Vers une approche hybride mêlant arbre de classification et treillis de Galois pour de l'indexation d'images

Nathalie N. Girard

UNIVERSITÉ DE LA ROCHELLE

ÉCOLE DOCTORALE S2iM

LABORATOIRE : L3I - Informatique, Image, Interaction

THÈSE présentée par :

Nathalie GIRARD

soutenue le : **5 juillet 2013**

pour obtenir le grade de : **Docteur de l'université de La Rochelle**

Discipline : **Informatique et applications**

Vers une approche hybride mêlant arbre de classification et treillis de Galois pour de l'indexation d'images

---

JURY :

François BRUCKER	Professeur des Universités, École Centrale de Marseille, Rapporteur
Engelbert MEPHU-NGUIFO	Professeur des Universités, Université B. Pascal, Clermont-Ferrand, Rapporteur
Amedeo NAPOLI	Directeur de recherche CNRS au LORIA, Nancy, Président du jury
Jean-Marc OGIER	Professeur des Universités, Université de La Rochelle, Examinateur
Karell BERTET	Maître de Conférences HdR, Université de La Rochelle, Directrice de thèse
Muriel VISANI	Maître de Conférences, Université de La Rochelle, Encadrante

Thèse réalisée au Laboratoire Informatique, Image et Interaction (L3i)
Pôle Sciences & Technologies, Université de La Rochelle
Avenue M. Crépeau,
17042 La Rochelle cedex 01

Tél : +33 5 46 45 82 62
Fax : + 33 5 46 45 82 42

Web : http ://l3i.univ-larochelle.fr/

Sous la direction de Karell Bertet karell.bertet@univ-lr.fr

Co-encadrement Muriel Visani muriel.visani@univ-lr.fr

Financement Allocation Ministérielle

Remerciements

Je tiens tout d'abord à remercier Engelbert Mephu-Nguifo et François Brucker d'avoir accepté d'être les rapporteurs de cette thèse. Je remercie également Amedeo Napoli, et Jean-Marc Ogier d'avoir accepté de participer à mon jury en tant qu'examinateurs. Leurs commentaires m'ont été très utiles pour la mise à jour de ce manuscrit pour sa version finale. Les diverses discussions seront également une source d'inspiration importante dans la poursuite de mes travaux.

Je tiens à remercier Karell Bertet et Muriel Visani qui m'ont encadrée au cours de cette thèse. Leurs thématiques de recherche différentes m'ont permis de m'enrichir et d'apprendre énormément. Merci pour les différents échanges scientifiques ainsi que les conférences partagées.

Plus généralement, je tiens aussi à remercier Stéphanie Guillas et Karell qui m'ont fait découvrir la recherche et surtout les treillis de Galois au cours de mon stage de licence. Ce fût certainement l'un des facteurs déterminants qui m'a fait m'intéresser à la recherche. Merci à Étienne Baudrier et Jean-Marc qui m'ont permis de découvrir l'analyse d'images au cours de mon stage de Master, ce qui m'a confortée dans mon orientation. Merci pour vos conseils avisés.

Je remercie les membres du L3i de la Rochelle pour leur accueil. Merci à l'ancienne équipe Imedoc de m'avoir permis de participer à l'organisation de GREC 2009, moment à la fois enrichissant et si sympathique. Merci l'ensemble des doctorants pour les moments partagés aussi bien culinaires, que scientifiques.

J'ai eu la chance de découvrir les joies de l'enseignement au cours de cette thèse. Je tiens à remercier ceux qui m'ont fait profiter de leur expérience, une mention particulière à Bertrand Vachon qui, en tant que responsable des moniteurs, nous a donné de nombreux conseils pédagogiques. Mais aussi, à Frédéric Bertrand pour sa confiance en me permettant d'intervenir dans un cours à long terme (4 ans), ainsi que l'équipe d'info. d'usage Vincent, Pedro, JC, Anthony et Sandrine, mais aussi à Karell et Muriel avec qui j'ai aussi pu partager des cours et des discussions pédagogiques.

Enfin et surtout, Jean-Luc Sabourin que j'aurai eu la chance d'avoir à la fois comme

enseignant lorsque j'étais étudiante mais aussi comme tuteur de monitorat. Ses conseils étaient précieux et sa bonne humeur toujours présente.

Je tiens aussi à remercier le LI et l'école polytechnique de Tours pour leur accueil ces derniers mois, et tout particulièrement l'équipe RFAI.

Une dédicace spéciale à la 121bis team (Micka., Matthieu, Nicolas, Élodie, Toï et Christophe), avec laquelle j'ai passé de très bons moments. Merci pour les dépannages (si nombreux ;)), les conseils SAV (tout aussi nombreux) et les discussions enrichissantes (enfin en général ;)). Merci à Cyril, pour l'entraide au cours de l'été passé ;). Merci à Kathy, Dom., Christelle, Caro. et Sarah pour les nombreux moments partagés dans la bonne humeur.

Enfin, je tiens à remercier mes proches et amis qui m'ont soutenue. Stéph. pour tes corrections patientes, Sab. pour les moments de décompression et l'équipe BZH pour les titres obtenus et tous les bons moments partagés. Finalement, une thèse c'est un peu comme un match, d'abord tu observes ce que les autres font, puis ensuite tu développes - ton jeu - et à la fin tu fais tout ce que tu peux pour obtenir le meilleur résultat !

Et bien entendu rien n'aurait été possible sans ma famille. Merci à mes parents et grands-parents pour leur soutien. A ma mère pour ses relectures et corrections, pour son coup de pinceau toujours aussi efficace lors des travaux, et enfin pour le délicieux pot qu'elle nous a préparé ! ! A mon père pour l'apéritif, à mon frère Mickaël pour ses remarques qui font finalement bien relativiser certaines choses, ainsi qu'à mon oncle et ma tante pour les pauses au voilier (et pour les travaux) ;)

Table des matières

Vers un modèle hybride 113

Table des figures

Liste des tableaux

Liste des algorithmes

Introduction

Contexte

Dans de nombreux domaines scientifiques, des données sont recueillies afin d'être analysées. Ces données sont de tous types et sont porteuses d'informations très variées. Nos travaux portent principalement sur **l'analyse de données extraites d'images**.

Les images considérées au sein de notre laboratoire, sont pour la plupart, des images graphiques issues de documents techniques ou historiques telles que des images de symboles électriques ou encore des images de lettrines. Ces images sont plus ou moins détériorées, selon leurs origines. Par exemple, les images issues de documents historiques sont détériorées en raison de la dégradation du papier (taches, parties effacées), ou encore des artéfacts dus à la numérisation peuvent être présents sur les images issues de documents scannés,. . ..

De nombreux travaux [Guillas 07, Prum 10, Coustaty 11b, Coustaty 11a] ont été menés, au sein du laboratoire, afin d'extraire de ces images des éléments porteurs de sens. Il en résulte des **signatures d'images** contenant une information condensée représentant par exemple la texture, les couleurs, ou encore les contours des formes contenues dans l'image. Les signatures permettent de décrire les images mais aussi et surtout de les caractériser dans un objectif de reconnaissance et/ou de classification d'images.
Les signatures extraites sont généralement des vecteurs numériques qui se présentent sous forme tabulaire (table : objets $\times$ attributs). Telles quelles, ces signatures sont difficilement interprétables, pour les analyser et en extraire de l'information, il existe différents modèles avec des objectifs variés. Pour notre part, nous nous intéressons à **la classification supervisée**. Plus précisément, nous nous intéressons à la définition de modèles de classification à partir d'un ensemble d'apprentissage contenant des exemples déjà répartis dans différents groupes (ou classes). Le modèle de classification construit par apprentissage est ensuite utilisé pour classer de nouveaux exemples dans les groupes pré-définis (*i.e.* le modèle prédit la valeur de l'attribut « groupe » pour ces exemples ; l'attribut groupe est appelé variable à prédire).

De nombreux modèles de classification supervisée sont proposés dans la littérature, **des**

modèles symboliques mais aussi des modèles dits numériques (*i.e.* statistiques, probabilistes) qui se distinguent généralement par le type d'attributs qu'ils analysent (attributs qualitatifs/symboliques ou attributs quantitatifs/numériques). Le choix de l'approche la plus adaptée au domaine (classification supervisée d'images) ainsi que son paramétrage restent des problématiques difficiles souvent guidées par les performances expérimentales (taux de reconnaissance, complexité du modèle - *i.e.* nombre concepts dans la structure -, ...).

Les approches numériques sont souvent comparées à des boîtes noires, du fait de la difficulté à comprendre leur fonctionnement et les raisons des résultats obtenus.

Les approches symboliques sont souvent plus intuitives, elles permettent une meilleure compréhension des données et des résultats obtenus. Elles offrent par leur lisibilité l'avantage de pouvoir choisir et adapter les paramètres du modèle lorsque cela est nécessaire.

L'utilisation de ces méthodes symboliques rend les modèles de classification et leurs résultats plus accessibles à des personnes non expertes du domaine informatique.

Dans les travaux présentés dans ce manuscrit, nous nous intéressons à deux modèles symboliques, l'arbre de décision et le treillis de Galois.

L'arbre de décision est un modèle symbolique utilisé depuis de nombreuses années dans les tâches d'apprentissage automatique. Les statisticiens en attribuent la paternité à Morgan et Sonquist [Morgan 63], ces derniers ayant introduit des arbres de régression (*i.e.* effectuant un apprentissage supervisée où la variable à prédire est quantitative). Ensuite, la famille des arbres de classification (supervisés) a suivi avec les **algorithmes d'induction** les plus populaires tels que ID3 [Quinlan 79, Quinlan 86a] défini par Quinlan en 1979, puis ChAID [Kass 80] défini par Kass en 1980, suivi de CART [Breiman 84] défini par Breiman dans son livre réputé en 1984. Puis, en 1993 Quinlan apporte des améliorations à son algorithme ID3 qu'il fait alors évoluer vers l'algorithme C4.5 [Quinlan 93] où il introduit entre autre le post-élagage et le traitement des données quantitatives.

L'arbre de classification est un graphe dont les nœuds contiennent des objets (des exemples de l'ensemble d'apprentissage) et les arcs sont associés à des tests sur les valeurs des attributs. Les algorithmes de construction cités précédemment diffèrent par rapport aux critères qu'ils utilisent pour construire l'arbre. En effet, la construction d'un arbre nécessite la définition d'un critère de division (ou de gain) pour guider la division des nœuds vers les fils, mais aussi d'un critère d'arrêt pour stopper l'expansion de l'arbre et étiqueter les feuilles obtenues avec la classe optimisant une fonction objectif sur les exemples de cette feuille. Parfois, un critère d'élagage est aussi utilisé pour simplifier la structure construite. Une fois l'arbre construit celui-ci est généralement utilisé pour classer de nouveaux exemples par navigation dans la structure. Pour ce faire, la structure est parcourue à partir de la racine jusqu'à une feuille, ce parcours étant réalisé par validation des tests (associés aux arcs) pour passer d'un nœud à l'un de ses successeurs. Le nouvel objet est alors classé dans la classe de la feuille atteinte.

De nombreuses études ont été réalisées pour comparer les performances des différents arbres [Dougherty 95, Quinlan 93, Rakotomalala 05a], pour proposer de nouveaux critères afin d'améliorer la construction de ceux-ci [Zighed 96, Zighed 00, Do 10], mais aussi pour augmenter leurs performances [Quinlan 93, Breiman 96a, Quinlan 96b].

Pour notre part, nous nous intéressons aux arbres de classification en raison de leurs liens avec les treillis de Galois ou treillis des concepts.

Un treillis est une structure algébrique utilisée en analyse de données depuis le début des années 1970. Dés la fin du 19éme siècle, la notion de treillis avait été introduite en tant que structure algébrique munie de deux opérateurs appelés borne inférieure et borne supérieure. Cette structure algébrique peut aussi être vue comme une structure ordonnée, définie par l'existence d'éléments particuliers appelés bornes supérieure et inférieure. Cependant le premier ouvrage de référence sur la théorie des treillis est la première édition du livre de Birkhoff en 1940 [Birkhoff 40], celui-ci sera suivi par deux nouvelles éditions en 1948 et 1967 [Birkhoff 48, Birkhoff 67]. On peut faire la distinction entre les éléments correspondant à une borne (éléments réductibles) et ceux qui ne sont pas une borne supérieure (sup-irréductible) ou borne inférieure (inf-irréductible). Les éléments irréductibles s'organisent sous forme d'une table binaire avec les inf-irréductibles en lignes et les sup-irréductibles en colonnes. Ils décrivent la structure même du treillis et permettent sa reconstruction. Ensuite en 1970 [Barbut 70], Barbut et Monjardet présentent un résultat fondamental de la théorie des treillis établissant que tout treillis fini est isomorphe au treillis de Galois de sa table binaire. Barbut et Monjardet introduisent ainsi le terme de **treillis de Galois**.

Depuis les années 2000, l'émergence de **l'analyse formelle des concepts (AFC)** [Ganter 99] dans divers domaines de l'informatique, que ce soit en extraction et représentation des connaissances, ou en bases de données, a ainsi mis en avant les structures de treillis des concepts.

Les treillis de Galois, aussi appelés treillis de concepts, **sont alors appliqués aux tables de données binaires (objets $\times$ attributs), nommées contextes**. Dans un cadre applicatif tel que la classification, la structure de **treillis de Galois fournit une représentation très intuitive des données** ; le treillis étant alors représenté sous forme d'un graphe, comme les arbres de classification. Ici, les nœuds contiennent un ensemble d'objets associé à un ensemble d'attributs et deux nœuds sont reliés par un arc lorsqu'une relation d'inclusion est vérifiée entre les ensembles de ces deux concepts. Les nœuds du treillis sont alors appelés **concepts formels**.

