
        Indice de tolérance pour le flou = 0.00
        Taille de la fenêtre de base = 50000
        Nombre de mauvaises classifications pour reconstruire = 50000
        Mesure choisie = SHANNON
        Mesure Floue choisie = ENTROPIE_STAR
        Nombre de filtrage = 0.05
        La base d'apprentissage est: Bas_Appr_xaa
[Chargement de la base d'apprentissage] [0"]
[Apprentissage] [0"]
        ++ ARBRE CONSTRUIT ++
[Racine]: Attribut 3
A 3 $1.10$: [-Inf,3.40](<<3.50)(-Inf, 1.00)(1.90, 1.00)(4.90, 0.00)      {135}
```

```
   A 3 $0.23$: [-Inf,2.45](<<3.00)(-Inf, 1.00)(1.90, 1.00)(3.00, 0.00)   {48}
                                                    =>Iris-setosa {45} $0.00$
   A 3 $0.23$: [2.45,Inf](1.90>>)(1.90, 0.00)(3.00, 1.00)(Inf, 1.00)     {48}
                                                    =>Iris-versicolor {3} $0.00$
A 3 $1.10$: [3.40,Inf](3.30>>)(1.90, 0.00)(4.90, 1.00)(Inf, 1.00)        {135}
   A 4 $0.69$: [-Inf,1.75](<<1.80)(-Inf, 1.00)(1.70, 1.00)(1.80, 0.00)   {87}
      A 3 $0.34$: [-Inf,5.35](<<5.60)(-Inf, 1.00)(5.00, 1.00)(5.60, 0.00) {46}
                                                    =>Iris-versicolor {41}
                                                    =>Iris-virginica {3} $0.25$
      A 3 $0.34$: [5.35,Inf](5.10>>)(5.00, 0.00)(5.60, 1.00)(Inf, 1.00)  {46}
                                                    =>Iris-virginica {2} $0.00$
   A 4 $0.69$: [1.75,Inf](1.70>>)(1.70, 0.00)(1.80, 1.00)(Inf, 1.00)     {87}
                                                    =>Iris-virginica {40}
                                                    =>Iris-versicolor {1} $0.11$
Nombre de branches: 5 Profondeur: Maximum = 3  Moyenne = 2.40
[Classification] ++ Fichier 1: Bas_Test_xaa      [0"]
Stricte :[0"] [0.000"]
        --> 15 bons pour 15 lus. soit 100.00% (E.R.: 0.00%)
        classe Iris-setosa: Bons=    5 sur    5 soit 100.00
        classe Iris-versicolor: Bons=    5 sur    5 soit 100.00
        classe Iris-virginica: Bons=    5 sur    5 soit 100.00
de Zadeh        :[0"] [0.000"]
        --> 15 bons pour 15 lus. soit 100.00% (E.R.: 0.00%)
        classe Iris-setosa: Bons=    5 sur    5 soit 100.00
        classe Iris-versicolor: Bons=    5 sur    5 soit 100.00
        classe Iris-virginica: Bons=    5 sur    5 soit 100.00
de Lukasiewicz  :[0"] [0.000"]
        --> 15 bons pour 15 lus. soit 100.00% (E.R.: 0.00%)
        classe Iris-setosa: Bons=    5 sur    5 soit 100.00
        classe Iris-versicolor: Bons=    5 sur    5 soit 100.00
        classe Iris-virginica: Bons=    5 sur    5 soit 100.00
Probabiliste    :[0"] [0.000"]
        --> 15 bons pour 15 lus. soit 100.00% (E.R.: 0.00%)
        classe Iris-setosa: Bons=    5 sur    5 soit 100.00
        classe Iris-versicolor: Bons=    5 sur    5 soit 100.00
        classe Iris-virginica: Bons=    5 sur    5 soit 100.00
                ----------------------------------
```

Données des formes d'ondes

Arbres à seuils flous et mesure de Shannon

```
                ----------------------------------
                -            Salammbô            -
                ----------------------------------
Paramètres de base :
        Découpage en 2
        Sans Ratio
        Sans optimisation (que MM)
        Classification floue demandée
```

```
        Seuil de continuité = 5
        Seuil d'entropie = 0.00
        Nombre minimum pour découper = 100
        Indice de tolérance pour le flou = 0.00
        Taille de la fenêtre de base = 50000
        Nombre de mauvaises classifications pour reconstruire = 50000
        Mesure choisie = SHANNON  ** Version sans Entropie Star **
        La base d'apprentissage est: BasAppr_xag
[Chargement de la base d'apprentissage] [18"]
[Apprentissage] [13"]
        ++ ARBRE CONSTRUIT ++ (Extrait)
[Racine]: Attribut 15
A 15 $1.10$: [-Inf,2.72](<<2.72) {4500}
   A 11 $0.96$: [-Inf,3.16](<<3.17) {2424}
      A 9 $0.75$: [-Inf,2.63](<<2.64) {1049}
         A 13 $0.48$: [-Inf,3.46](<<3.46) {299}
            A 15 $0.33$: [-Inf,1.40](<<1.41) {274}
               A 18 $0.52$: [-Inf,-0.06](<<-0.05){132}=> 1 {19} => 0 {19}$0.69$
               A 18 $0.52$: [-0.06,Inf](-0.06>>){132} => 0 {85} => 1 {9} $0.32$
            A 15 $0.33$: [1.40,Inf](1.39>>) {274} => 0 {142} $0.00$
         A 13 $0.48$: [3.46,Inf](3.45>>) {299} => 2{12} => 0{12 => 1{1} $0.83$
      A 9 $0.75$: [2.63,Inf](2.63>>) {1049}
        ...........
 A 15 $1.10$: [2.72,Inf](2.71>>) {4500}
   A 11 $0.83$: [-Inf,3.55](<<3.55) {2076}
      A 5 $0.72$: [-Inf,1.39](<<1.39) {1351}
        ...........
Nombre de branches: 78 Profondeur: Maximum = 9  Moyenne = 6.85
[Classification] ++ Fichier 1: BasTest_xag        [1"]
Stricte        :[1"] [0.012"]
        --> 372 bons pour 500 lus. soit 74.40% (E.R.: 25.60%)
        classe  0: Bons=  111 sur  166 soit 66.87
        classe  1: Bons=  125 sur  165 soit 75.76
        classe  2: Bons=  136 sur  169 soit 80.47
de Zadeh        :[1"] [0.033"]
        --> 374 bons pour 500 lus. soit 74.80% (E.R.: 25.20%)
        classe  0: Bons=  114 sur  166 soit 68.67
        classe  1: Bons=  124 sur  165 soit 75.15
        classe  2: Bons=  136 sur  169 soit 80.47
de Lukasiewicz  :[1"] [0.027"]
        --> 374 bons pour 500 lus. soit 74.80% (E.R.: 25.20%)
        classe  0: Bons=  115 sur  166 soit 69.28
        classe  1: Bons=  123 sur  165 soit 74.55
        classe  2: Bons=  136 sur  169 soit 80.47
Probabiliste    :[1"] [0.026"]
        --> 374 bons pour 500 lus. soit 74.80% (E.R.: 25.20%)
        classe  0: Bons=  114 sur  166 soit 68.67
        classe  1: Bons=  124 sur  165 soit 75.15
        classe  2: Bons=  136 sur  169 soit 80.47
                ----------------------------------
```

Arbres construits avec l'entropie étoile

```
                ---------------------------------
                -           Salammbô            -
                ---------------------------------
Paramètres de base :
        Découpage en 2
        Sans Ratio
        Sans optimisation (que MM)
        Classification floue demandée
        Seuil de continuité = 5
        Seuil d'entropie = 0.00
        Nombre minimum pour découper = 100
        Indice de tolérance pour le flou = 0.00
        Taille de la fenêtre de base = 50000
        Nombre de mauvaises classifications pour reconstruire = 50000
        Mesure choisie = SHANNON
        Mesure Floue choisie = ENTROPIE_STAR
        Nombre de filtrage = 0.01
        La base d'apprentissage est: BasAppr_xag
[Chargement de la base d'apprentissage] [17"]
[Apprentissage] [141"]
        ++ ARBRE CONSTRUIT ++  (Extrait)
[Racine]: Attribut 7
A 7 $1.10$: [-Inf,2.95](<<2.95)(-Inf, 1.00)(-0.75, 1.00)(6.65, 0.00) {4500}
  A 8 $0.97$: [-Inf,1.65](<<1.65)(-Inf, 1.00)(-0.74, 1.00)(4.03, 0.00) {2542}
    A 10 $0.71$: [-Inf,2.12](<<2.13)(-Inf, 1.00)(0.33, 1.00)(3.93, 0.00) {1229}
          ............
    A 10 $0.71$: [2.12,Inf](2.12>>)(0.33, 0.00)(3.93, 1.00)(Inf, 1.00) {1229}
      ............
 A 7 $1.10$: [2.95,Inf](2.94>>)(-0.75, 0.00)(6.65, 1.00)(Inf, 1.00) {4500}
   A 10 $0.84$: [-Inf,3.25](<<3.25)(-Inf, 1.00)(2.02, 1.00)(4.47, 0.00) {1958}
      A 11 $0.73$: [-Inf,2.49](<<2.49)(-Inf, 1.00)(1.18, 1.00)(3.79, 0.00){954}
        .............
Nombre de branches: 68 Profondeur: Maximum = 7  Moyenne = 6.21
[Classification] ++ Fichier 1: BasTest_xag        [1"]
Stricte       :[1"] [0.015"]
        --> 371 bons pour 500 lus. soit 74.20% (E.R.: 25.80%)
        classe  0: Bons=  111 sur  166 soit 66.87
        classe  1: Bons=  127 sur  165 soit 76.97
        classe  2: Bons=  133 sur  169 soit 78.70
de Zadeh        :[1"] [0.197"]
        --> 393 bons pour 500 lus. soit 78.60% (E.R.: 21.40%)
        classe  0: Bons=  131 sur  166 soit 78.92
        classe  1: Bons=  128 sur  165 soit 77.58
        classe  2: Bons=  134 sur  169 soit 79.29
Probabiliste    :[1"] [0.185"]
        --> 396 bons pour 500 lus. soit 79.20% (E.R.: 20.80%)
        classe  0: Bons=  124 sur  166 soit 74.70
        classe  1: Bons=  143 sur  165 soit 86.67
        classe  2: Bons=  129 sur  169 soit 76.33
```

```
                ----------------------------------
```

Données électriques

Arbres à seuils flous et mesure de Shannon

```
                ----------------------------------
                -            Salammbô            -
                ----------------------------------
Paramètres de base :
        Découpage en 2
        Sans Ratio
        Sans optimisation (que MM)
        Classification floue demandée
        Seuil de continuité = 5
        Seuil d'entropie = 0.06
        Nombre minimum pour découper = 2
        Indice de tolérance pour le flou = 0.00
        Taille de la fenêtre de base = 50000
        Nombre de mauvaises classifications pour reconstruire = 50000
        Mesure choisie = SHANNON  ** Version sans Entropie Star **
        La base d'apprentissage est: learn35.dat
[Chargement de la base d'apprentissage] [2"]
[Apprentissage] [2"]
        ++ ARBRE CONSTRUIT ++  (Extrait)
[Racine]: Attribut 3
A 3 $0.69$: [-Inf,1031.47](<<1032.00) {1000}
   A 3 $0.30$: [-Inf,953.76](<<954.00) {474}
      A 5 $0.13$: [-Inf,1670.22](<<1671.00) {360} => 1 {329} $0.00$
      A 5 $0.13$: [1670.22,Inf](1662.00>>) {360}
      ........
   A 3 $0.30$: [953.76,Inf](953.00>>) {474}
      A 2 $0.60$: [-Inf,-231.04](<<-219.90) {114}
         A 1 $0.48$: [-Inf,2900.78](<<2907.00) {27}
            A 7 $0.67$: [-Inf,428.08](<<457.00) {13}
               A 3 $0.50$: [-Inf,1013.20](<<1018.00) {5} => 1 {4} $0.00$
               A 3 $0.50$: [1013.20,Inf](994.00>>) {5} => 0 {1} $0.00$
            A 7 $0.67$: [428.08,Inf](410.00>>) {13}
               A 3 $0.38$: [-Inf,955.50](<<959.00) {8} => 1 {1} $0.00$
               A 3 $0.38$: [955.50,Inf](955.00>>) {8} => 0 {7} $0.00$
         A 1 $0.48$: [2900.78,Inf](2895.00>>) {27} => 0 {14} $0.00$
      A 2 $0.60$: [-231.04,Inf](-234.50>>) {114}
      ........
A 3 $0.69$: [1031.47,Inf](1031.00>>) {1000}
   A 1 $0.37$: [-Inf,1159.11](<<1160.00) {526}
      A 2 $0.66$: [-Inf,-102.67](<<-95.10) {56}
      ........
      A 2 $0.66$: [-102.67,Inf](-105.70>>) {56}
      ........
   A 1 $0.37$: [1159.11,Inf](1159.00>>) {526}
```

```
      A 3 $0.23$: [-Inf,1109.24](<<1110.00) {470}
      ........
      A 3 $0.23$: [1109.24,Inf](1109.00>>) {470}
      ........
Nombre de branches: 42 Profondeur: Maximum = 10  Moyenne = 6.17
[Classification] ++ Fichier 1: test35.dat       [2"]
Stricte        :[2"] [0.036"]
        --> 1829 bons pour 2000 lus. soit 91.45% (E.R.: 8.55%)
        classe  0: Bons=  919 sur 1025 soit 89.66
        classe  1: Bons=  910 sur  975 soit 93.33
de Zadeh        :[2"] [0.097"]
        --> 1827 bons pour 2000 lus. soit 91.35% (E.R.: 8.65%)
        classe  0: Bons=  917 sur 1025 soit 89.46
        classe  1: Bons=  910 sur  975 soit 93.33
de Lukasiewicz  :[2"] [0.094"]
        --> 1835 bons pour 2000 lus. soit 91.75% (E.R.: 8.25%)
        classe  0: Bons=  925 sur 1025 soit 90.24
        classe  1: Bons=  910 sur  975 soit 93.33
Probabiliste    :[2"] [0.211"]
        --> 1827 bons pour 2000 lus. soit 91.35% (E.R.: 8.65%)
        classe  0: Bons=  917 sur 1025 soit 89.46
        classe  1: Bons=  910 sur  975 soit 93.33
                ----------------------------------
```

Arbres construits avec l'entropie étoile

```
                ----------------------------------
                -           Salammbô             -
                ----------------------------------
Paramètres de base :
        Découpage en 2
        Sans Ratio
        Sans optimisation (que MM)
        Classification floue demandée
        Seuil de continuité = 5
        Seuil d'entropie = 0.06
        Nombre minimum pour découper = 2
        Indice de tolérance pour le flou = 0.00
        Taille de la fenêtre de base = 50000
        Nombre de mauvaises classifications pour reconstruire = 50000
        Mesure choisie = SHANNON
        Mesure Floue choisie = ENTROPIE_STAR
        Nombre de filtrage = 0.00
        La base d'apprentissage est: learn35.dat
[Chargement de la base d'apprentissage] [1"]
[Apprentissage] [6"]
        ++ ARBRE CONSTRUIT ++
[Racine]: Attribut 3
A 3 $0.69$: [-Inf,1029.5](<<1030)(-Inf, 1)(968, 1)(1092, 0) {1000}
  A 3 $0.30$: [-Inf,968.5](<<969)(-Inf, 1)(968, 1)(969, 0)   {469}
    A 5 $0.16$: [-Inf,1793](<<1798)(-Inf, 1)(1788, 1)(1798, 0) {379}
```

```
    => 1 {359} => 0 {5} $07$
  A 5 $0.16$: [1793,Inf](1788>>)(1788, 0)(1798, 1)(Inf, 1){379}
   A 6 $0.67$: [-Inf,15](<<22.1)(-Inf, 1)(7.90, 1)(22.1, 0){15}
     => 0 {8} $0$
   A 6 $0.67$: [15,Inf](7.90>>)(7.90, 0)(22.1, 1)(Inf, 1){15}
     => 1 {6} => 0 {1} $0.41$
 A 3 $0.30$: [968.5,Inf](968>>)(968, 0)(969, 1)(Inf, 1){469}
  A 2 $0.61$: [-Inf,-107.15](<<-102.1)(-Inf, 1)(-238.80, 1)(18.7, 0){90}
   A 1 $0.66$: [-Inf,2482.5](<<2907)(-Inf, 1)(2028, 1)(2955, 0) {32}
    A 7 $0.65$: [-Inf,447](<<484)(-Inf, 1)(410, 1)(484, 0){17}