Le treillis de Galois est défini de façon unique pour une table binaire, il n'est pas dépendant de paramétrages comme peut l'être l'arbre de classification. Les liens entre arbres de classification et treillis de Galois sont de plus en plus étudiés ces dernières années [Polaillon 98, Belohlávek 07, Nijssen 07, Guillas 08, Bertet 09, Nijssen 10]. En particulier, la classification par navigation dans les deux structures est très proche. Comme mentionné précédemment, dans l'arbre de classification, la structure est par-

courue à partir de la racine jusqu'à une feuille. La méthode Navigala, introduite par Guillas [Guillas 07, Visani 11], s'inspire de cette navigation dans les arbres de classification, la structure du treillis est parcourue à partir du plus petit concept par rapport à la relation d'ordre, appelé concept minimal, jusqu'à un concept contenant (majoritairement) des objets de même classe, appelé **concept terminal**. Là encore, le parcours est réalisé par validation d'attributs pour passer d'un concept à l'un de ses successeurs. Cependant, de par sa définition, le treillis contient plus de chemins, et plus particulièrement il contient plusieurs chemins pour atteindre un même concept terminal, tandis que l'arbre ne contient qu'un seul chemin de la racine à une feuille particulière. De ce fait, et tel que démontré dans [Guillas 07], le treillis est plus robuste pour la classification de données bruitées.

Cependant, les méthodes symboliques ont généralement été conçues pour l'analyse de données décrites par des attributs qualitatifs. Or, dans le contexte de l'analyse d'images, les signatures extraites sont pour la plupart constituées d'attributs quantitatifs. Les méthodes symboliques nécessitent donc une phase de transformation des données lorsque les données initiales sont décrites par ce type d'attributs, cette transformation est appelée **discrétisation**. Lors de la discrétisation, les valeurs de chaque attribut quantitatif sont réparties dans des intervalles qui deviennent des modalités d'un attribut qualitatif. L'intérêt des chercheurs pour le domaine de la discrétisation date du début des années 90 [Fayyad 93, Dougherty 95, Quinlan 96b, Zighed 97, Muhlenbach 02, Muhlenbach 05]. Bien qu'utilisée auparavant dans les modèles symboliques, elle n'était jamais mentionnée en tant que telle [Breiman 84]. Cet intérêt est fortement lié au développement des ordinateurs et de leurs capacités de calcul. Ces derniers ont permis le développement de méthodes d'apprentissage (telle que la définition des arbres de classification) intégrant tous types de données, la discrétisation devenant alors une étape importante de la phase d'apprentissage.

Selon le modèle symbolique utilisé, différents types de discrétisation existent, soit en pré-traitement (*i.e.* avant la définition du modèle) il s'agit de **discrétisation globale** [Dougherty 95, Frank 99], soit au cours de la définition du modèle, il s'agit alors de **discrétisation locale** [Quinlan 96a, Rakotomalala 05a]. Pour la construction des arbres de classification, une discrétisation locale est généralement utilisée, celle-ci a la particularité d'être guidée par le modèle construit. En effet, dans ce cas, les attributs sont discrétisés localement, *i.e.* de manière indépendante pour chaque nœud de l'arbre, et la discrétisation est réalisée en fonction des observations et des attributs contenus dans le nœud courant. Pour la construction des treillis de Galois, c'est en général une discrétisation globale qui est utilisée, celle-ci ne tient pas compte de la structure du modèle construit [Ganter 99, Guillas 07, Kaytoue 09b].

Les intervalles construits au cours d'une discrétisation sont le plus souvent disjoints [Fayyad 93, Dougherty 95, Muhlenbach 05], mais ils peuvent aussi être non disjoints [Ganter 99]. Selon s'ils sont disjoints ou non, les modèles de classification construits peuvent vérifier différentes propriétés. En particulier, comme nous le verrons dans la

suite, les treillis de Galois définis suite à une discrétisation en intervalles disjoints sont des treillis dichotomiques.

La définition de treillis de Galois à partir de données quantitatives est un domaine de recherche où les contributions sont généralement liées à l'application souhaitée [Ganter 99, Polaillon 98, Pfaltz 07, Kaytoue 09b, Kaytoue 11]. Les liens forts existants entre les arbres de classification et les treillis de Galois ainsi que la définition existante des arbres de classification à partir de données quantitatives, nous semblent deux notions importantes à étudier pour **définir un modèle hybride entre ces deux modèles. Le modèle hybride profitera des avantages de chacun** à savoir le faible espace mémoire et le traitement adapté (*i.e.* local) des données quantitatives de l'arbre, ainsi que la robustesse aux données bruitées du treillis, tout en conservant la lisibilité des deux modèles.

Objectifs

L'objectif majeur de cette thèse est de proposer un modèle de classification symbolique hybride entre arbre de classification et treillis de Galois pour le traitement de données quantitatives. Pour cela, nous nous focaliserons sur :

- la gestion des données quantitatives par les treillis, en généralisant au treillis le modèle de discrétisation locale qui permet aux arbres de classification de prendre en compte les données quantitatives, de manière plus efficace et plus adaptée à la structure même du modèle ;

- l'amélioration de l'espace mémoire nécessaire aux treillis en proposant un modèle de simplification de la structure, basé sur un indice local de stabilité des concepts du treillis et une adaptation automatique du seuil par optimisation des performances du treillis.

Notons de plus, que nous souhaitons que le modèle hybride ait la même robustesse que les treillis face aux données bruitées.

Contributions

Nos contributions se situent à différents niveaux :

- Contribution 1 (p 115) : Tout d'abord, **nous rappelons les liens existants entre arbres de classification et treillis de Galois**. En particulier, nous détaillons **les liens en usage** (classification par navigation) et **les liens structurels** (**inclusion** des arbres de classification dans le treillis, et correspondance entre d'une part la

fusion de tous les arbres et d'autre part le treillis). Le lien de fusion est lié à la définition de treillis particuliers que sont les treillis dichotomiques, *i.e.* les treillis définis à partir de données où les attributs sont complémentaires, c'est le cas des attributs qualitatifs construits par discrétisation en intervalles disjoints. **Nous donnons la définition formelle des treillis dichotomiques et nous démontrons leurs propriétés.**

- Contribution 2 (p 166) : Ensuite, **nous proposons une discrétisation locale pour les treillis** basée sur les propriétés des treillis dichotomiques et qui s'inspire de la discrétisation locale mise en œuvre dans les arbres de classification. En effet, selon différentes études la discrétisation locale permet d'obtenir de meilleurs résultats en classification avec les arbres de classification, comparée à une discrétisation globale. Parce qu'elle est guidée par le modèle construit, la discrétisation locale est plus adaptée à celui-ci. **Nous montrons par des expérimentations que la discrétisation locale** que nous avons définie, **permet de diminuer la complexité des treillis tout en conservant leur robustesse** face aux données bruitées. De fait, la discrétisation locale étant guidée par le modèle de classification construit, les attributs qualitatifs ainsi définis sont plus adaptés à la structure de treillis, d'où une complexité structurelle observée, diminuée en pratique.
- Contribution 3 (p 193) : Enfin, **nous étudions les possibilités de simplification de la structure de treillis**. Il s'agit de supprimer, une fois le treillis complètement construit, les concepts les moins utiles en termes de classification, cela afin de diminuer encore sa complexité tout en conservant ses performances en classification. Nous étudions **deux méthodes qui se distinguent par la monotonie du critère utilisé**. Lorsque le critère utilisé est monotone, la structure obtenue est un $\vee$-demi-treillis, car les concepts sont supprimés de manière ascendante, un concept interne ne peut être supprimé tant que ses successeurs ne l'ont pas été. Lorsque le critère utilisé est non-monotone, la propriété de treillis n'est pas conservée car les concepts peuvent être supprimés n'importe où dans la structure. La première méthode utilise **une adaptation de la mesure de coût-complexité** existante pour l'élagage des arbres, ce critère est monotone. La seconde méthode, quant à elle, utilise un indice propre aux treillis, **l'indice de stabilité des concepts**, ce critère est non-monotone. La première méthode, inspirée du post-élagage dans les arbres de classification, n'engendre pas nécessairement de diminution du nombre de concepts du treillis entre deux étapes consécutives, mais plutôt une diminution du nombre d'arcs, et donc une réduction de la robustesse du treillis vis-à-vis des données bruitées. Contrairement à la première, la seconde méthode nous permet de proposer une simplification efficace de la structure avec une diminution du nombre de concepts allant jusqu'à 50% tout en maintenant les performances en classification. **Nous obtenons ainsi un modèle hybride entre arbre de classification et treillis de Galois, qui est aussi robuste que le treillis complet tout en étant moins complexe.**

Organisation de la thèse

Ce manuscrit comprend deux parties distinctes, intitulées « État de l'art » et « Vers un modèle hybride ». La première partie nous permet d'introduire de manière générale le cadre de cette thèse et le domaine de recherche de l'analyse de données. Mais aussi de présenter les deux modèles symboliques objets de cette thèse : l'arbre de classification et le treillis de Galois. Nous y présentons les propriétés de ces modèles lorsqu'ils sont définis à partir de données qualitatives. La seconde partie présente nos différentes contributions. Nous rappelons et étudions les liens entre les deux modèles. A ces fins, nous définissons formellement les treillis dichotomiques et étudions leurs propriétés. Puis nous introduisons la discrétisation, afin de généraliser aux treillis la discrétisation locale généralement utilisée pour les arbres. Enfin, la dernière étape du développement de notre modèle est formalisée, *i.e.* la réduction de la complexité structurelle.

I. État de l'art :

La première partie se décompose en trois chapitres. **Le chapitre 1 présente le contexte général de l'analyse de données.** Nous présentons d'abord les différents types de données qui sont utilisés dans ce domaine : données qualitatives et données quantitatives. Le formalisme et les notations qui seront utilisés dans ce manuscrit sont aussi présentés. Puis afin de mettre en évidence les avantages des modèles de classification symboliques, nous présentons aussi dans ce chapitre différents modèles de classification connus, leurs avantages et leurs inconvénients.

Le chapitre 2 présente, au travers d'une étude bibliographique, **les arbres de classification.** Nous commençons par rappeler l'algorithme général de construction des arbres de classification à partir de données qualitatives. Nous détaillons les deux phases de construction que sont l'expansion et l'élagage de l'arbre. Enfin nous présentons les principaux algorithmes d'induction d'arbres avant de nous intéresser aux différentes utilisations des arbres de classification dans un cadre de classification supervisée.

Le chapitre 3 porte sur le second modèle symbolique de classification qui nous intéresse dans cette thèse : **le treillis de Galois ou treillis de concepts**. Nous rappelons les définitions à la base de la théorie des treillis avant de nous intéresser plus particulièrement aux treillis de Galois. Nous donnons aussi différentes propriétés fondamentales des treillis qui nous seront utiles dans la suite de ces travaux, et en particulier les éléments irréductibles qui en décrivent la structure. Enfin, nous nous intéressons aux différentes utilisations du treillis de Galois dans un cadre de classification supervisée.

II. Vers un modèle hybride :

La seconde partie se décompose, elle aussi, en trois chapitres. **Le chapitre 4 est une présentation et une formalisation des liens existants entre les arbres de**

classification et les treillis de Galois lorsque ces modèles sont issus de mêmes données qualitatives. Tout d'abord, nous rappelons les similitudes entre les utilisations de ces modèles lors d'un processus de classification par navigation. Ensuite, nous démontrons le lien d'inclusion existant entre les deux structures, plus précisément nous montrons que **tout arbre de classification est inclus dans le treillis**. Puis, nous présentons **les treillis dichotomiques**, treillis issus de données qualitatives complémentaires, *i.e.* les modalités des attributs sont mutuellement exclusives, ce qui signifie qu'il est possible d'associer à tout attribut, un ensemble d'attributs complémentaires par rapport aux objets. En particulier, les tables de données qualitatives issues d'une discrétisation en intervalles disjoints, vérifient cette propriété. **Nous démontrons les propriétés de ces treillis particuliers c'est à dire leur $\vee$-complémentarité mais aussi leur coatomicité.** En effet, la complémentarité entre attributs a un impact sur la structure du treillis qui est $\vee$-complémenté. Par ailleurs, cette complémentarité est observée sur l'ensemble complet des objets, nous montrons que dans ces treillis, les inf-irréductibles ont pour successeur immédiat le concept maximal du treillis. Il s'agit donc des co-atomes du treillis, d'où la propriété de coatomicité. Ces propriétés sont la base des développements présentés dans la suite du manuscrit. Enfin, nous démontrons que **le treillis dichotomique est la fusion de tous les arbres possibles** construits à partir d'une même table de données qualitatives.

La définition du treillis de Galois repose sur l'existence d'une table de données binaires, appelée contexte. Le traitement des données quantitatives, comme le sont la plupart des signatures issues des images, n'est pas immédiat et nécessite généralement une étape de discrétisation menée, jusqu'à présent, de manière globale. Dans la littérature, les algorithmes de construction des arbres de classification ont majoritairement recours eux aussi à la discrétisation pour le traitement des données quantitatives. Cependant, cette discrétisation y est menée localement.

Le chapitre 5 présente le cadre général de la discrétisation. Cet outil est un modèle de transformation des données permettant leur passage du type quantitatif au type qualitatif. Il existe différents modes de discrétisation. Par une étude bibliographique, nous présentons les différents critères utiles à la discrétisation et leur formalisation. Nous explicitons les différents modes de discrétisation, leurs avantages et leurs inconvénients. Nous nous intéressons plus particulièrement à la discrétisation supervisée et aux différences existantes entre les deux types de discrétisation utilisées d'une part par les arbres de classification, et d'autre part par les treillis de Galois.