      => 1 {11} => 0 {1} $0.29$
    A 7 $0.65$: [447,Inf](410>>)(410, 0)(484, 1)(Inf, 1){17}
      => 0 {5} $0$
   A 1 $0.66$: [2482.5,Inf](2058>>)(2028, 0)(2955, 1)(Inf, 1){32}
     => 0 {14} => 1 {1} $0.24$
  A 2 $0.61$: [-107.15,Inf](-112.20>>)(-238.80, 0)(18.7, 1)(Inf, 1){90}
   A 1 $0.37$: [-Inf,3478.5](<<3876)(-Inf, 1)(3081, 1)(4068, 0){58}
     => 1 {47} => 0 {1} $0.1$
   A 1 $0.37$: [3478.5,Inf](3081>>)(3081, 0)(4068, 1)(Inf, 1)   {58}
    A 16 $0.67$: [-Inf,416.30](<<416.60)(-Inf, 1)(416, 1)(416.60, 0){10}
     A 1 $0.69$: [-Inf,4058](<<4068)(-Inf, 1)(4048, 1)(4068, 0){8}
       => 1 {4} => 0 {1} $0.5$
     A 1 $0.69$: [4058,Inf](4048>>)(4048, 0)(4068, 1)(Inf, 1){8}
       => 0 {3} $0$
    A 16 $0.67$: [416.30,Inf](416>>)(416, 0)(416.60, 1)(Inf, 1){10}
       => 0 {2} $0$
A 3 $0.69$: [1029.5,Inf](1029>>)(968, 0)(1092, 1)(Inf, 1) {1000}
 A 3 $0.38$: [-Inf,1134.5](<<1135)(-Inf, 1)(1078, 1)(1191, 0){531}
  A 1 $0.60$: [-Inf,2679](<<3090)(-Inf, 1)(1089, 1)(4128, 0){196}
   A 4 $0.69$: [-Inf,215](<<24.30)(-Inf, 1)(-66.60, 1)(104.60, 0){92}
    A 4 $0.57$: [-Inf,-64.15](<<-61.7)(-Inf, 1)(-66.60, 1)(-61.7, 0){51}
     A 16 $0.43$: [-Inf,415.95](<<416)(-Inf, 1)(415.90, 1)(416, 0){33}
        => 0 {5} => 1 {1} $0.45$
     A 16 $0.43$: [415.95,Inf](415.90>>)(415.90, 0)(416, 1)(Inf, 1){33}
      A 4 $0.42$: [-Inf,-100.40](<<-95.1)(-Inf,1)(-105.7,1)(-95.1,0){27}
        => 0 {22} => 1 {1} $0.18$
      A 4 $0.42$:[-100.40,Inf](-105.7>>)(-105.7,0)(-95.1,1)(Inf,1){27}
        => 1 {3} => 0 {1} $0.56$
    A 4 $0.57$: [-64.15,Inf](-66.60>>)(-66.60, 0)(-61.7, 1)(Inf, 1) {51}
     A 1 $0.69$: [-Inf,2127](<<2148)(-Inf, 1)(2098, 1)(2174, 0) {18}
       => 1 {7} => 0 {2} $0.53$
     A 1 $0.69$: [2127,Inf](2106>>)(2098, 0)(2174, 1)(Inf, 1){18}
       => 0 {8} => 1 {1} $0.35$
   A 4 $0.69$: [215,Inf](17.80>>)(-66.60, 0)(104.60, 1)(Inf, 1){92}
    A 17 $0.37$: [-Inf,417.80](<<419.60)(-Inf, 1)(416, 1)(419.60, 0)  {41}
       => 1 {33} => 0 {4} $0.34$
    A 17 $0.37$: [417.80,Inf](416>>)(416, 0)(419.60, 1)(Inf, 1)41}
       => 1 {3} => 0 {1} $0.56$
  A 1 $0.60$: [2679,Inf](2268>>)(1089, 0)(4128, 1)(Inf, 1){196}
   A 13 $0.27$: [-Inf,409.5](<<410)(-Inf, 1)(401, 1)(418, 0){104}
      => 0 {62} => 1 {2} $0.14$
```

```
      A 13 $0.27$: [409.5,Inf](409>>)(401, 0)(418, 1)(Inf, 1){104}
        A 13 $0.42$: [-Inf,417.5](<<418)(-Inf, 1)(417, 1)(418, 0) {40}
             => 0 {26} => 1 {1} $0.16$
        A 13 $0.42$: [417.5,Inf](417>>)(417, 0)(418, 1)(Inf, 1){40}
          A 10 $0.67$: [-Inf,-102.95](<<-99)(-Inf, 1)(-106.90, 1)(-99, 0){13}
               => 0 {8} $0$
          A 10 $0.67$: [-102.95,Inf](-106.90>>)(-106.90, 0)(-99, 1)(Inf, 1){13}
               => 1 {5} $0$
  A 3 $0.38$: [1134.5,Inf](1134>>)(1078, 0)(1191, 1)(Inf, 1) {531}
    A 4 $0.13$: [-Inf,450.90](<<462.5)(-Inf, 1)(439.30, 1)(462.5, 0) {335}
      A 1 $0.1$: [-Inf,1784](<<2270)(-Inf, 1)(1141, 1)(2330, 0) {332}
        A 1 $0.24$: [-Inf,1165.5](<<1166)(-Inf, 1)(1141, 1)(1191, 0){94}
          A 13 $0.61$: [-Inf,407](<<408)(-Inf, 1)(404, 1)(410, 0){17}
               => 0 {9} => 1 {1} $0.33$
          A 13 $0.61$: [407,Inf](406>>)(404, 0)(410, 1)(Inf, 1) {17}
            A 16 $0.68$: [-Inf,422.20](<<428.40)(-Inf, 1)(416, 1)(428.40, 0){7}
                 => 1 {4} => 0 {2} $0.64$
            A 16 $0.68$: [422.20,Inf](416>>)(416, 0)(428.40, 1)(Inf, 1){7}
                 => 0 {1} $0$
        A 1 $0.24$: [1165.5,Inf](1165>>)(1141, 0)(1191, 1)(Inf, 1){94}
           => 0 {76} => 1 {1} $07$
      A 1 $0.1$: [1784,Inf](1298>>)(1141, 0)(2330, 1)(Inf, 1){332}
          => 0 {237} => 1 {1} $03$
    A 4 $0.13$: [450.90,Inf](439.30>>)(439.30, 0)(462.5, 1)(Inf, 1){335}
        => 1 {3} $0.0$
Nombre de branches: 27 Profondeur: Maximum = 7  Moyenne = 5.22
[Classification] ++ Fichier 1: test35.dat      [2"]
Stricte :[2"] [0.040"]
        --> 1851 bons pour 2000 lus. soit 92.55% (E.R.: 7.45%)
        classe  0: Bons=  925 sur 1025 soit 90.24
        classe  1: Bons=  926 sur  975 soit 94.97
de Zadeh        :[2"] [0.192"]
        --> 1851 bons pour 2000 lus. soit 92.55% (E.R.: 7.45%)
        classe  0: Bons=  952 sur 1025 soit 92.88
        classe  1: Bons=  899 sur  975 soit 92.21
de Lukasiewicz  :[2"] [0.121"]
        --> 1857 bons pour 2000 lus. soit 92.85% (E.R.: 7.15%)
        classe  0: Bons=  941 sur 1024 soit 91.89
        classe  1: Bons=  916 sur  975 soit 93.95
        Non classes     :    1
Probabiliste    :[2"] [0.148"]
        --> 1864 bons pour 2000 lus. soit 93.20% (E.R.: 6.80%)
        classe  0: Bons=  949 sur 1025 soit 92.59
        classe  1: Bons=  915 sur  975 soit 93.85
                ----------------------------------
```

Notations utilisées

$\mathcal{E}$	Ensemble d'éléments ou exemples
A_j	Un attribut
$\mathcal{A}$	Ensemble de tous les attributs possibles
C	Classe ou concept
e_i	Un exemple ou objet
$e_i(A_j)$	Valeur de l'attribut A_j pour l'exemple e_i
$e_i(C)$	Valeur de la classe pour l'exemple e_i
v_{jl}	Une modalité symbolique ou numérique-symbolique de l'attribut A_j
X	Ensemble de valeurs numériques
X_j	Ensemble des valeurs d'un attribut numérique A_j
x_{jl}	Une valeur numérique d'un attribut numérique A_j
c_k	Une valeur de la classe
n	Nombre d'exemples dans l'ensemble $\mathcal{E}$: $n = \|\mathcal{E}\|$
N	Nombre d'attributs possibles
K	Nombre de modalités de la classe
m_j	Nombre de valeurs possibles pour l'attribut A_j
$P(x)$	Probabilité de l'événement x
$P(x\|y)$	Probabilité conditionnelle de l'événement x connaissant l'événement y
$P^*(x)$	Probabilité de l'événement flou x
$H(E)$	Mesure de l'ensemble E
$H_S(E)$	Entropie de Shannon de l'ensemble E
$\mathcal{V}$	Ensemble flou d'éléments ou exemples
$\mu_{\mathcal{V}}$	Fonction d'appartenance du sous-ensemble flou $\mathcal{V}$
log	Fonction logarithme en base 2

Table des mesures

Bibliographie

Aczél, J. et Daróczy, Z. (1975). *On Measures of Information and their Characterizations*, volume 115 of *Mathematics in Science and Engineering*. Academic Press, New York.

Adamo, J. M. (1980). Fuzzy decision trees. *Fuzzy Sets and Systems*, 4:207–219.

Aladenise, Nathalie et Bouchon-Meunier, Bernadette (1997). Acquisition de connaissances imparfaites : mise en évidence d'une fonction d'appartenance. *Revue Internationale de Systémique*, 11(1):109–127.

Araya, Roberto (1994). Induction of decision trees when examples are descibed with noisy measurements and with fuzzy class membership. INRIA Seminar, Projet CLOREC.

Autebert, Jean-Marie (1987). *Langages Algébriques*. Masson.

Barone, Joseph M. (1992). Pruning fuzzy decision trees. In *Proceedings of the 2nd International Conference on Fuzzy Logic & Neural Networks*, volume 1, pages 321–324, Iizuka, Japan.

Bernhardt, M., Welch, Meryl, et Toulson, Darren L. (1997). Dual waveband infra-red target tracking and acquisition system. In *Advances in Soft-Computing Technologies and Application in Mission Systems*, n° 210 in AGARD Lecture series, pages 6.1–6.13. North Atlantic Treaty Organization.

Bloch, Isabelle (1996). Information combination operators for data fusion: A comparative review with classification. *IEEE Transations on Systems, Man, and Cybernetics*, 26(1):52–67.

Bloch, Isabelle et Maître, Henri (1992). Ensembles flous et morphologie mathématique. Technical Report 92 D 007, Télécom Paris, France.

Bloch, Isabelle et Maître, Henri (1993a). Constructing a fuzzy mathematical morphology: Alternative ways. In *Proceedings of the Second IEEE International Conference on Fuzzy Systems*, San Francisco, USA.

Bloch, Isabelle et Maître, Henri (1993b). Fuzzy mathematical morphology. *Annals of Mathematics and Artificial Intelligence*, 9(3,4).

Bothorel, Sylvie (1996). *Analyse d'image par arbre de décision flou. Application à la classification sémiologique des amas de microcalcifications.* Thèse de doctorat, Université P. et M. Curie, Paris, France.

Bouchon, Bernadette (1980). Information transmitted by a system of fuzzy events. In Lasker, G. E., editor, *Applied Systems and Cybernetics*, volume 4, pages 2767–2772. Pergamon Press. International Congress on Applied Systems Research and Cybernetics.

Bouchon, Bernadette (1981). Fuzzy questionnaires. *Fuzzy Sets and Systems*, 6(1):1–9.

Bouchon, Bernadette (1982a). Comparison and improvment of models. In Ballester, A., Cardus, D., et Trillas, E., editors, *Second World Conference on Mathematics at the Service of Man*, pages 172–175, Las Palmas.

Bouchon, Bernadette (1982b). Fuzzy information and construction of questionnaires. In *IFAC Symposium on Theory and Application of Digital Control*, volume 1, pages 22–26, New Delhi.

Bouchon, Bernadette (1985). On measure of fuzziness and information. Technical report, CNRS, Université P. et M. Curie, Paris 6. Presented at the fifth International Conference on Mathematical Modelling (Berkeley, Ca.), July 29–31, 1985.

Bouchon, Bernadette (1988). Entropic models: a general framework for measures of uncertainty and information. In Gupta, M. M. et Yamakawa, T., editors, *Fuzzy Logic in Knowledge-Based Systems, Decision and Control*, pages 93–105. North-Holland.

Bouchon-Meunier, Bernadette (1989). Incertitude, information, imprécision : une réflexion sur l'évolution de la théorie de l'information. *Revue Internationale de Systémique*, 3(4):375–385.

Bouchon-Meunier, Bernadette (1994). *La Logique Floue.* N° 2702 in Collection *Que-sais-je ?* Presses Universitaires de France, 2ème édition.

Bouchon-Meunier, Bernadette (1995). *La logique floue et ses applications.* Collection *Vie Artificielle.* Addison Wesley France.

Bouchon-Meunier, Bernadette, Delechamp, Jannick, Marsala, Christophe, et Rifqi, Maria (1997a). Several forms of fuzzy analogical reasoning. In *Proceeding of the Sixth IEEE International Conference on Fuzzy Systems, FUZZ'IEEE'97*, volume 1, pages 45–50, Barcelona, Spain.

Bouchon-Meunier, Bernadette, Marsala, Christophe, et Ramdani, Mohammed (1995). Arbres de décision et théorie des sous-ensembles flous. In *Actes des 5èmes journées du PRC-GDR d'Intelligence Artificielle*, pages 50–53.

Bouchon-Meunier, Bernadette, Marsala, Christophe, et Ramdani, Mohammed (1997b). Learning from imperfect data. In D. Dubois, H. Prade et Yager, R. R., editors, *Fuzzy Information Engineering: a Guided Tour of Applications*, pages 139–148. John Wileys and Sons.

Bouchon-Meunier, Bernadette et Nguyen, Hung T. (1996). *Les incertitudes dans les systèmes intelligents.* N° 3110 in Collection *Que-sais-je?* Presses Universitaires de France, 1ère édition.

Bouchon-Meunier, Bernadette, Rifqi, Maria, et Bothorel, Sylvie (1996). Towards general measures of comparison of objects. *Fuzzy Sets and Systems*, 84(2):143–153.

Bouzy, Bruno (1995). Modélisation cognitive du joueur de go. Thèse de l'Université P. et M. Curie, Rapport TH95/1, LAFORIA-IBP, Paris.

Boyen, Xavier (1995). Design of fuzzy logic-based decision trees applied to power system transient stability assessment. Mémoire de fin d'études, Université de Liège, Faculté des Sciences Appliquées, Belgique.