Puis, **nous présentons notre algorithme de discrétisation locale spécialement adapté à la structure de treillis, et basé sur des concepts spécifiques du treillis** : les concepts terminaux qui correspondent aux inf-irréductibles. Nous proposons deux modes de calcul du critère de gain pour cette discrétisation. Cet algorithme a pour objectif majeur d'améliorer les performances en classification des treillis, ce phénomène ayant été observé pour les arbres de classification, la discrétisation locale permettant de définir des arbres plus performants que la discrétisation globale. Nous démontrons

ensuite, par des expérimentations que cet algorithme permet non seulement d'améliorer les performances en classification des treillis mais aussi de diminuer leur complexité structurelle.

A la différence de l'arbre de classification, le treillis est connu pour avoir une taille importante (exponentielle en la taille de données analysées dans le pire des cas). Depuis quelques années différentes méthodes de réduction de complexité sont proposées. L'arbre de classification, selon l'algorithme utilisé, est doté d'une phase d'élagage permettant de réduire sa complexité mais aussi de diminuer le phénomène de sur-apprentissage.

Le chapitre 6 est une étude des simplifications possibles de la structure. Comme mentionné précédemment, nous étudions deux méthodes qui se distinguent par la monotonie du critère utilisé.

Tout d'abord, **nous proposons une généralisation des algorithmes d'élagage des arbres de classification aux treillis de Galois**. Basée sur une adaptation du critère considéré comme le plus performant pour les arbres de classification (la mesure de coût-complexité), cette généralisation est mise en œuvre. Nous démontrons alors que l'évolution de la structure du treillis au cours de l'élagage n'est pas similaire à celle des arbres. En effet, alors que cette mesure engendre une diminution du nombre de nœuds des arbres d'étape en étape, ce n'est pas toujours le cas pour les treillis, seul le nombre d'arcs diminue systématiquement. De ce fait, la robustesse des treillis face aux données bruitées est réduite, les expérimentations appuieront ces observations.

Nous nous orientons vers les simplifications existantes dans les treillis et plus particulièrement vers l'utilisation d'un critère non-monotone à l'inverse de celui précédemment utilisé. Ces simplifications ont recours à des indices définis localement sur les concepts du treillis. **Nous étudions plus particulièrement l'indice de stabilité**. Ce dernier étant non-monotone, il permet une simplification différente de celle mise en œuvre par la généralisation proposée précédemment. En effet, de par sa non-monotonie, les concepts peuvent être supprimés n'importe où dans la structure, ce qui permet d'obtenir une structure simplifiée qui ne vérifie plus la propriété de treillis à l'inverse des structures obtenues après simplification par un critère monotone. La simplification basée sur cet indice de stabilité nécessite la définition d'un seuil. Un tel paramétrage étant toujours complexe, **nous proposons un algorithme de définition automatique d'un seuil optimal.** Notre objectif étant alors de réduire la complexité du treillis obtenu, tout en conservant ses performances en classification, **nous présentons des résultats expérimentaux qui permettent de valider notre approche.**

Pour finir, dans la conclusion nous rappellerons les objectifs et les solutions proposées, avant de donner des perspectives de recherche et d'évolution du modèle hybride.

État de l'art

Chapitre 1

Analyse de données

En analyse de données [Bouroche 80], chaque donnée est représentée, décrite de manière différente selon le domaine d'étude et selon l'objectif de l'analyse. Les différences de représentation s'observent principalement sur le type des attributs utilisés. Nous nous intéressons aux différents types d'attributs existant en analyse de données dans la première section de ce chapitre. Les notations utilisées dans ce manuscrit y sont aussi présentées.

L'analyse de données est un vaste domaine de recherche qui regroupe différentes méthodes, chacune avec des objectifs différents. La deuxième section de ce chapitre introduit les différentes méthodes de l'analyse de données et leurs objectifs.

Ces travaux s'inscrivent plus particulièrement dans l'analyse de données issues d'images. Plus précisément, nous souhaitons créer un modèle symbolique de classification supervisée qui traite des données quantitatives issues de caractéristiques extraites des images. Pour motiver l'utilisation de modèles symboliques pour un tel traitement, nous décrivons dans la troisième et dernière section de ce chapitre, différents modèles de classification de la littérature. Nous mettons en avant leurs avantages et leurs inconvénients.

1.1 Représentation des données

Dans le domaine de l'analyse de données, l'ensemble des données à analyser est classiquement représenté sous forme d'une **table objets** $\times$ **attributs**. Chaque objet correspond à une ligne de la table et est décrit par un ensemble d'attributs. Chaque attribut représente une caractéristique de l'objet et correspond à une colonne de la table. Pour décrire un objet, il existe différents types d'attributs : **les attributs qualitatifs** et **les attributs quantitatifs** [Breiman 84, Tufféry 07, Tufféry 10].

Selon le domaine d'étude, il existe différentes terminologies :

- une donnée est aussi appelée observation, individu, enregistrement ou objet ;
- un attribut est aussi appelé caractéristique, descripteur ou champ ;

- un attribut dit quantitatif, en statistiques, est aussi appelé (dans le domaine informatique) attribut continu ou numérique ;
- un attribut dit qualitatif, en statistiques, est aussi appelé (dans le domaine informatique) attribut discret, nominal ou symbolique ;

Remarque. Il faut noter que dans les domaines mathématiques ou statistiques, un attribut quantitatif peut être continu (*i.e.* défini sur $\mathbb{R}$) ou discret (*i.e.* défini sur $\mathbb{Z}$). C'est pourquoi pour éviter toute ambiguïté, **nous choisissons d'utiliser la terminologie d'attribut quantitatif et d'attribut qualitatif** et non la terminologie attribut continu et attribut discret.

Quelque soit le type d'un attribut, l'ensemble des valeurs qu'il peut prendre est appelé **domaine** de définition de l'attribut.

1.1.1 Les différents types d'attributs

Les attributs qualitatifs

Définition 1. Un attribut est dit qualitatif lorsque les valeurs observées pour cet attribut ne sont pas des quantités. Les valeurs observées indiquent une qualité, exprimée généralement de manière alphabétique, mais qui dans certains cas peut être codée par des nombres.

Les attributs qualitatifs ont un domaine de définition qui leur est propre. Chaque valeur possible d'un attribut qualitatif est appelée **modalité** de l'attribut.

Quelques propriétés :
- Le nombre de modalités d'un attribut qualitatif est dénombrable.
- Lorsque toutes les modalités du domaine d'un attribut sont observées sur l'ensemble des objets, le jeu de données correspondant est dit **exhaustif** vis à vis de cet attribut. Sinon il est dit **non exhaustif** vis à vis de cet attribut.
- Certains attributs qualitatifs sont dits **binaires**, ou encore **booléens**, lorsqu'ils comptent uniquement deux modalités. Ces modalités sont 1, vrai ou $\times$ dans le cas où l'objet vérifie l'attribut ; 0, faux ou « » dans le cas inverse. Notons que tout attribut qualitatif peut être **binarisé**, *i.e.* chaque modalité de l'attribut devient un attribut binaire, la **binarisation** est aussi appelée codage disjonctif.

Les attributs qualitatifs peuvent être soit nominaux, soit ordinaux :
- attribut **ordinal** : les modalités de l'attribut peuvent être rangées les unes par rapport aux autres selon un ordre donné. Cependant, il est impossible de quantifier l'écart entre deux modalités différentes de cet attribut. Il est possible d'appliquer certains opérateurs arithmétiques/mathématiques à ce type d'attribut.
- attribut **nominal** : les modalités de l'attribut ne peuvent être rangées selon aucun ordre.

Exemple. Les attributs âge "mineur/majeur" et taille "S/M/L/XL" sont des **attributs qualitatifs ordinaux**. L'attribut couleur "rouge/vert/bleu" ou encore une adresse sont des **attributs qualitatifs nominaux**. Enfin l'attribut réussite "vrai/faux" est un **attribut qualitatif nominal binaire**.

Les attributs quantitatifs

Définition 2. Un attribut est dit quantitatif lorsqu'il exprime une quantité. Les valeurs observées pour cet attribut sont numériques, et il est possible d'appliquer des opérateurs arithmétiques sur ces valeurs.

Quelques propriétés :

- Les attributs quantitatifs peuvent être exprimés sur un domaine de valeurs **continues** (sur $\mathbb{R}$) ou **discrètes** (sur $\mathbb{N}$ ou $\mathbb{Z}$).
- La cardinalité du domaine peut être infinie.
- Il existe une relation d'ordre entre deux valeurs de l'attribut, et les écarts entre deux valeurs sont quantifiables.

Exemple. Les attributs salaire, température sont des **attributs quantitatifs** tout comme la plupart des attributs/caractéristiques extraites des images.

Remarque. Une des principales *différences entre un attribut quantitatif et un attribut qualitatif ordinal* réside dans la possibilité de faire des calculs tels qu'une moyenne des valeurs observées. Il est possible de faire ce type de calcul pour un attribut quantitatif, mais pas pour un attribut qualitatif ordinal.

La Table 1.1 présente un exemple de ces différents types d'attributs. La table 1.2 présente une binarisation de l'attribut *Couleur*.

1.1.2 Attributs explicatifs et à expliquer

L'un des objectifs de l'analyse de données peut être de construire un modèle capable d'estimer/prédire la valeur d'un attribut donné en fonction d'autres attributs. Dans ce cas, il est nécessaire de distinguer **les attributs explicatifs** (aussi appelés attributs prédictifs ou attributs exogènes) et **l'attribut à expliquer** (aussi appelé attribut à prédire ou attribut endogène). Les attributs explicatifs sont les attributs décrivant les objets. L'attribut à expliquer est l'attribut que le modèle devra estimer/prédire.

L'attribut à expliquer peut être soit qualitatif, soit quantitatif. **Dans cette thèse, nous nous intéressons principalement au premier cas**, **l'attribut à expliquer est alors appelé classe**, label, catégorie ou encore étiquette.

Ensemble d'objets	Ensemble d'attributs		
	att. qualitatif nominal	att. qualitatif ordinal	att. quantitatif
	Couleur	Taille	Épaisseur (mg/m^3)
Objet$_1$	Blanc	S	10
Objet$_2$	Bleu	M	15
Objet$_3$	Rouge	S	30.5
Objet$_4$	Jaune	XL	65
Objet$_5$	Blanc	L	25
Objet$_6$	Bleu	XXL	15.5
Objet$_7$	Jaune	M	25

TABLE 1.1: Exemple des différents types d'attributs

Ensemble d'objets	Ensemble d'attributs					
	att. binaires				att. binaires	att. quantitatif
	Couleur				Taille	Épaisseur (mg/m^3)
	Blanc	Bleu	Rouge	Jaune		
Objet$_1$	×				S	10
Objet$_2$		×			M	15
Objet$_3$			×	×	S	30.5
Objet$_4$					XL	65
Objet$_5$	×				L	25
Objet$_6$		×			XXL	15.5
Objet$_7$				×	M	25

TABLE 1.2: Exemple de binarisation

1.1.3 Cas des données images : attributs et notations

Images et attributs

Comme évoqué précédemment, les caractéristiques extraites des images sont généralement des attributs quantitatifs continus. L'ensemble de ces attributs est appelé vecteur de caractéristiques ou **signature de l'image**.

Il est possible d'extraire un grand nombre de caractéristiques d'une image. Le choix des descripteurs est généralement guidé par le type d'images traitées (images naturelles, images de documents, ...). Dans les articles de références [Trier 96, Loncaric 98, Yang 08], un point de vue général, sur la catégorisation des descripteurs, est proposé. En particulier, les descripteurs issus de calculs de moments géométriques et polynomiaux (moments invariants, moments de Zernike, ...) sont distingués de ceux issus de calculs de transformées (transformée de Fourier, transformée de Radon, ...) et de ceux dits locaux (*Shape context* [Belongie 02, Rusiñol 10], *Scale Invariant Feature Transform* SIFT [Lowe 99], ...).

Notons qu'en reconnaissance d'images, les signatures les plus pertinentes sont celles qui satisfont certaines contraintes (invariance en translation, rotation, et au changement d'échelle). De plus, ces signatures doivent être robustes au bruit (*i.e.* présence d'artéfacts dans l'image, ou de flou dans l'image, ...).

Dans la communauté document, d'où sont issues les images traitées au sein de notre laboratoire, il est courant de catégoriser les signatures soit selon leurs objectifs (description de la texture, des couleurs, des contours, ...) [Zuwala 06, Yang 08, Coustaty 11a], soit selon leur type (statistique ou structurel) [Lladós 02, Guillas 07]. Dans les paragraphes suivant, nous allons succinctement catégoriser quelques signatures selon leur type.

Signatures statistiques :

Les signatures statistiques sont des mesures calculées sur les pixels des images. Ces mesures sont généralement des fonctions mathématiques (moments géométriques et polynomiaux et transformées).

Parmi ces signatures, nous pouvons citer, pour les plus connues, les signatures basées sur des transformations telles que la transformée de Fourier [Chen 09], la transformée de Fourier-Mellin [Derrode 99, Singhal 09], la transformée de Radon [Radon 17] ou encore sur des moments géométriques, les moments invariants [Tzimiropoulos 09], les moments de Zernike [Teague 80], les moments de Hu [Hu 62].

Signatures structurelles :

Les signatures structurelles permettent de décrire l'agencement relatif de certaines primitives élémentaires constituant l'image (proches des descripteurs locaux). Les primitives élémentaires peuvent être soit des régions particulières de l'image, soit des primitives géométriques qui correspondent à des formes géométriques simples comme par exemple des droites, des cercles, des ellipses,... Ces primitives sont extraites au moyen de méthodes telles que la transformée de Hough [Hough 62] ou de Radon.