Boyen, Xavier et Wehenkel, Louis (1995). Fuzzy decision tree induction for power system security assessment. In *Proceedings of the IFAC Control of Power Plants and Power Systems, SIPOWER'95*, pages 151–156, Cancún, Mexico.

Boyen, Xavier et Wehenkel, Louis (1996). Automatic induction of continuous decision trees. In *Proceedings of the 6th International Conference IPMU*, volume 1, pages 419–424, Granada, Spain.

Bracker, Holger (1996). *Utilisation de la théorie de Dempster/Shafer pour la classification d'images satellitaires à l'aide de données multisources et multitemporelles.* Thèse de doctorat, Université de Rennes I.

Breiman, Leo, Friedman, Jerome H., Olshen, Richard A., et Stone, Charles J. (1984). *Classification And Regression Trees.* Chapman and Hall, New York.

Buntine, Wray et Niblett, Tim (1992). A further comparison of splitting rules for decision-tree induction. *Machine Learning*, 8(1):75–85. *Technical note.*

Carbonell, J. G., Michalski, Ryszard S., et Michell, T. M. (1984). An overview of machine learning. In Michalski, R. S., Carbonell, J. G., et Mitchell, T. M., editors, *Machine Learning, An Artificial Intelligence Approach*, chapter 1, pages 3–23. Springer-Verlag.

Carter, Chris et Catlett, Jason (1987). Assessing credit card applications using machine learning. *IEEE Expert*, Fall Issues:71–79.

Catlett, Jason (1991). On changing continuous attributes into ordered discrete attributes. In Kodratoff, Y., editor, *Machine Learning – Proceedings of EWSL-91*, n° 482 in Lecture notes in Artificial Intelligence, pages 164–178. Springer-Verlag.

Chang, Robin L. P. et Pavlidis, Theodosios (1977). Fuzzy decision tree algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 7(1):28–35.

Christensen, Ronald (1980). *Entropy Minimax Sourcebook*, volume III: Computer Implementation. Entropy Limited. *Collection of papers.*

Cios, Krzysztof J. et Liu, Ning (1992). A machine learning method for generation of a neural network architecture: A continuous ID3 algorithm. *IEEE Transactions on Neural Networks*, 3(2):280–290.

Cios, Krzysztof J. et Sztandera, Leszek M. (1992). Continuous ID3 algorithm with fuzzy entropy measures. In *Proceedings of the first International IEEE Conference on Fuzzy Systems*, San Diego.

Coster, Michel et Chermant, Jean-Louis (1989). *Précis d'analyse d'images.* Presses du CNRS.

Criado, F. et Gachechiladze, T. (1997). Entropy of fuzzy events. *Fuzzy Sets and Systems*, 88:99–106.

Csiszár, I. (1978). Information measures: a critical survey. In *Transactions of the 7th Prague Conference on Information Theory, Statistical Decision Functions, Random Processes and of the 1974 European Meeting of Statisticians*, volume B, pages 73–86. D. Reidel Publishing Company.

D'Alché-Buc, Florence (1993). *Modèles neuronaux et algorithmes constructifs pour l'apprentissage de règles de décision.* Thèse de doctorat, Université Paris-Sud, Orsay, France.

De Baets, Bernard, Kwasnikowska, Natalia, et Kerre, Etienne (1997). Fuzzy morphology based on conjunctive uninorms. In Mareš, M., Mesiar, R., Novák, V., Ramik, J., et Stupňanová, A., editors, *Proceedings of the Seventh International Fuzzy Systems Association World Congress*, volume 1, pages 215–220, Prague, Czech Republic.

De Luca, Aldo et Termini, Settimo (1972). A definition of a nonprobabilistic entropy in the setting of fuzzy sets theory. *Information and Control*, 20:301–312. reprinted in Readings in Fuzzy Sets for Intelligent Systems, D. Dubois, H. Prade and R. R. Yager eds, Morgan Kaufmann Publishers, 1993.

De Luca, Aldo et Termini, Settimo (1979). Entropy and energy measures of a fuzzy set. In Gupta, M.M̃. Ragade, R.K̃. et Yager, R.R̃. editors, *Advances in Fuzzy Set Theory and Applications*, pages 321–338. North-Holland.

Dougherty, James, Kohavi, Ron, et Sahami, Mehran (1995). Supervised and unsupervised discretization of continuous features. In Prieditis, Armand et Russell, Stuart, editors, *Machine Learning: Proceedings of the Twelfth International Conference*, San Francisco, CA. Morgan Kaufmann.

Dubois, Didier, Fariñas del Cerro, Luis, Herzig, Andreas, et Prade, Henri (1994). An ordinal view of independence with application to plausible reasoning. In López de Mántaras, Ramon et Poole, David, editors, *Uncertainty in Artificial Intelligence, Proceedings of the tenth conference*, pages 195–203, University of Washington, Seattle. Morgan Kaufmann Publishers.

Dubois, Didier, Mo, Xiu, et Prade, Henri (1990). Fuzzy-valued variables and fuzzy discrimination trees in pattern-directed inference. In *Eigth International Congress of Cybernetics and Systems*, New York, USA.

Dubois, Didier, Mo, Xiu, et Prade, Henri (1991). Fuzzy discrimination trees. In *Fuzzy Engineering toward Human Friendly Systems. Proceedings of the International Fuzzy Engineering Symposium IFES'91*, volume 1, pages 250–260, Yokohama, Japan.

Dubois, Didier et Prade, Henri (1982). A unifying view of comparison indices in a fuzzy set-theoretic framework. In Yager, Ronald R., editor, *Fuzzy Set and Possibility Theory*, pages 3–13. Pergamon Press.

Dubois, Didier, Prade, Henri, et Testemale, Claudette (1988). Weighted fuzzy pattern matching. *Fuzzy Sets and Systems*, 28(3):313–331.

Fayyad, Usama M. et Irani, Keki B. (1992a). The attribute selection problem in decision tree generation. In *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 104–110. AAAI.

Fayyad, Usama M. et Irani, Keki B. (1992b). On the handling of continuous-valued attributes un decision tree generation. *Machine Learning*, 8(1):87–102. *Technical note.*

Fayyad, Usama M. et Irani, Keki B. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, volume 2, pages 1022–1027.

Fayyad, Usama M., Piatetsky-Shapiro, Gregory, et Smyth, Padhraic (1996). From data mining to knowledge discovery in databases. *AI Magazine*, 17(3):37–54.

Fayyad, Usama M., Weir, Nicolas, et Djorgovski, S. (1993). SKICAT: A machine learning system for automated cataloging of large scale sky surveys. In *Machine Learning, Proceedings of the 10th International Conference*, pages 112–119, Amherst, Massachusetts. Morgan Kaufmann Publishers, Inc.

Feray, Nicolas et de Brucq, Denis (1996). Fusion d'information provenant d'une famille d'arbres de décision. application à la reconnaissance de chiffres manuscrits. In *Information Signal ImageS (ISIS), Journées thématiques en fusion d'informations, GDR-PRC*, pages 4–13, Paris, France.

Friedman, Jerome H. (1977). A recursive partitioning decision rule for nonparametric classification. *IEEE Transactions on Computers*, C-26(4):404–408.

Ganascia, Jean-Gabriel (1987). AGAPE et CHARADE : deux techniques d'apprentissage symbolique appliquées à la construction de bases de connaissances. Thèse d'état, Université de Paris-sud.

Ginsburg, Seymour (1966). *The Mathematical Theory of Context Free Languages*. McGraw-Hill, New-York.

Goffman, Casper et Pedrick, Georges (1965). *First Course in Functional Analysis.* Prentice-Hall series in Modern Analysis. Prentice-Hall Inc.

Groupe LSD (1997). Lois, structures et dépendances. In *Actes des 6e journées nationales du PRC-GDR Intelligence Artificielle*, pages 145–160, Nancy, France. Hermès.