Puis, une signature quantitative est extraite de l'agencement de ces primitives entre elles (position spatiale, point d'intersection, angles, ...). Parmi les signatures structurelles se trouvent les approches proposées dans [Allen 83] ou encore celles appliquées aux images de documents [Rusiñol 06, Mas 10, Coustaty 11b].

La définition de ces caractéristiques est un domaine de recherche à part entière. Dans cette thèse, nous utilisons seulement des signatures existantes. Pour des définitions plus formelles, le lecteur peut se référer aux différents états de l'art [Trier 96, Loncaric 98, Lladós 02, Tabbone 03b, Guillas 07, Yang 08, Coustaty 11a].

Notations

Comme précisé précédemment, en analyse de données, **l'ensemble de données** est classiquement représenté sous forme d'une table où chaque objet correspond à une ligne et chaque attribut correspond à une colonne. Au sein d'une même table, les attributs peuvent être de type différent, comme illustré dans la Table 1.1. Lorsque les valeurs de l'attribut à expliquer sont connues pour un ensemble de données, alors ces données sont dites **étiquetées**.

La Table 1.3 présente un ensemble de huit images étiquetées, chaque image est représentée par cinq attributs explicatifs de type quantitatif et une classe (attribut à expliquer qualitatif). Cette table est un extrait d'un ensemble d'images que nous utilisons dans nos expérimentations [GREC 03] ; les images y sont décrites par une signature structurelle [Coustaty 11b].

De nombreuses méthodes d'analyse de données sont définies pour un seul type d'attribut. Pour pallier à ce problème, il existe des processus permettant de transformer les attributs quantitatifs en attributs qualitatifs et inversement. La transformation du type qualitatif au type quantitatif peut se faire *via* une Analyse des Correspondances Multiples ; nous renvoyons le lecteur à [Tufféry 10], pour plus de précisions. Pour notre part, nous nous intéresserons à la première transformation appelée **discrétisation**, dans le chapitre 5. Précisons juste qu'à partir d'un attribut quantitatif, la discrétisation définit

	Att_1	Att_2	Att_3	Att_4	Att_5	**Classe**
Im_1	1.46	2.82	0.62	-0.53	0.19	C1
Im_2	2.32	1.86	0.62	-0.53	0.19	C1
Im_3	0.32	1.22	-0.05	-0.53	-0.85	C2
Im_4	-0.81	-0.36	-0.05	-0.53	-0.85	C2
Im_5	-0.24	0.27	1.30	2.83	1.24	C3
Im_6	-0.24	-0.36	1.30	2.83	-0.85	C3
Im_7	0.03	-0.68	-0.74	-0.53	1.24	C4
Im_8	0.03	-0.68	-0.74	0.58	0.72	C4

TABLE 1.3: Exemple d'un ensemble d'images étiquetées

des intervalles de valeurs, chaque intervalle devenant alors une modalité de l'attribut qualitatif ordinal correspondant.

Ainsi dans ce manuscrit, il nous sera nécessaire de distinguer les attributs quantitatifs des attributs qualitatifs. De plus, nous nous intéressons à des modèles d'analyse de données permettant de déterminer, pour certains objets, la valeur d'un attribut à expliquer. Nous considérons uniquement les attributs à expliquer de **type qualitatif**.

Nous notons donc de manière générique les ensembles de données étiquetées sous la forme$\Omega = (O, \mathcal{Y}, K)$. Nous décomposons l'ensemble $\mathcal{Y}$ en deux sous-ensembles $\mathcal{V}$ et $\mathcal{I}$:
- $O = \{o_n, n \in \{1, \ldots, N\}\}$ est l'ensemble des objets ;
- $\mathcal{Y} = \mathcal{V} \cup \mathcal{I}$ est l'ensemble des attributs avec :
  - $\mathcal{V} = \{V_{.h},\ h \in \{1, \ldots, H\}\}$ est l'ensemble des attributs quantitatifs. Soit $V_{.h} \in \mathcal{V}$ un attribut quantitatif, les valeurs vérifiées pour cet attribut sont notées v_{nh} ($n \in \{1, \ldots, N\}$) ;
  - $\mathcal{I} = \{I_{.z},\ z \in \{1, \ldots, Z\}\}$ est l'ensemble des attributs qualitatifs. Soit $I_{.z}$ un attribut qualitatif les modalités vérifiées pour cet attribut sont notées m_{nz} .
- $K = \{k_{o_n}, n \in \{1, \ldots, N\}\}$ l'attribut à expliquer (de type qualitatif).

Pour simplifier les notations par la suite, nous considérons qu'un **objet o_n est représenté par le vecteur de ses attributs que nous noterons $\vec{o_n}$.**
Si un ensemble de données ne contient que des attributs explicatifs qualitatifs, il sera alors noté $\Omega = (O, \mathcal{I}, K)$ et s'il ne contient que des attributs explicatifs quantitatifs, il sera noté $\Omega = (O, \mathcal{V}, K)$.

L'ensemble des ces notations génériques est présenté dans la Table 1.4.

Cette première section présentait les différents types d'attributs utiles à la description de données et les notations que nous utilisons dans ce manuscrit pour nous y référer. Nous nous intéressons à l'analyse de ces données dans la section suivante.

Objets / Attributs	$\mathcal{Y}$										
O	$\mathcal{V}$					$\mathcal{I}$					K
	$V_{.1}$	...	$V_{.h}$	...	$V_{.H}$	$I_{.1}$	...	$I_{.z}$	...	$I_{.Z}$	
o_1	v_{11}		v_{1h}		v_{1H}	m_{11}		m_{1z}		m_{1Z}	k_{o_1}
$\vdots$			$\vdots$					$\vdots$			$\vdots$
o_n	v_{n1}	...	v_{nh}	...	v_{nH}	m_{n1}	...	m_{nz}	...	m_{nZ}	k_{o_n}
$\vdots$			$\vdots$					$\vdots$			$\vdots$
o_N	v_{N1}		v_{Nh}		v_{NH}	m_{N1}		m_{Nz}		m_{NZ}	k_{o_N}

TABLE 1.4: Notation générique d'un ensemble de données étiquetées

1.2 Les méthodes d'analyse de données

1.2.1 Terminologies : Analyse de données ou *data mining*

L'analyse de données a d'abord été définie pour traiter des ensembles de données de petites tailles, puis suite à l'émergence d'immenses ensembles de données le terme de *data mining* a été préféré [Saporta 06]. Le *data mining* est alors défini comme une étape d'un processus global appelé Extraction de Connaissances à partir de Données (ECD). Notons que selon Zighed et Rakotomalala [Zighed 02], *une confusion subsiste encore entre « data mining », que nous appelons en français « fouille de données » et « knowledge discovery in data bases » (KDD). Le « data mining » est l'un des maillons de la chaîne de traitement pour la découverte des connaissances à partir des données.*

L'ECD regroupe non seulement l'accès aux données, leur mise en forme (transformation des types de données par exemple), la sélection de certaines données si nécessaire (certains objets ou certains attributs), l'ajout de nouveaux attributs (moyenne, combinaison d'attributs existants), mais aussi la construction d'un modèle à partir des données et l'exploitation du modèle construit, voire la diffusion des résultats obtenus avec le modèle. Le *data mining* correspond alors au traitement allant de l'étape de mise en forme des données jusqu'à celle de l'exploitation du modèle.

La terminologie *data mining* étant liée à la gestion de grandes ensembles de données, **nous privilégions la terminologie analyse de données**.

Les méthodes d'analyse de données ont été introduites il y a longtemps, tout d'abord au début du 19ème siècle avec l'analyse canonique basée entre autres sur les travaux de Pearson [Pearson 01], suivis par ceux de Hotelling en 1933 avec les fondements de l'analyse en composante principale (ACP) [Hotelling 33a, Hotelling 33b]. Ces méthodes initiées dans le cadre de la statistique, ont été appliquées et enrichies dans le cadre informatique suite au développement des ordinateurs et à l'accumulation de données, entre autres au sein des entreprises [Saporta 06]. Les méthodes actuelles ont donc des origines

variées (statistique, informatique, bases de données) mais elles s'appliquent toutes à des tables de données (*cf.* section 1.1). Les analyses menées peuvent avoir des objectifs variés.

1.2.2 Objectifs

En statistique, il est courant de séparer les méthodes selon deux objectifs, l'objectif de description ou **d'exploration** et l'objectif inférentiel ou **décisionnel** [Bouroche 80, Saporta 90, Saporta 06].

En Informatique, généralement trois objectifs se distinguent, l'objectif de **description**, l'objectif de **structuration** et l'objectif de **prédiction** (ou explication). Les deux premiers sont inclus dans l'objectif d'exploration identifié en statistique et le dernier correspond à l'objectif décisionnel en statistique. En informatique, ces objectifs sont généralement liés à différents types **d'apprentissage** (*cf.* paragraphes suivants). L'apprentissage correspond à l'acquisition de savoir-faire, il est souvent référencé sous le terme **d'apprentissage automatique** en informatique (*i.e. machine learning* - ML). [Rakotomalala 97] : *L'apprentissage est ainsi le champ d'étude où l'on essaie de mimer et de reproduire la capacité de l'homme à apprendre.*

Nous nous plaçons du point de vue informatique pour expliciter ces différents objectifs et les méthodes associées.

Objectifs de description

En informatique, l'objectif de description consiste à synthétiser, à résumer l'information contenue dans un ensemble de données. Il s'agit de reformuler l'information sous une forme différente (moyenne d'un attribut âge, distribution des objets selon les modalités d'un attribut qualitatif, ...). Les **méthodes descriptives** mettent en évidence des informations présentes mais cachées dans les données [Tufféry 10].

Les méthodes descriptives issues de la statistique sont des méthodes d'analyse exploratoire. Parmi elles, les méthodes d'analyse factorielle qui permettent de représenter les données sous la forme d'un nuage de points dans un espace de dimension réduite. L'analyse factorielle est composée entre autre de l'Analyse en Composantes Principales (ACP) qui traite des données quantitatives, de l'Analyse Factorielle des Correspondances (AFC) traitant des données qualitatives mais aussi de l'Analyse des Correspondances Multiples (ACM) qui traite plutôt des données hétérogènes [Saporta 06].

Parmi les méthodes descriptives issues de l'informatique se trouvent les cubes de données définis avec les systèmes OLAP [Codd 93] ; ils permettent une représentation intuitive de l'information et sont très utilisés dans le domaine de l'aide à la décision.

Objectifs de structuration

L'objectif de structuration identifié en informatique, correspond à la **détection**, dans un ensemble de données, **de groupes d'objets homogènes** (*clusters*).

En statistique, ces méthodes sont des méthodes d'analyse exploratoire. Parmi elles, les méthodes de *clustering* telles que les **méthodes de partitionnement** (méthodes du type « nuées dynamiques » (*k-means*)) ou les **méthodes hiérarchiques** (classification ascendante hiérarchique CAH). Ces méthodes définissent les groupes selon des similarités entre attributs explicatifs, ces similarités étant parfois appelées « régularités » [Chandon 81].

En informatique, les méthodes de structuration sont les méthodes associées à **l'apprentissage non supervisé.** La terminologie « non supervisé » est liée au fait que les groupes ne sont pas connus *a priori*. Dans ce cadre, il est courant de parler de **classification non supervisée** pour le *clustering*. Parmi les méthodes de classification non supervisée, se trouvent les réseaux de neurones non supervisés telles que les cartes de Kohonen [Kohonen 82, Kohonen 01].

Objectifs de prédiction

L'objectif de **prédiction** est de transposer/extrapoler des informations présentes dans un ensemble de données à de nouvelles données. Les informations à transposer peuvent être soit qualitatives, soit quantitatives.

Les **méthodes prédictives** s'appliquent aux tables de données constituées d'attributs explicatifs et d'au moins un attribut à expliquer [Buhot 99].

En informatique, ces méthodes sont définies par **un apprentissage supervisé,** au sens où l'on dispose de connaissances *a priori* à propos de l'attribut à expliquer, au travers d'un ensemble d'exemples étiquetés. Parmi ces **méthodes prédictives** se trouvent les méthodes de **classification supervisée** où l'attribut à expliquer est de type qualitatif, et les **méthodes de régression** où l'attribut à expliquer est de type quantitatif.

Parmi les méthodes de classification supervisée, citons les modèles à base de **règles de décision** [Quinlan 79], ou encore les **machines à vecteurs de support** (SVM) [Vapnik 63] et **les arbres de classification** [Kass 80, Breiman 84, Quinlan 86a].

Parmi les nombreuses méthodes de régression [Saporta 90, Saporta 06], citons les modèles symboliques à base **d'arbres de régression** [Breiman 84].

Remarque. Le terme *classification* est porteur d'ambiguïté. En effet en **statistique**, il désigne les méthodes de *clustering* tandis que les termes de *discrimination* et *segmentation* sont préférés à celui de *classification supervisée*. En anglais, le terme de *classification* est réservé à la classification supervisée, la classification non supervisée étant appelée *clustering*. Distinguant les méthodes d'un point de vue informatique, **nous choisissons d'utiliser les terminologies de « classification supervisée » (ou simplement « classification ») et de « classification non supervisée »**.

Rappelons que les travaux présentés dans ce manuscrit ont pour objectif principal **la construction d'un modèle de classification supervisée** à partir de données issues

d'images étiquetées. Ce modèle doit classer de nouvelles images dans les classes connues *a priori*.

Cette section présentait les différents objectifs de l'analyse de données, les différentes terminologies associées et catégorisait certaines méthodes. La section suivante présente plus précisément la classification supervisée, les différentes qualités attendues d'un modèle de classification supervisée ainsi que les différents modèles de la littérature, leurs avantages et leurs inconvénients. Nous expliquons alors notre choix de nous orienter vers des modèles dits symboliques pour des données à analyser de type quantitatif.

1.3 Classification supervisée

La classification supervisée est un processus qui se décompose généralement en deux étapes (*cf.* Figure 1.3.1) [Zighed 02] :

1. **Étape d'apprentissage** (aussi appelée phase inductive) : **construction d'un modèle de classification par apprentissage des données** existantes (cadre horizontal et bleu dans la Figure 1.3.1) ;
2. **Étape de classification** (aussi appelée phase prédictive) : **utilisation du modèle construit pour la classification** de nouveaux objets (cadre vertical et rouge dans la Figure 1.3.1).

Ces deux étapes mettent en évidence d'un côté la phase d'apprentissage aboutissant à la construction d'un modèle de classification et de l'autre l'utilisation de ce modèle, **un même modèle pouvant être utilisé de plusieurs manières**.

Dans le cadre informatique, Breiman *et al.* donnent une définition générale de la classification en 1984 [Breiman 84], cette définition regroupe les deux catégories de classification (supervisée et non supervisée) : *Selon le problème, l'objectif de base d'une étude de classification peut être soit de produire un modèle de classification soit de découvrir la structure prédictive du problème.* Ils donnent aussi une définition mathématique qui illustre plutôt la classification supervisée :

Définition 3. Un modèle de classification est une fonction $\phi(.)$ définie sur $(O, \mathcal{Y})$ telle que pour tout $(o, \vec{o}) \in (O, \mathcal{Y})$, $\phi((o, \vec{o}))$ est égale à l'une des classes de l'ensemble K.

$$\begin{aligned} \phi : \quad (O, \mathcal{Y}) &\rightarrow \quad K \\ (o, \vec{o}) &\rightarrow \quad k_o \end{aligned}$$

La Table 1.5 présente un exemple de données standard et générique issu des problématiques bancaires, et traité par des modèles de classification supervisée. Le modèle construit doit répondre à la question d'accord de prêt pour un nouveau client. Les attributs explicatifs sont l'âge, le loyer et les revenus, et l'attribut à expliquer est l'accord de prêt.

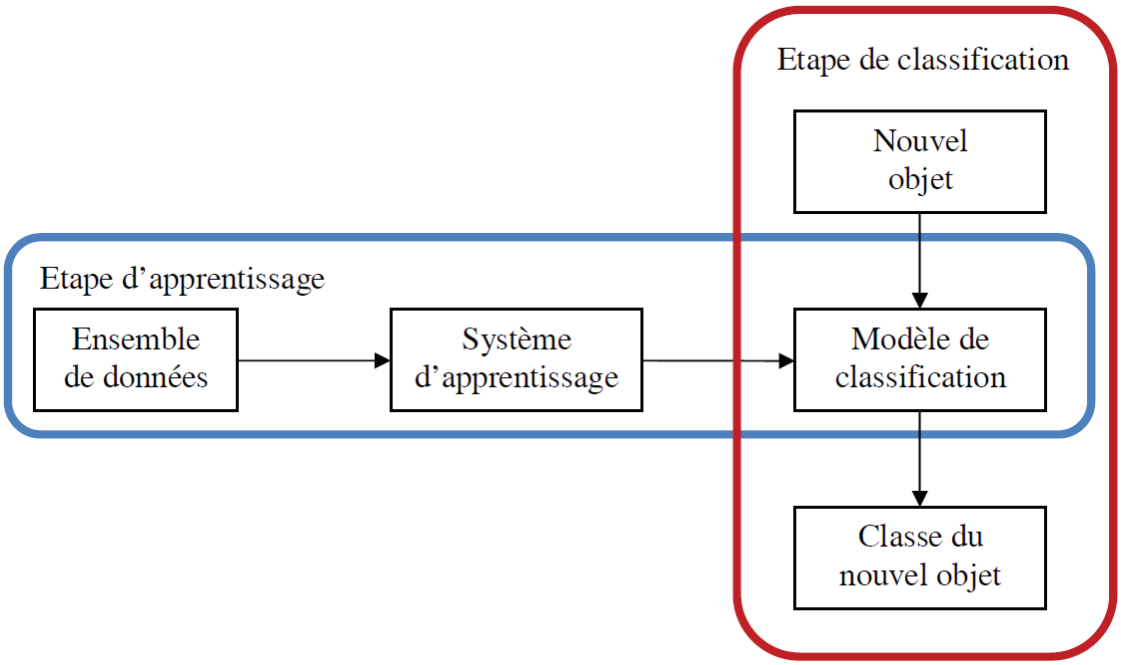


FIGURE 1.3.1: Les deux étapes de la classification supervisée

Ensemble de clients				
Id_Client	**Age**	**Loyer**	**Revenus**	**Prêt accordé ?**
1	36	0	1299	oui
2	50	240	1282	non
3	40	768	1394	oui
4	39	0	1018	oui
5	44	334	512	non
6	26	631	722	non

Nouveau client à classer				
Id_Client	**Age**	**Loyer**	**Revenus**	**Prêt accordé ?**
7	30	260	1100	?

TABLE 1.5: Exemple de données à traiter par un modèle de classification

Lors de la définition d'un modèle de classification supervisée, l'objectif final est de le transmettre à des utilisateurs pour faciliter leurs recherches et leurs décisions. Dans l'exemple précédent, le modèle de classification est défini pour classer les clients selon leur capacité à rembourser un prêt. Le modèle est alors construit à partir d'un ensemble de clients issus de l'ensemble de clients de la banque et ayant déjà fait une demande de prêt. Une fois le modèle construit, il est diffusé pour aider les agents à valider ou non les demandes de prêts des nouveaux clients. Pour éviter tout risque, il est clair que la banque souhaite utiliser **un modèle performant**, *i.e.* **un modèle qui commet peu d'erreur de prédiction**. De plus, l'agent a souvent besoin de justifier les refus de prêt, pour cela il doit comprendre le résultat donné par le modèle, aussi **le modèle doit être lisible** pour des non experts des modèles de classification. Pour faire le choix du meilleur modèle, la banque en compare plusieurs et choisit celui qui répond le mieux à ses attentes.

Cette démarche de comparaison de performances est courante. En informatique, lors de la définition ou de l'amélioration de modèle, il est nécessaire de s'appuyer sur une comparaison avec d'autres modèles pour démontrer l'apport du nouveau modèle. Cette comparaison se base sur plusieurs mesures que nous décrivons dans la section suivante.

1.3.1 Évaluation de la qualité d'un modèle de classification

Les modèles construits via un apprentissage, tels que les modèles de classification, sont généralement évalués selon deux critères : leur **taux d'erreur** (ou de manière similaire leur taux de reconnaissance) et leur **complexité** (structurelle et algorithmique).

Taux d'erreur

Lorsqu'un modèle de classification est construit, il dépend des données utilisées pour sa construction. Il correspond alors à une fonction $\tilde{\phi}((O, \mathcal{Y})) = \tilde{K}$ qui approxime le modèle exact $\phi((O, \mathcal{Y})) = K$. **L'objectif est de construire un modèle le plus précis possible** - *i.e.* $\tilde{K} = K$. Cependant, il n'est pas possible de connaître le taux d'erreur qu'un modèle aura sur des données inconnues au moment de sa construction. Alors pour évaluer ce taux, il faut appliquer le modèle à un ensemble de données étiquetées et vérifier s'il classe correctement les objets de cet ensemble. Pour obtenir une évaluation la plus réaliste possible, la construction du modèle ne se fait pas à partir de la totalité des données étiquetées à disposition, mais seulement à partir d'une fraction de celle-ci appelée **base d'apprentissage.** Le taux d'erreur est alors évalué sur l'ensemble des données étiquetées restantes appelé **base de test**.

Pour la définition de certains modèles, l'ajustement de paramètres est nécessaire, en particulier pour éviter le phénomène de **sur-apprentissage** (ou sur-ajustement). Pour ce faire, une troisième base est définie, la **base de validation** qui est un extrait de la base d'apprentissage.

Le sur-apprentissage (*overfitting* en anglais) correspond au fait que le modèle construit a appris trop précisément les données de la base d'apprentissage. Le modèle est alors capable de classer ces données de manière exacte (**Spécialisation**). En revanche, il est peu performant, voire incapable de classer des données n'appartenant pas à cette base d'apprentissage (**Généralisation**). Ce phénomène est en général provoqué par un mauvais dimensionnement du modèle construit tel qu'illustré dans les Figures 1.3.2, 1.3.3, 1.3.4 (extraites de [Markowetz 03]). Ces Figures représentent un ensemble de données projetées dans un espace 2D, réparties en deux classes (les ronds rouges, les carrés verts), le modèle construit (le trait noir) et un objet à classer (le carré « ? » jaune). La Figure 1.3.2 illustre le sous-apprentissage, avec un modèle trop simple, *i.e.* ce modèle ne sera pas performant aussi bien pour classer les données apprises que pour classer d'autres données tel que l'objet « ? ». La Figure 1.3.4 illustre le sur-apprentissage avec un modèle trop complexe, *i.e.* ce modèle collant aux données apprises, il sera performant pour classer ces données mais pas pour classer d'autres données tel que l'objet « ? ». Enfin la Figure 1.3.3 illustre le « bon » modèle, celui présentant **le meilleur compromis entre ajustement** (performant sur les données apprises) **et généralisation** (performant sur d'autres données). La base de validation permet d'ajuster les paramètres et la taille d'un modèle pour trouver ce compromis.

Ainsi, il est donc important de différencier trois bases distinctes constituées de **données indépendantes et identiquement distribuées** issues de l'ensemble des données à disposition :

1. La **base d'apprentissage (BA)** pour la construction du modèle. Cette base va permettre au modèle **d'apprendre l'agencement** des données. (Spécialisation)
2. La **base de validation (BV)** pour l'ajustement des paramètres du modèle. Cette base va permettre d'ajuster les paramètres et d'éviter le **sur-apprentissage** en classification. (Généralisation)
3. La **base de test (BT)** pour évaluer le taux d'erreur du modèle. (Évaluation)

Les bases d'apprentissage et de validation sont associées à l'étape d'apprentissage, et la base de test à celle de classification (*cf.* Figure 1.3.1).

Mesurer le taux d'erreur d'un modèle $\phi(.)$ sur un ensemble de Γ objets correspond à compter **le nombre d'objets mal classés** par le modèle. Ce taux d'erreur se calcule généralement à partir de **la matrice de confusion** (table 1.6).

Cette matrice donne les effectifs d'objets par classe d'affectation (*i.e.* trouvée par le modèle) en colonne et par classe d'origine en ligne. Ainsi n_{ij} désigne le nombre d'objets de la classe k_i affectés à tort à la classe k_j.

Nous notons le nombre d'objets mal classés :

$$MC(\phi(.), BT) = \sum_{i=1}^{\mathcal{H}} \sum_{\substack{j=1 \\ j \neq i}}^{\mathcal{H}} n_{ij} = |\{(o_i, \vec{o_i}) \in BT \text{ tel que } \phi((o_i, \vec{o_i})) \neq k_{o_i}\}|$$

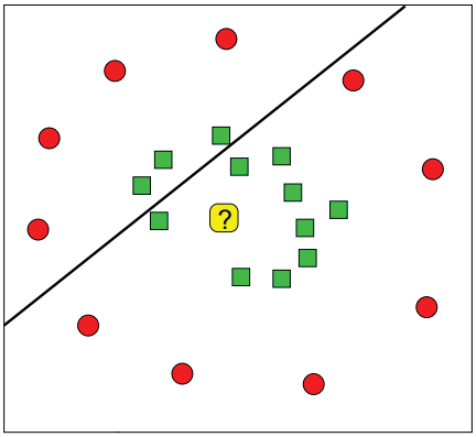


FIGURE 1.3.2: Sous-apprentissage - modèle trop simple

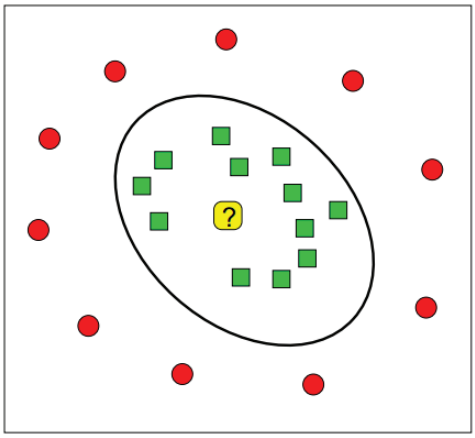


FIGURE 1.3.3: Bon apprentissage - modèle généraliste

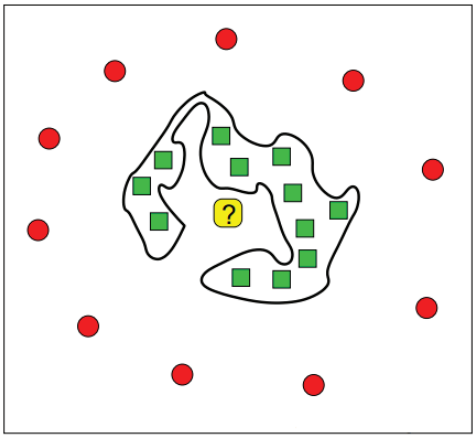


FIGURE 1.3.4: Sur-apprentissage - modèle trop complexe

		Classe d'affectation				
		$\tilde{k_1}$	...	$\tilde{k_j}$	...	$\tilde{k_\mathcal{H}}$
Classe d'origine	k_1	n_{11}		n_{1j}		$n_{1\mathcal{H}}$
	$\vdots$			$\vdots$		
	k_i	n_{i1}	...	n_{ij}	...	$n_{i\mathcal{H}}$
	$\vdots$			$\vdots$		
	$k_\mathcal{H}$	$n_{\mathcal{H}1}$		$n_{\mathcal{H}j}$		$n_{\mathcal{H}\mathcal{H}}$

TABLE 1.6: Matrice de confusion

Définition 4. Le taux d'erreur du modèle $\phi(.)$ sur la base BT est $\varepsilon(\phi(.), BT) = \frac{MC(\phi(.),BT)}{\Gamma}$.