Guessoum, Zahia (1996). *Un environnement opérationnel de conception et de réalisation de systèmes multi-agents.* Thèse de doctorat, Université Pierre et Marie Curie, Paris, France. Aussi publiée en rapport du LAFORIA-IBP n° TH96/09.

Guiaşu, Silviu et Theodorescu, Radu (1968). *La théorie mathématique de l'information.* Dunod, Paris.

Harrison, Michael A. (1978). *Introduction to Formal Language Theory.* Addison-Wesley Series in Computer Science. Addison-Wesley Pub.

Henrichon, Ernest G. et Fu, King-Sun (1969). A nonparametric partitioning procedure for pattern classification. *IEEE Transactions on Computers*, C-18(7):614–624.

Ho, Tin Kam, Hull, Jonathan J., et Srihari, Sargur N. (1992). On multiple classifier systems for pattern recognition. In *Proceedings of the 11th IAPR International Conference on Pattern Recognition*, volume II, pages 84–87, The Hague, Netherlands.

Hunt, Earl B., Marin, Janet, et Stone, Philip J. (1966). *Experiments in induction.* Academic Press.

Ichihashi, H. Shirai, T. Nagasaka, K. et Miyoshi, T. (1996). Neuro-fuzzy ID3: a method of inducing fuzzy decision trees with linear programming for maximizing entropy and an algebraic method for incremental learning. *Fuzzy Sets and Systems*, 81(1):157–167.

Ishikawa, A. et Mieno, H. (1979). The fuzzy entropy concept and its application. *Fuzzy Sets and systems*, 2(1):113–123.

ISIS (1996). *Journées thématiques en fusion d'information*, Paris. GDR-PRC Information Signal ImageS.

Jang, Jyh-Shing Roger (1994). Structure determination in fuzzy modeling: a fuzzy CART approach. In *Proceedings of the 3rd IEEE Int. Conf. on Fuzzy Systems*, volume 1, pages 480–485, Orlando. IEEE.

Janikow, Cezary Z. (1994). Fuzzy decision trees: FIDMV. In *JCIS'94 Proceedings of the Joint Conference on Information Sciences*, pages 232–235, Pinehurst, USA.

Janikow, Cezary Z. (1995). A genetic algorithm for optimizing fuzzy decision trees. In Eshelman, L. J., editor, *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 421–428, Pittsburgh. Morgan Kaufmann.

Janikow, Cezary Z. (1996). Exemplar learning in fuzzy decision trees. In *Proceedings of the 5th Conference on Fuzzy System, FUZZ'IEEE*, volume 2, pages 1500–1505, New Orleans, USA.

Kacprzyk, Janusz et Iwanski, Cesare (1990). Machine learning from misclassified and erroneous examples. In *Proceedings of the International IPMU'90 Conference*, pages 213–215, Paris.

Kacprzyk, Janusz et Iwański, Cezary (1991). Fuzzy quantifiers in inductive learning with perception of errors in data. In *Proceedings of the first International IEEE Conference on Fuzzy Systems*, pages 477–484, San Diego.

Kacprzyk, Janusz et Iwański, Cezary (1992). Learning from erroneous examples using fuzzy logic and "textbook" knowledge. In Bouchon-Meunier, B., Valverde, L., et Yager, R. R., editors, *IPMU'92 - Advanced Methods in Artificial Intelligence*, pages 183–191. Springer Verlag.

Kampé de Fériet, Joseph (1971). Mesure de l'information fournie par un événement. Séminaire sur les questionnaires.

Kampé de Fériet, Joseph (1975). L'indépendance des événements dans la théorie généralisée de l'information. In *Journées lyonnaises des questionnaires*, volume 1 of *Structures de l'Information – Publications*, pages 1–30, Paris. CNRS – Université Pierre et Marie Curie.

Kampé de Fériet, Joseph et Forte, Bruno (1967). Information et probabilité. In Gauthier-Villars, editor, *Comptes Rendus des Scéances de l'Académie des Sciences*, volume 265 of *série A*, pages 110–114, 142–146, 350–353, Paris.

Kaufmann, Arnold (1965). *Cours moderne de calcul des probabilités.* Cours de mathématiques supérieures appliquées. Albin Michel, Paris.

Kaufmann, Arnold (1975). *Introduction à la théorie des sous-ensembles flous*, volume 3 : *Applications à la classification et à la reconnaissance des formes, aux automates et aux systèmes, aux choix des critères.* Masson.

Kaufmann, Arnold (1977). *Introduction à la théorie des sous-ensembles flous*, volume 4 : *Compléments et nouvelles applications.* Masson.

Kelman, Antoine (1996). *Modèles flous pour l'agrégation de données et l'aide à la décision.* Thèse de doctorat, Université Pierre et Marie Curie, Paris, France. Aussi publiée en rapport du LAFORIA-IBP n^o TH96/12.

Kerber, Randy (1992). ChiMerge: Discretization of numeric attributes. In *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 123–128. AAAI.

Klir, Georges J. (1991). Generalized information theory. *Fuzzy Sets and Systems*, 40:127–142.

Klir, Georges J. et Folger, Tina (1988). *Fuzzy Sets, Uncertainty and Information.* Prentice Hall Eds.

Klir, Georges J. et Yuan, Bo (1995). *Fuzzy Sets and Fuzzy Logic. Theory and Applcations.* Prentice Hall.

Kodratoff, Y. et Diday, E., editors (1991). *Induction Symbolique et Numérique à Partir de données*, volume I. Cépaduès-Éditions.

Kosko, Bart (1997). *Fuzzy Engineering.* Prentice-Hall Inc.

Kuncheva, Ludmila I. (1997). An application of OWA operators to the aggregation of multiple classification decisions. In Yager, Ronald R. et Kacprzyk, Janusz, editors, *The Ordered Weighted Averaging operators. Theory and Applications*, pages 330–343, USA. Kluwer Academic Publishers.

Lerman, Israël-César et da Costa, Joaquim F. P. (1996). Cœfficients d'association et variables à très grand nombre de catégories dans les arbres de décision : application à l'identification de la structure secondaire d'une protéine. Technical Report 2803, INRIA, centre de Rennes.

Lewis, P. M. (1962). The characteristic selection problem in recognition systems. *IRE Transactions on Information Theory*, IT-8(2):171–178. Revue qui deviendra *IEEE Transaction on Information Theory.*

Loonis, Pierre (1996). *Contribution à la minimisation de l'a priori en Reconnaissance des Formes.* Thèse de doctorat, Université de La Rochelle, France.

López de Mántaras, Ramon (1991). A distance-based attribute selection measure for decision tree induction. *Machine Learning*, 6:81–92. *Technical Note.*

Lorigny, Jacques (1980). Théorie des questionnaires et reconnaissance des intitulés. Premier bilan : le chiffrement du code Profession. Technical Report 358/930, INSEE, Paris.

Maher, P. E. et Saint-Clair, D. (1993). Uncertain reasonning in an ID3 machine learning framework. In *2nd IEEE Int. Conf. on Fuzzy Systems*, pages 7–12.

Marsala, Christophe (1994). Arbres de décision et sous-ensembles flous. Rapport 94/21, LAFORIA-IBP, Université Pierre et Marie Curie, Paris, France.

Marsala, Christophe (1995). Fuzzy partition inference over a set of numerical values. Rapport 95/22, LAFORIA-IBP, Université Pierre et Marie Curie, Paris, France.

Marsala, Christophe et Bouchon-Meunier, Bernadette (1996). Fuzzy partioning using mathematical morphology in a learning scheme. In *Proceedings of the 5th Conference on Fuzzy System, FUZZ'IEEE*, volume 2, pages 1512–1517, New Orleans, USA.

Marsala, Christophe et Ramdani, Mohammed (1995). Connaissances expertes floues et système d'apprentissage descendant. In *Rencontres Francophones sur la Logique Floue et ses Applications, LFA'95*, pages 163–168, Paris. Cépaduès éditions.