Remarque. De manière équivalente, il est possible d'utiliser **le taux de reconnaissance**, *i.e.* taux défini à partir du nombre d'objets bien classés.

Dans la littérature, certains modèles utilisent le taux d'erreur sur la base d'apprentissage, appelée **erreur en resubstitution** que nous noterons $\varepsilon(\phi(.), BA)$. Néanmoins, ce taux d'erreur est en général trop optimiste. Lors de l'ajustement des paramètres d'un modèle, il est courant d'utiliser le taux d'erreur sur la base de validation que nous noterons $\varepsilon(\phi(.), BV)$.

La répartition en deux bases à partir d'un ensemble de données étiquetées se fait fréquemment selon les ratios 2/3-1/3, c'est à dire que 2/3 des objets de l'ensemble à disposition sont utilisés comme base d'apprentissage et 1/3 comme base de test. Si une base de validation est nécessaire, 1/3 des objets est réservé pour la base de validation. Cette répartition n'a pas de justification théorique *a priori*, de plus lorsque le nombre d'objets de l'ensemble à disposition n'est pas très grand [1], l'exclusion d'un tiers des objets pour l'étape d'apprentissage est reconnue pour être très pénalisant [Breiman 84, Quinlan 86b].

Pour pallier à ce problème, une autre méthodologie est souvent préférée, elle est issue de **la validation croisée** (*cross-validation*). Dans ce cadre, l'ensemble de données à disposition est divisé en plusieurs paquets. Puis, chaque paquet est utilisé une fois en tant que base de validation pour un modèle construit à partir de la base d'apprentissage constituée des autres paquets. Généralement, 10 paquets sont définis (*10-fold cross-validation*), ainsi chaque paquet contient 10% des objets de l'ensemble à disposition. A partir de ces paquets, dix bases apprentissage sont constituées, chacune avec 9 paquets soit 90% de l'ensemble à disposition et à chacune des bases d'apprentissage est associé à une base de validation, *i.e.* le 10éme paquet ($BV_i, i \in \{1, \ldots, 10\}$).

1. Grand ~ 20 000 objets pour Breiman *et al.*

Initialement la validation croisée a été définie dans le cadre statistique pour ajuster les paramètres des modèles. Comme mentionné précédemment, pour pallier aux évaluations biaisées, elle est souvent utilisée pour évaluer les performances des modèles en termes de généralisation. Dans ce cas, la répartition se fait entre base d'apprentissage et base de test.

Le modèle est alors construit sur chaque base d'apprentissage et testé sur chaque base de test associé ($BT_i, i \in \{1, \ldots, 10\}$). Ainsi le modèle est associé à dix taux d'erreur, le taux d'erreur « global » du modèle est alors définie comme la moyenne de ces taux d'erreur, *i.e.* $\varepsilon(\phi(.)) = \frac{1}{10}\sum_{i=1}^{10}\varepsilon(\phi(.), BT_i)$ (*cf.* Figure 1.3.5). Généralement, à ce taux d'erreur est associée la **variance** : $\sigma_\varepsilon^2 = \frac{1}{10}\sum_{i=1}^{10}(\varepsilon(\phi(.), BT_i) - \varepsilon(\phi(.)))^2$.

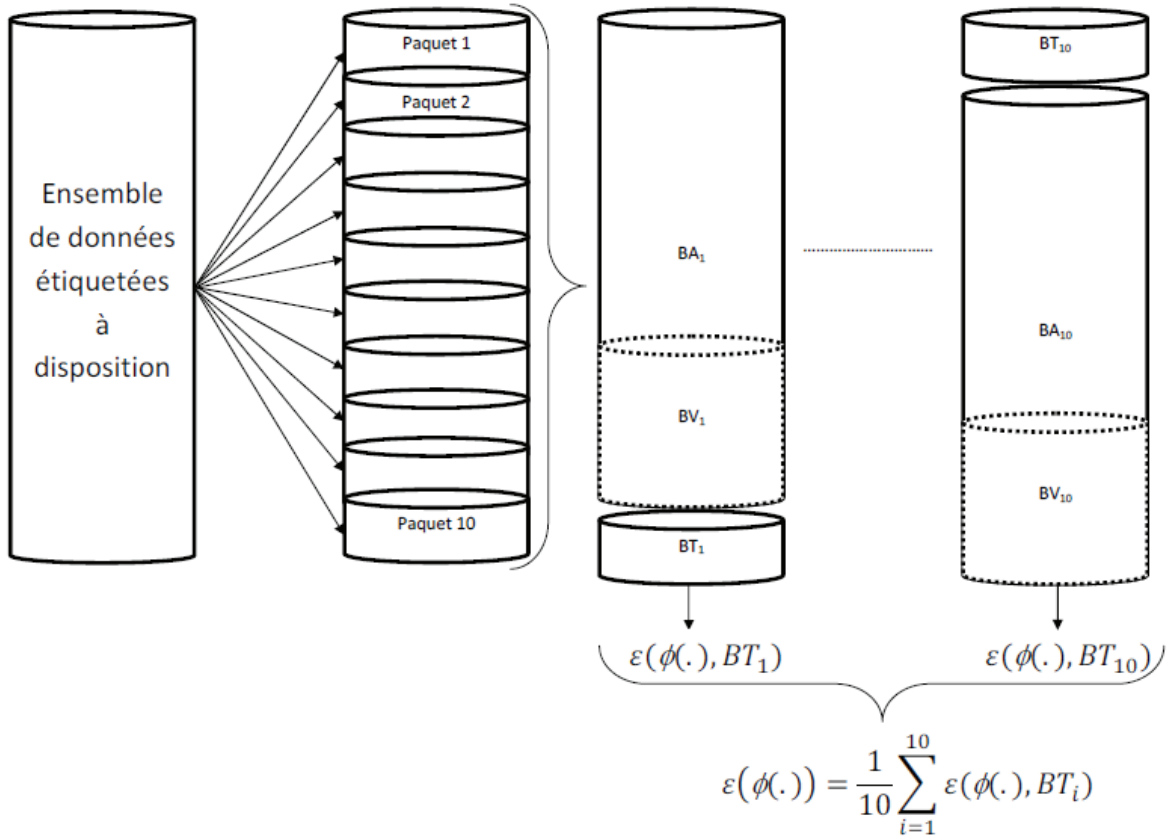


FIGURE 1.3.5: Méthodologie de la validation croisée à 10 paquets

Au cours de nos expérimentations (*cf.* chapitres 5,6), nous avons privilégié cette méthodologie avec une validation croisée à 10 paquets.

Complexité structurelle et algorithmique

Il faut distinguer deux formes de complexité, la complexité structurelle d'un modèle (nombre de nœuds d'un arbre par exemple), en lien avec la lisibilité et les capacités de stockage et la complexité algorithmique (*i.e.* temps d'exécution nécessaire au calcul du modèle), en lien avec la rapidité.

Au delà de l'ajustement et de la généralisation, la **complexité structurelle,** *i.e.* **la lisibilité** du modèle construit est le second objectif le plus fréquemment recherché. Dans sa thèse [Rakotomalala 97], Rakotomalala décrit parmi les qualités désirées d'un classifieur l'aspect *compréhensibilité,* en particulier pour un objectif d'explication. En effet il rappelle à juste titre que *prédire la classe d'un individu est une chose, expliquer pourquoi* [cette classe a été prédite] *en est une autre.* Comme nous le verrons dans la suite, il existe de nombreux modèles réputés et performants mais souvent décrits comme des « boîtes noires », du fait que l'explication du résultat obtenu n'est pas directe. La complexité structurelle du modèle est l'un des facteurs de lisibilité d'un modèle, nous reviendrons sur ce sujet dans les parties suivantes.

Enfin, malgré la montée en puissance des ordinateurs, les ensembles de données se complexifiant, **la complexité algorithmique**, *i.e.* **la rapidité** de construction d'un modèle (rapidité d'apprentissage) mais aussi sa rapidité en utilisation restent des facteurs à prendre en considération. Selon le domaine d'étude, il est parfois nécessaire d'obtenir le résultat d'une classification dans des délais très courts, aussi le modèle utilisé devra être rapide.

Selon ces deux critères, le meilleur modèle est défini comme celui offrant le meilleur compromis entre faible taux d'erreur, lisibilité et rapidité.

1.3.2 Les types de modèles

Selon le type de données étudiées, il existe trois grandes familles de classifieurs, les classifieurs probabilistes, les classifieurs statistiques et les classifieurs symboliques, chacun ayant ses caractéristiques.

1.3.2.1 Les classifieurs probabilistes

Les classifieurs probabilistes s'appliquent généralement aux données décrites par des attributs qualitatifs nominaux et utilisent la probabilité qu'un objet appartienne à une classe connaissant sa description, *i.e.* les valeurs des attributs explicatifs pour cet objet.

Pour inférer la classe d'un objet, il existe alors différentes règles présentées de manière non-exhaustive ci-dessous.

La règle du maximum de vraisemblance :

Cette règle consiste à assigner à tout nouvel objet o la classe k^* telle que la probabilité $p(o/k)$ qu'un objet de cette classe ait pour description celle de o est maximale. Dans ce cas $k^* = \underset{k \in K}{argmax}(p(o/k))$, les probabilités $p(o/k)$ étant estimées sur les données de la base d'apprentissage.

Notons que cette règle est utilisée parfois par d'autres modèles, où elle est alors appliquée à des sous-ensembles d'objets.

La règle de Bayes d'erreur minimale :

Cette règle consiste à assigner à tout nouvel objet o la classe k^* telle que la probabilité $p(k/o)$ est maximale. La classe est donc définie par $k^* = \underset{k \in K}{argmax}(p(k/o))$ où $p(k/o)$ est appelée **probabilité *a posteriori***. Cette règle est basée sur le théorème de Bayes [Bayes 63] : $p(k/o) = \frac{p(o/k) * p(k)}{p(o)}$ calculée sur les données de la base d'apprentissage.

La formule du théorème de Bayes permet de voir que ce dernier transforme la probabilité *a priori* d'une classe k ($p(k)$), en probabilité *a posteriori via* la description de l'objet o.

Il a été démontré que la règle de Bayes est optimale au sens de l'erreur de classification [Duda 73]. Cependant la principale critique faite envers cette méthode dans la littérature porte sur la difficulté d'estimation des probabilités *a posteriori* qui sont généralement inconnues. De fait, elle est alors souvent citée comme une référence théorique non praticable.

Malgré cela, cette règle est la base d'un autre modèle très souvent cité, le **classifieur bayésien naïf** qui correspond à une 'simplification' des probabilités à estimer. La simplification apportée réside principalement dans l'hypothèse d'indépendance des attributs deux à deux. Ce classifieur est non optimal à l'inverse de la règle de Bayes mais il a pour avantages sa simplicité et ses bonnes performances [Rish 01]. Cependant, son traitement de **données quantitatives continues** nécessite le calcul des densités de probabilités ou une phase préalable de discrétisation. Nous nous intéressons à cette transformation dans le chapitre 5.

Ainsi les avantages reconnus de ces méthodes sont leur simplicité et leurs bons résultats dans des cas simples. Leurs inconvénients sont leurs difficultés de mise en œuvre pour le traitement de données quantitatives continues, le fait qu'elles se heurtent à la représentativité des données dans la base d'apprentissage (il est difficile de classer un nouvel objet dont la description n'existe pas dans la base d'apprentissage) et l'hypothèse forte faite sur l'indépendance des attributs explicatifs. Enfin, **le résultat obtenu est peu lisible dans le sens où il est difficile d'estimer le poids de tel ou tel attribut dans une décision**. Ce dernier point est d'autant plus important pour des utilisateurs non spécialistes des probabilités.

1.3.2.2 Les classifieurs statistiques

Les classifieurs statistiques s'appliquent aux données décrites généralement par des attributs quantitatifs ou qualitatifs ordinaux. Ces modèles reposent sur l'analyse des données de la base d'apprentissage, *i.e.* analyse de la distribution des données dans la base d'apprentissage. Le principe général de ces modèles est la séparation de l'espace par des frontières telles que des hyperplans calculés selon la répartition des données dans l'espace, et selon la représentation fréquentielle de ces données. La classification d'un nouvel objet est alors fonction de sa disposition par rapport aux frontières construites. Parmi les modèles statistiques fréquemment utilisés se trouvent les réseaux de neurones [McCulloch 88, Bishop 95], mais les plus souvent cités et utilisés restent les **machines à vecteurs de supports** (Support Vector Machines-SVM) aussi appelées séparateurs à vaste marge. Le paragraphe suivant présente ce dernier modèle que nous utilisons lors de nos expérimentations. Pour les autres méthodes, nous renvoyons le lecteur aux références citées ci-dessus.

Les machines à vecteurs de supports :

Les SVM sont une généralisation des travaux introduits en 1963 dans l'article [Vapnik 63] par Vapnik et Lerner. Dans la littérature ces travaux sont souvent associés aux travaux de Vapnik sur la théorie de l'apprentissage [Vapnik 95, Vapnik 98]. Parmi les articles introduisant ces modèles on peut citer entre autre ceux de Boser *et al.* [Boser 92] et de Cortes *et al.* [Cortes 95]. Notons les articles de Guermeur et Paugam-Moisy [Guermeur 99] et de Burges [Burges 98] qui reprennent le raisonnement de Vapnik dans son ensemble de la théorie de l'apprentissage avec la définition de la VC-dimension [2] et de la VC-entropie [3] [Vapnik 71] jusqu'à la définition des SVM. Pour plus de précisions sur ces mesures nous renvoyons le lecteur aux références [Vapnik 71, Burges 98, Guermeur 99, Saporta 06].

Les SVM s'appliquent uniquement aux attributs quantitatifs et ont été définis pour des problèmes binaires, *i.e.* les données sont réparties dans deux classes distinctes. Cependant, il est possible de traiter des problèmes multi-classes *via* un processus de vote majoritaire. Dans ce cas, plusieurs SVM sont construites, séparant les classes deux à deux. Lors de la classification d'un nouvel objet, la classe choisie sera celle qui emporte le plus de voix au sens des classifications par chacune des SVM construites [Guermeur 07].

Dans le **cas de données d'apprentissage linéairement séparables** (*i.e.* les fron-

2. La VC-dimension (dimension de Vapnik-Chervonenkis) correspond au nombre d'objets pouvant toujours être séparés par une famille de classifieurs.

3. La VC-entropie quant à elle correspond aux nombres de séparations différentes existantes pour un ensemble de données.

tières existantes entre les objets de classes différentes sont linéaires), l'objectif des SVM est de définir le meilleur hyperplan ou **l'hyperplan séparateur optimal** qui séparera les données de classes différentes. La Figure 1.3.6 présente un ensemble de données réparties en deux classes k_0 et k_1 linéairement séparables. Quelques hyperplans séparateurs candidats sont représentés. L'hyperplan optimal est défini comme l'hyperplan qui **maximise la marge** aux objets les plus proches. En reprenant l'exemple précédent, le meilleur hyperplan est présenté dans la Figure 1.3.7. Les points les plus proches de l'hyperplan optimal sont appelés **points supports ou vecteurs supports**.

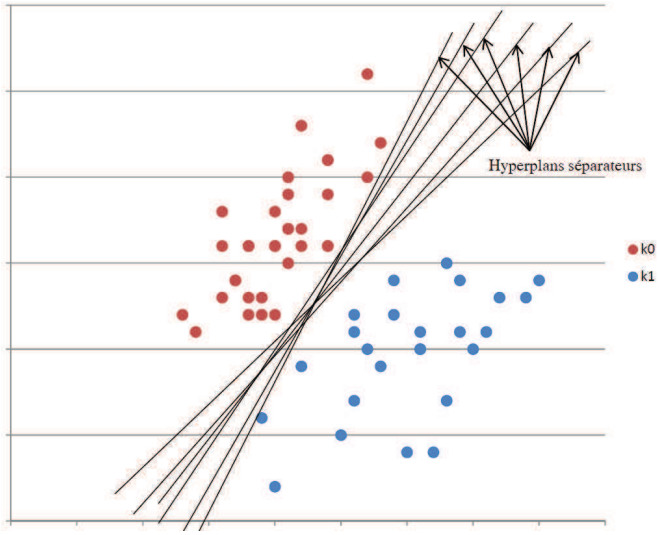


FIGURE 1.3.6: Exemple de données linéairement séparables et d'hyperplans séparateurs

Dans le cas de **données non linéairement séparables**, il est possible de **projeter les données d'apprentissage dans un espace de plus grande dimension** que celle de l'espace d'origine. Cette transformation est nommée le *kernel trick* ou astuce du noyau et a été introduit dans l'article [Boser 92]. Ce nouvel espace, noté $\mathcal{F}$, est muni d'un produit scalaire et peut être de dimension infinie. La projection des données dans ce nouvel espace est une transformation non linéaire $\Phi : \mathcal{V} \rightarrow \mathcal{F}$. Dans ce nouvel espace, il est alors possible de trouver un hyperplan séparateur. L'hyperplan séparateur optimal est défini de la même manière que dans les cas linéairement séparables, sans nécessiter une représentation complète des données dans le nouvel espace, mais uniquement en utilisant des **fonctions noyaux** appliquées aux données de l'espace d'origine. Il existe de nombreuses fonctions noyaux, les plus connues sont les noyaux polynomiaux de différents degrés, le noyau Gaussien, le noyau sigmoïde, ou encore la fonction à base radiale (RBF) [Cortes 95, Guermeur 99, Saporta 06]. Les fonctions noyaux jouent un rôle très important dans les performances de la SVM construite. Le noyau choisi doit coïncider avec la 'forme' des données dans leur espace d'origine. La Figure 1.3.8 illustre cette trans-

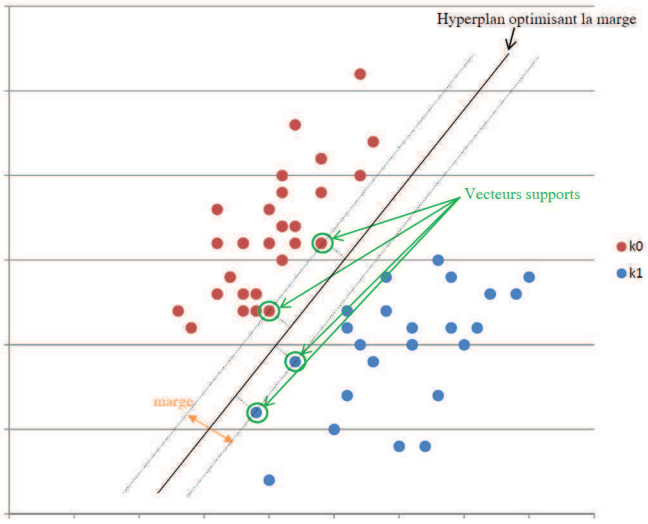


FIGURE 1.3.7: Hyperplan séparateur optimisant la marge

formation, la séparation des données dans l'espace d'origine (image de gauche) n'est pas évidente ; la projection des données dans un espace de dimension supérieure (image de droite) permet d'identifier plus facilement l'hyperplan optimal.

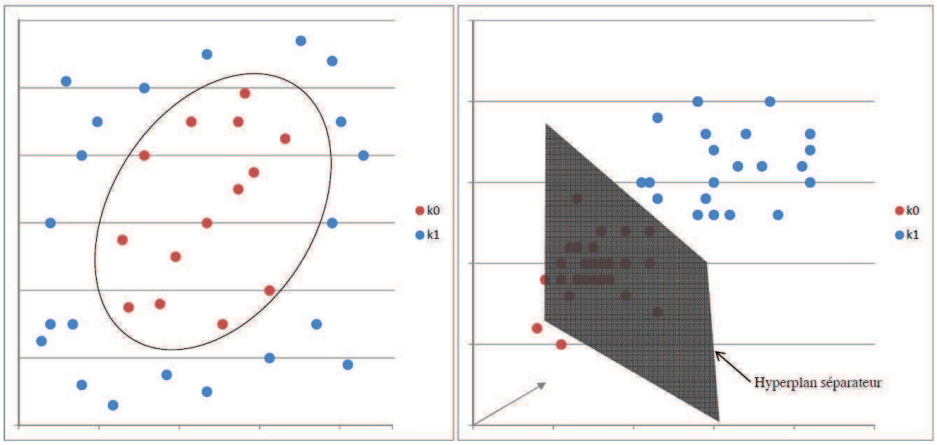


FIGURE 1.3.8: Utilisation d'une fonction noyau pour retranscrire le problème dans un espace de dimension supérieure

Les SVM sont des méthodes réputées en classification supervisée pour les données quantitatives. Ils sont souvent mis en avant en raison de leurs très bonnes performances. Cependant, il faut choisir le bon noyau et les bons paramètres, en fonction des données. Cela n'est pas forcement évident pour un utilisateur non spécialiste [Cornuéjols 02]. Dans

la littérature le noyau le plus souvent recommandé est la fonction à base radiale (RBF). Quelques travaux tentent de définir comment choisir le meilleur noyau mais ces études sont souvent liées aux domaines d'applications [Sahak 09, Ben Ayed Mezghani 10].

Les méthodes statistiques sont souvent paramétriques, et le choix de ces paramètres peut parfois influencer les résultats obtenus. De plus ces résultats sont, comme pour les méthodes probabilistes, difficilement interprétables, il n'est pas possible de connaître le poids de chaque attribut dans une décision. Ainsi ces méthodes sont souvent comparées à des « boîtes noires » [Saporta 06].

1.3.2.3 Les classifieurs symboliques

Les modèles symboliques s'appliquent généralement aux données décrites par des attributs qualitatifs. Grâce à un processus de **discrétisation**, ils peuvent aussi s'appliquer aux données décrites par des attributs quantitatifs. Les valeurs des attributs quantitatifs sont alors regroupées en intervalles qui deviennent des modalités d'attributs qualitatifs. Nous présenterons ce processus dans le chapitre 5. Parmi les modèles symboliques, nous pouvons citer les systèmes de règles de classification ou encore d'association [Quinlan 79, Niblett 87, Oosthuizen 88, Sahami 95, Antonie 02b, Antonie 02a, Wang 05, Bouzouita 06], les arbres de classification [Kass 80, Breiman 84, Quinlan 86a, Quinlan 93] et les treillis de Galois [Birkhoff 40, Barbut 70, Wille 82]. Ces modèles déterminent généralement la classe d'un nouvel objet par validations successives de ses attributs explicatifs.

Système de règles

Un système de règles est un ensemble de règles de type *prémisse→conclusion*. Pour la classification supervisée, la conclusion correspond à la classe et la prémisse correspond à l'association de plusieurs modalités issues de différents attributs qualitatifs. Il existe dans la littérature de nombreuses méthodes pour définir un système de règles, il est d'ailleurs fréquent d'extraire les règles à partir d'autres modèles tels que les arbres de classification [Quinlan 79, Quinlan 87a, Quinlan 93] ou les treillis de Galois [Oosthuizen 88, Sahami 95, Antonie 02b, Antonie 02a, Wang 05, Bouzouita 06].

Prenons un exemple simple pour illustrer la classification supervisée par système de règles. Soit la Table 1.7, cette table permet de déterminer l'activité d'une journée selon les conditions météorologiques. La classe ou l'attribut à expliquer est l'attribut 'Activité'.

Le système de règles qui pourrait être extrait est le suivant :
- (R1) 'Temps = Pluie' → 'Rester à la maison'
- (R2) 'Temps = Soleil' et 'Vagues = entre 1m50 et 2m50' → 'Aller surfer'
- (R3) 'Temps = Soleil' et 'Vagues = < 1m50' et 'Marée = Montante' → 'Aller surfer'

	Temps	Vagues	Marée	Activité
o_1	Soleil	$<$ 1m50	Montante	Aller surfer
o_2	Soleil	$<$ 1m50	Descendante	Rester à la maison
o_3	Soleil	entre 1m50 et 2m50	Descendante	Aller surfer
o_4	Soleil	$>$ 2m50	Montante	Rester à la maison
o_5	Pluie	$<$ 2m50	Descendante	Rester à la maison
o_6	Soleil	$<$ 1m50	Montante	? ?

TABLE 1.7: Table de données qualitatives pour définir des règles

- (R4) 'Temps = Soleil' et 'Vagues = $<$ 1m50' et 'Marée = Descendante $\rightarrow$ 'Rester à la maison'
- (R5) 'Temps = Soleil' et 'Vagues = $>$ 2m50' $\rightarrow$ 'Rester à la maison'

L'utilisation de ce système de règles pour la nouvelle donnée o_6 correspond à valider les valeurs de ses attributs explicatifs : 'Temps = Soleil', (règles retenues (R2) à (R5)), puis l'attribut 'Vagues = $<$ 1m50' (règles retenues (R3) et (R4)), enfin 'Marée = Montante', la règle retenue est (R3) l'attribut à expliquer 'Activité', de la donnée o_6, est alors prédit comme 'Aller surfer'.

A l'instar des modèles précédents (probabilistes et statistiques), les modèles symboliques ont une **lisibilité** bien plus importante. En effet, ces derniers permettent une compréhension des résultats beaucoup plus directe ; dans la classification de l'objet o_6 par exemple, la règle appliquée met en évidence les attributs qui ont conduit à la classe choisie, ce qui n'est pas le cas dans les autres modèles. De plus, **les modèles symboliques** n'utilisent pas d'hypothèse sur la distribution des objets à classer, à l'inverse des modèles statistiques et probabilistes ; en ce sens ils **sont non paramétriques** [Cornuéjols 10]. Enfin ils peuvent s'appliquer à tous types de données (*via* une discrétisation) alors que les modèles probabilistes ne peuvent pas traiter les données quantitatives.

1.4 Conclusion

Dans ce premier chapitre, nous avons présenté les notations et terminologies que nous avons choisies pour représenter un ensemble de données, avant de nous intéresser aux différents modèles d'analyse de données existants dans la littérature. Nous avons focalisé notre attention sur les modèles de classification supervisée, qui constituent l'objet principal de cette thèse.

Comme mentionné précédemment, nous souhaitons faire de la classification d'images. D'un point de vue modèle de classification, la lisibilité des résultats procurés par les modèles symboliques, encourage à leur utilisation. Les données extraites des images sont

généralement quantitatives, or les modèles symboliques ont été définis initialement pour le traitement de données qualitatives.

Aussi, **la problématique majeure de cette thèse est la définition d'un modèle symbolique pour le traitement de données quantitatives.** Les performances des treillis de Galois en classification d'images, ainsi que les liens forts entre arbres de classification et les treillis de Galois, ayant été mis en avant dans [Guillas 07] ; **le modèle défini devra être hybride entre ces deux modèles, *i.e.* utiliser les avantages de chacun.**

Les chapitres suivants présentent les deux modèles symboliques étudiés dans cette thèse : les arbres de classification et les treillis de Galois. Nous introduisons ces modèles tout d'abord lorsqu'ils sont utilisés pour traiter des données qualitatives, puis nous nous intéressons à leur traitement des données quantitatives dans le chapitre 5. Dans le chapitre 6, nous nous intéressons aux simplifications possibles du modèle construit.