Mehta, Manish, Rissanen, Jorma, et Agrawal, Rakesh (1995). MDL-based decision tree pruning. In Fayyad, U. M. et Uthurusamy, R., editors, *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 216–221, Montréal, Québec, Canada.

Ménage, Xavier (1996). *Apprentissage pour le contrôle de qualité, approche basée sur la théorie des possibilités et la théorie des réseaux de neurones.* Thèse de doctorat, Université P. et M. Curie, Paris, France.

Michalski, Ryszard S. (1983). A theory and methodology of inductive learning. In Michalski, R. S., Carbonell, J.G., et Mitchel, T. M., editors, *Machine learning: an artificial intelligence approach*, volume 1, chapter 4, pages 83–125. Morgan Kaufmann.

Michalski, Ryszard S. (1986). Understanding the nature of learning. In Michalski, R. S., Carbonell, J.G., et Mitchel, T. M., editors, *Machine Learning, an Artificial Intelligence Approach*, volume II. Morgan Kaufmann Publisher, Inc.

Mingers, John (1989a). An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, 4:227–243.

Mingers, John (1989b). An empirical comparison of selection measures for decision-tree induction. *Machine Learning*, 3(4):319–342.

Mo, Xiu (1990). *Compilation de bases de connaissances avec prise en compte de l'imprécision et de l'incertitude.* Thèse de doctorat, Université Paul Sabatier, Toulouse, France.

Munakata, Toshinori et Pawlak, Zdzislaw (1996). Rough control. application of rough set theory to control. In *Proceedings of the EUFIT'96 Conference*, pages 209–217.

Narazaki, Hiroshi et Ralescu, Anca L. (1994). An alternative method for inducing a membership function of a category. *International Journal of Approximate Reasoning*, 11(1):1–28.

Nilsson, Nils J. (1965). *Learning machines.* Morgan Kaufmann, San Mateo, California. Reprinted as *The mathematical foundations of learning machines*, Morgan Kaufmann, 1990.

Pawlak, Z. (1996). Rough sets present state and perspectives. In *Proceedings of the 6th International Conference IPMU*, volume 3, pages 1137–1145, Granada, Spain.

Pazzani, Michael J. (1995). An iterative improvement approach for the discretization of numeric attributes in bayesian classifiers. In Fayyad, U. M. et Uthurusamy, R., editors, *Proceedings of the First International conference on Knowledge Discovery and Data Mining*, pages 228–233, Montréal, Québec, Canada.

Pearl, Judea (1988). *Probabilistic reasoning in intelligent systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers Inc., San Mateo, California.

Picard, Claude-François (1963). *Théorie des questionnaires.* Thèse de doctorat, Université des Sciences Mathématiques, Paris.

Picard, Claude-François (1965). *Théorie des questionnaires.* Gauthier-Villars. Traduit en allemand : *Théorie der Fragebogen*, Akademie-Verlag, Berlin 1971.

Picard, Claude-François (1972). *Graphes et Questionnaires*, volume Tome 2 of *Collection "Programmation".* Gauthier-Villars, Paris.

Picard, Claude-François (1980). *Graphs and questionnaires.* North Holland.

Quinlan, J. Ross (1984). Learning efficient classification procedures and their application to chess endgames. In Michalski, R. S., Carbonell, J.G., et Mitchell, T. M., editors, *Machine Learning, An Artificial Intelligence Approach*, chapter 15, pages 463–482. Springer-Verlag.

Quinlan, J. Ross (1986). Induction of decision trees. *Machine Learning*, 1(1):86–106.

Quinlan, J. Ross (1990). Probabilistic decision trees. In Michalski, R. S., Carbonell, J. G., et Mitchell, T. M., editors, *Machine Learning*, volume 3, chapter 5, pages 140–152. Morgan Kaufmann Publishers.

Quinlan, J. Ross (1993). *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, Ca.

Quinlan, J. Ross (1996). Improved use of continuous attributes in C4.5. *Journal of Artificial Intelligence Research*, 4:77–90.

Rabaséda-Loudcher, Sabine (1996). *Contributions à l'extraction automatique de connaissances. Une application à l'analyse clinique de la marche.* Thèse de doctorat, Université Lumière Lyon 2, France.

Rabaséda-Loudcher, Sabine, Sebban, Marc, et Rakotomalala, Ricco (1995). Discretisation of continuous attribute: a survey of methods. In *Proceedings of the 2nd annual Joint Conference on Information Sciences*, pages 164–166, Wrightsville Beach, North Carolina, USA.

Ralescu, Dan (1994). Fuzzy probabilities and their applications to statistical inference. In Bouchon-Meunier, B., Yager, R. R., et Zadeh, L. A., editors, *Advances in Intelligent Computing – IPMU'94*, Lecture Notes in Computer Sciences, pages 217–222. Springer. selected papers of the proceedings of the 5th International Conference on Information and Management of Uncertainty in Knowledge-Based Systems, Paris, France, July 4–8, 1994.

Ralescu, Dan (1995). Cardinality, quantifiers, and the aggregation of fuzzy criteria. *Fuzzy Sets and Systems*, 69:355–365.

Ramdani, Mohammed (1992). Une approche floue pour traiter les valeurs numériques en apprentissage. In *Journées Francophones d'apprentissage et d'explication des connaissances*.

Ramdani, Mohammed (1993). Apprentissage à partir de données imparfaites. In *1er congrés biennal de l'association française des sciences et technologies de l'information et des systèmes, AFCET'93*, pages 209–218, Versailles.

Ramdani, Mohammed (1994). *Système d'Induction Formelle à Base de Connaissances Imprécises*. Thèse de doctorat, Université P. et M. Curie, Paris, France. Aussi publiée en rapport du LAFORIA-IBP n^o TH94/1.

Rifflet, Jean-Marie (1992). *La communication sous UNIX*. Addison-Wesley, Paris, France.

Rifqi, Maria (1996). *Mesures de comparaison, typicalité et classification d'objets flous : théorie et pratique*. Thèse de doctorat, Université P. et M. Curie, Paris, France. Aussi publiée en rapport du LAFORIA-IBP n^o TH96/15.

Sanchez, Elie (1979). Inverses of fuzzy relations. Application to possibility distributions and medical diagnosis. *Fuzzy Sets and Systems*, 2(1):75–86.

Sanchez, Elie, Shibata, Takanori, et Zadeh, Lotfi A., editors (1997). *Genetic algorithm and fuzzy logic systems. Soft Computing Perspectives.*, volume 7 of *Advances in Fuzzy Systems – Applications and Theory*. World Scientific Publishing Co.

Serra, Jean-Paul (1982). *Image Analysis and Mathematical Morphology*. Academic Press, New York.

Shafer, Glenn A. (1976). *Mathematical Theory of Evidence*. Princeton University Press, New Jersey.

Shafer, Glenn A. (1987). Belief functions and plausibility measures. In Bezdek, James C., editor, *Analysis of Fuzzy Information*, volume I, chapter 3, pages 51–84. CRC Press.

Shafer, Glenn A. Shenoy, Prakash P. et Mellouli, Khaled (1987). Propagating belief functions in qualitative markov trees. *International Journal of Approximate Reasoning*, 1:349–400.

Shannon, Claude E. (1948). *The Mathematical Theory of Communication*. University of Illinois Press, Urbana, USA. C. E. Shannon and W. Weaver Eds.

Shavlik, J. W. et Dietterich, T. G., editors (1990). *Readings in Machine Learning*. Morgan Kaufmann Publishers, Inc.

Shibata, Takanori (1997). Skill acquisition and skill-based motion planning for hierarchical intelligent control of a redundant manipulator. In Sanchez, E., Shibata, T., et Zadeh, L. A., editors, *Genetic algorithms and Fuzzy Logic Systems. Soft Computing Perspectives*, volume 7 of *Advances in Fuzzy Systems*, pages 19–35. World Scientific.

Shibata, Takanori, Abe, Tamotsu, Tanie, Kazuo, et Nose, Matsuo (1995). Motion planning of a redundant manipulator based on criteria of skilled operators using fuzzy-ID3 and GMDH. In *IFSA'95 Proceedings of the 6th International Fuzzy Systems Association World Congress*, volume 1, pages 613–616, São Paulo, Brazil.

Simon, Herbert A. (1984). Why should machines learn? In Michalski, R. S., Carbonell, J. G., et Mitchell, T. M., editors, *Machine Learning, An Artificial Intelligence Approach*, volume 1, chapter 2. Springer-Verlag.

Smets, P. (1982). Probability of a fuzzy event: an axiomatic approach. *Fuzzy Sets and Systems*, 7:153–164.

Sugeno, M. (1977). Fuzzy measures and fuzzy integrals - a survey. In Gupta, M. M., Saridis, G. N., et Gaines, B. R., editors, *Fuzzy Automata and Decision Processes*, pages 89–102. North Holland.

Tanaka, H. Okuda, T. et Asai, K. (1979). Fuzzy information and decision in statistical model. In Gupta, M.M̃. Ragade, R.K̃. et Yager, R.R̃. editors, *Advances in Fuzzy Set Theory and Applications*, pages 303–320. North-Holland.

Terrenoire, M. (1970). *Un modèle mathématique de processus d'interrogation : les pseudo-questionnaires.* Thèse de doctorat, Université de Grenoble.

Tours (1983). *Journées tourangelles, "Utilisation de l'information, des questionnaires et des ensembles flous dans les probl'emes décisionnels"*, volume 34 of *Publications, Structure de l'information*, Paris. CNRS/Université Pierre et Marie Curie, Groupe de recherche Claude François Picard.

Umano, M., Okamoto, H., Hatono, I., Tamura, H., Kawachi, F., Umedzu, S., et Kinoshita, J. (1994). Fuzzy decision trees by fuzzy ID3 algorithm and its application to diagnosis systems. In *Proceedings of the 3rd IEEE Conference on Fuzzy Systems*, volume 3, pages 2113–2118, Orlando.

Utgoff, Paul E. et Clouse, Jeffrey A. (1996). A Kolmogoroff-Smirnoff metric for decision tree induction. Technical Report 96-3, University of Massachusetts, Amherst, MA, USA.

Van de Merckt, Thierry (1993). Decision trees in numerical attribute spaces. In *IJCAI-93 Proceedings of the 13th International Joint Conference on Artificial Intelligence*, volume 2, pages 1016–1021.

Watanabe, Satosi (1960). Information-theoretical aspects of inductive and deductive inference. *IBM Journal of research and development*, 4(2):208–231.

Weber, Richard (1992). Fuzzy-ID3: A class of methods for automatic knowledge acquisition. In *IIZUKA'92 Proceedings of the 2nd International Conference on Fuzzy Logic*, volume 1, pages 265–268.

Wehenkel, Louis (1993). Decision tree pruning using an additive information quality measure. In Bouchon-Meunier, B., Valverde, L., et YAGER, R. R., editors, *Uncertainty in Intelligent Systems*. Elsevier – North Holland.

Wehenkel, Louis (1996). On uncertainty measures used for decision tree induction. In *Proceedings of the 6th International Conference IPMU*, volume 1, pages 413–418, Granada, Spain.

Wehenkel, Louis (1997). Discretization of continuous attributes for supervised learning. variance evaluation and variance reduction. In Mareš, M., Mesiar, R., Novák, V., Ramik, J., et Stupňanová, A., editors, *Proceedings of the Seventh International Fuzzy Systems Association World Congress*, volume 1, pages 381–388, Prague, Czech Republic.

Wehenkel, Louis et Pavella, Mania (1996). Why and which automatic learning approaches to power systems security assessment. In *Proceedings of CESA'96, IMACS Multiconference on Computational Engineering in Systems Appliactions*, pages 1072–1077, Lille, France.

Xie, W. X. et Bedrosian, S. D. (1984). An information measure for fuzzy sets. *IEEE Transactions on Systems, Man and Cybernetics*, 14(1):151–156.

Xu, Lei, Krzyzak, Adam, et Suen, Ching Y. (1992). Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 22(3):418–435.

Xuecheng, Liu (1992). Entropy, distance measure and similarity measure of fuzzy sets and their relations. *Fuzzy Sets and Systems*, 52:305–318.

Yager, Ronald R. (1977). A note on probabilities of fuzzy events. Technical Report RRY 77-14B, Iona College, New Rochelle, New York 10801. Published in Information Sciences 18 (1979) pp. 113–122.

Yager, Ronald R. (1984). A representation of the probability of a fuzzy subset. *Fuzzy Sets and Systems*, 13:273–283.

Yager, Ronald R. (1988). On ordered weighted average aggregation operators in multicriteria decision making. *IEEE Transactions on Systems, Man and Cybernetics*, 18:183–190.

Yaglom, A. M. et Yaglom, I. M. (1959). *Probabilité et Information.* monographie. Dunod, Paris.

Yuan, Yufei et Shaw, Michael J. (1995). Induction of fuzzy decision trees. *Fuzzy Sets and systems*, 69:125–139.

Zadeh, Lotfi A. (1965). Fuzzy sets. *Information and Control*, 8:338–353.

Zadeh, Lotfi A. (1968). Probability measures of fuzzy events. *Journal Math. Anal. Applic.*, 23. reprinted in "*Fuzzy Sets and Applications: selected papers by L. A. Zadeh*", R. R. Yager and S. Ovchinnikov and R. M. Tong and H. T. Nguyen eds, pp. 45–51.

Zadeh, Lotfi A. (1975). Calculus of fuzzy restrictions. In Zadeh, L. A., Fu, K. S., Shimura, M., et Tanaka, K., editors, *Fuzzy Sets and Their Applications to Cognitive and Decision Processes*. Academic Press, New York, USA.

Zadeh, Lotfi A. (1976). The concept of a linguistic variable and its application to approximate reasoning, part 3. *Information Sciences*, 9:43–80.

Zadeh, Lotfi A. (1978). Pruf-a meaning representation language for natural languages. *Int. Journal of Man-Machine Studies*, 10:395–460.

Zadeh, Lotfi A. (1983). The role of fuzzy logic in the management of uncertainty in expert systems. *Fuzzy Sets and Systems*, 11:199–227. reprinted in "*Fuzzy Sets and Applications: selected papers by L. A. Zadeh*", R. R. Yager and S. Ovchinnikov and R. M. Tong and H. T. Nguyen eds, pp. 413–441.

Zadeh, Lotfi A. (1997). Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. *Fuzzy Sets and Systems*, 90(2):111–127.

Zeidler, Jens et Schlosser, Michael (1996). Continuous-valued attributes in fuzzy decision trees. In *Proceedings of the 6th International Conference IPMU*, volume 1, pages 395–400, Granada, Spain.

Zhang, Lianwen (1993). Representation, independence, and combination of evidence in the Dempster-Shafer theory. In Yager, Ronald R., Kacprzyk, Janusz, et Fedrizzi, Mario, editors, *Advances in the Dempster-Shafer Theory of Evidence*, pages 51–69. Wiley Sons.

Zighed, Djamel A. (1985). *Méthode et outils pour les processus d'interrogation non arborescents*. Thèse de doctorat, Université Lyon I.

Zighed, Djamel A., Auray, Jean-Paul, et Duru, Gérard (1991). *Sipina, méthode et logiciel*. N° 2 in Collection Mathématiques Appliquées. Édition Alexandre Lacassagne, Lyon.

Zighed, Djamel A., Rakotomalala, Ricco, et Rabaséda, Sabine (1996). A discretization method of continuous attributes in induction graphs. In Trappl, R., editor, *Cybernetics and systems'96*, volume 2, pages 997–1002. Austrian Society for Cybernetics Studies.