Points clés

Positionnement

- ❏ Nous avons précisé les notations que nous utilisons pour désigner les ensembles de données et les différentes bases nécessaires à la définition d'un modèle de classification.
- ❏ Nous avons rappelé les objectifs de l'analyse de données et nous nous sommes placés dans le cadre de la classification supervisée.
- ❏ Les principes fondamentaux de la classification supervisée ont été mis en avant. Les qualités désirées pour les modèles construits ont été expliquées.
- ❏ Les différents types de modèles ont été présentés.
- ❏ Nous avons justifié notre choix d'utiliser des modèles symboliques pour le traitement de données quantitatives et présenté les objectifs de cette thèse.

Chapitre 2

Arbres de classification

2.1 Introduction

L'arbre de décision (aussi appelé graphe d'induction) est un modèle symbolique utilisé depuis de nombreuses années dans les tâches d'apprentissage automatique. On distingue deux types d'arbres de décision :

- Les **arbres de régression** où l'attribut à expliquer est de type quantitatif.
- Les **arbres de classification** où l'attribut à expliquer est de type qualitatif.

Morgan et Sonquist [Morgan 63] ayant construit les premiers des **arbres de régression**, les statisticiens leur attribuent la paternité de ce domaine de recherche. La famille des **arbres de classification** a été développée après les arbres de régression toujours sous l'impulsion de Sonquist avec en particulier l'algorithme AID [Sonquist 71]. Ensuite différents algorithmes d'induction ont été développés tels que ID3 [Quinlan 79, Quinlan 86a] défini par Quinlan en 1979, puis ChAID [Kass 80] défini par Kass en 1980, suivi de CART [Breiman 84] défini par Breiman dans son livre réputé en 1984. Puis, en 1993, Quinlan apporte des améliorations à son algorithme ID3 qu'il fait alors évoluer vers l'algorithme C4.5 [Quinlan 93] (traitement des données quantitatives et post-élagage). Ces algorithmes sont parmi les plus populaires encore aujourd'hui, mais il en existe bien d'autres.
De nombreuses études ont été réalisées pour comparer les performances de chaque arbre [Dougherty 95, Quinlan 93, Rakotomalala 05a], mais aussi pour proposer de nouveaux critères nécessaires à la construction de ceux-ci [Zighed 96, Zighed 00, Do 10], ou encore pour améliorer leurs performances [Quinlan 93, Breiman 96a, Quinlan 96b].

Les arbres de décision sont connus pour leur **simplicité** aussi bien en termes de construction qu'en termes d'utilisation. Ils offrent une **lisibilité** et une **facilité de compréhension** qui n'existent que très rarement dans les modèles d'analyse de données. Pour notre part, nous nous intéressons plus particulièrement à la famille des arbres de classification. Dans la suite de ce chapitre, nous présentons l'algorithme générique de

construction des arbres avant de nous intéresser plus précisément aux différents modèles connus et aux différences existantes entre ceux-ci.

2.2 Construction

2.2.1 Structure & représentation

Un arbre de classification [Rakotomalala 05a] est une arborescence constituée de **nœuds** et **d'arcs**.

Le nœud origine de l'arbre est appelé **racine**, les nœuds finaux sont appelés **feuilles** et les nœuds intermédiaires sont appelés **nœuds internes**. La Figure 2.2.1 présente un arbre construit à partir de la Table 2.1. La racine y est présentée sous forme de diamant, les feuilles sous forme de rectangles et les nœuds internes sous forme d'ellipses. Chaque nœud est étiqueté avec l'ensemble des objets qu'il contient et un ensemble d'attributs. La racine contient tous les objets de la base d'apprentissage. Les feuilles sont également étiquetées avec la classe à laquelle elles correspondent.

Les arcs (aussi appelés branches) représentent le lien existant entre deux nœuds, il est courant de parler de **successeurs** ou fils d'un nœud pour représenter la relation descendante et de **prédécesseur** ou père pour représenter la relation ascendante. Chaque arc est associé à un **test comparant un attribut à une valeur** particulière de son domaine. Par exemple sur la Figure 2.2.1, la racine a trois nœuds successeurs chacun associé à un test différent mais portant sur le même attribut. Le premier test est *Prévisions = ensoleillé* donc seuls les objets contenus dans la racine et vérifiant ce test appartiendront au nœud successeur correspondant. Le deuxième test est *Prévisions=couvert* et le dernier est *Prévisions=pluie*. Ainsi, tous les objets de la racine sont répartis dans ses successeurs. Cette répartition dépend de la valeur que chaque objet vérifie pour l'attribut *Prévisions*.

Ainsi, l'ensemble d'attributs qui étiquète un nœud correspond à l'ensemble des attributs des arcs menant à ce nœud depuis la racine. Les objets du nœud vérifient cet ensemble d'attributs.

Dans la littérature, **les représentations graphiques** des arbres de classification sont diverses ; il est courant de trouver des arbres où les nœuds sont étiquetés avec la distribution en classe des objets qu'ils contiennent ; et les arcs sont étiquetés avec le test associant un attribut à une valeur test [Kass 80, Breiman 84, Rakotomalala 97]. Parfois les attributs tests étiquettent les nœuds internes alors que les valeurs tests étiquettent les arcs et les feuilles sont étiquetées par la classe majoritaire des objets qu'elles contiennent ; dans ce cas les objets ne sont pas représentés [Quinlan 86a].

Pour des raisons de lisibilité et pour faciliter notre comparaison aux treillis de Galois dans les chapitres suivants, nous choisissons, dans la majorités des exemples didactiques de ce manuscrit, **d'étiqueter les nœuds avec la liste des objets**

qu'ils contiennent et la liste des attributs validés jusqu'à chaque nœud ; les arcs sont étiquetés par les tests. La classe associée aux feuilles est donnée dans leur étiquette (*cf.* Figure 2.2.1).

Dans la section 2.4, pour illustrer le processus d'élagage d'un arbre, ainsi que les mesures d'évaluation associée, **nous représenterons aussi les arbres en étiquetant les nœuds avec la répartition en classes des objets qu'ils contiennent** (*cf.* Figure 2.2.2 équivalente à la Figure 2.2.1**).**

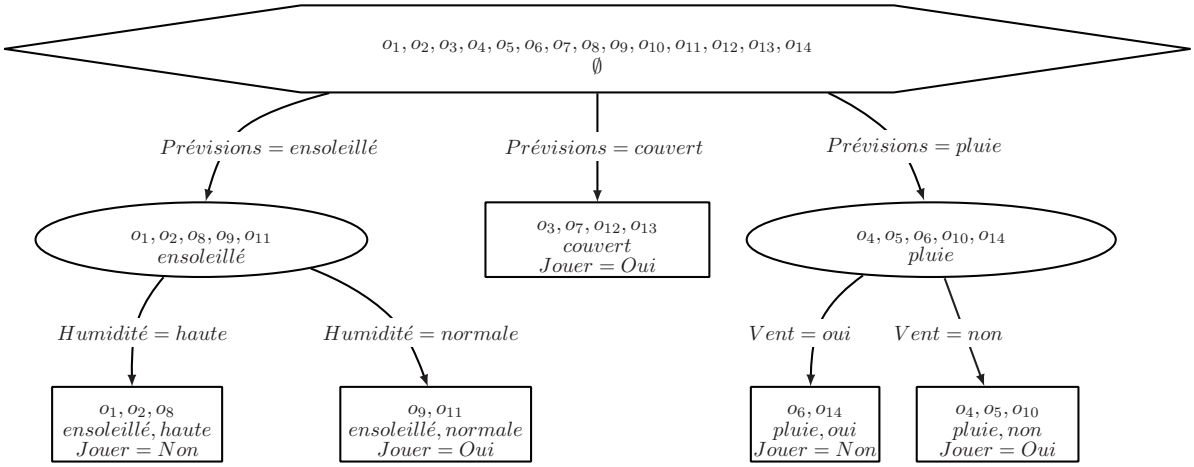


FIGURE 2.2.1: Exemple d'arbre de classification

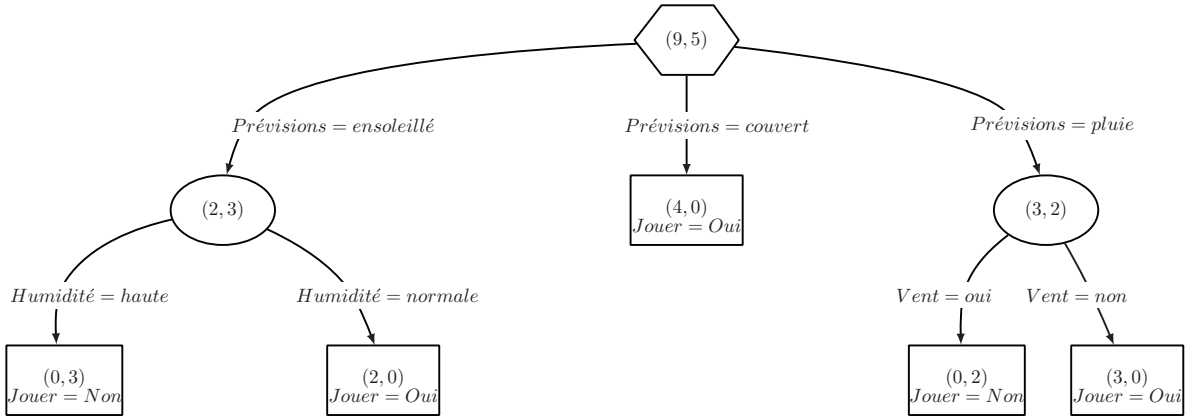


FIGURE 2.2.2: Représentation des arbres selon la répartition des objets en classes

2.2.2 Algorithme générique

Les arbres de classification sont construits à partir de la racine, qui regroupe l'ensemble des objets, jusqu'aux feuilles. Cette construction est dite **top-down**, *i.e.* du haut

vers le bas ou de la racine vers les feuilles [Rokach 08]. **Pour simplifier la lecture des algorithmes de construction, nous choisissons de les présenter pour le traitement d'attributs qualitatifs**, *i.e.* $\Omega = (O, \mathcal{I}, K)$. Le traitement des données quantitatives sera abordé au chapitre 5. Cette construction dépend de deux critères :

1. **Le critère de division** encore appelé critère de segmentation ou de coupe. Appliqué à un ensemble de données, il indique l'attribut qui permet la meilleure séparation de ces objets. Lors du traitement de données quantitatives, il indique aussi le point de coupe optimal associé à la séparation (*cf.* chapitre 5)
2. **Le critère d'arrêt** qui permet de stopper la construction de la branche courante.

La construction de l'arbre peut se limiter à une *phase descendante* : **la division** avec ou sans pré-élagage, mais elle peut aussi avoir une *phase dite ascendante* : **le post-élagage**. Le post-élagage est présenté dans la section 2.4.

Dans la littérature, il est courant de présenter l'algorithme générique de construction des arbres de classification à partir d'un exemple, nous ne dérogerons pas à cette règle. Avant de présenter plus formellement les deux phases de construction, nous reprenons l'exemple célèbre utilisé par Quinlan dans son article [Quinlan 86a]. Cet exemple consiste en la construction d'un arbre de classification à partir d'une base d'apprentissage constituée de 14 objets décrits par 4 attributs prédictifs de type qualitatif nominal et un attribut à expliquer de type qualitatif binaire. Cette base est présentée dans la Table 2.1.

La construction de l'arbre à partir de cette base va consister à trouver un partitionnement des objets de telle sorte qu'il sépare les objets de classes différentes et les répartisse dans différentes feuilles. Le processus de construction d'un arbre de classification correspond à **un algorithme récursif**. Le processus est composé de trois étapes simples que sont :

1. *Le choix de l'attribut de test*
2. *La construction des successeurs*
3. *L'étiquetage des feuilles*

En suivant la méthode ID3 de Quinlan, l'arbre construit est celui présenté par la Figure 2.2.1. Les étapes de construction sont les suivantes :

- Choix de l'attribut de test : Soit le nœud racine contenant l'ensemble des objets. Selon **le critère de division**, l'attribut *Prévisions* est choisi comme étant le meilleur pour séparer les objets de la racine : il est le plus discriminant ;
- Construction des successeurs : Les nœuds successeurs de la racine sont construits, **un successeur par valeur possible pour l'attribut test** *Prévisions* **selon les objets contenus** dans la racine soit trois nœuds, un pour *ensoleillé*, un pour *couvert* et le dernier pour *pluie*. Les objets de la racine sont ensuite répartis dans ces successeurs selon la valeur qu'ils vérifient pour l'attribut test ;

O	$\mathcal{I}$				Classe
	Prévisions	Température	Humidité	Vent	Jouer ?
o_1	ensoleillé	chaud	haute	non	Non
o_2	ensoleillé	chaud	haute	oui	Non
o_3	couvert	chaud	haute	non	Oui
o_4	pluie	doux	haute	non	Oui
o_5	pluie	frais	normale	non	Oui
o_6	pluie	frais	normale	oui	Non
o_7	couvert	frais	normale	oui	Oui
o_8	ensoleillé	doux	haute	non	Non
o_9	ensoleillé	frais	normale	non	Oui
o_{10}	pluie	doux	normale	non	Oui
o_{11}	ensoleillé	doux	normale	oui	Oui
o_{12}	couvert	doux	haute	oui	Oui
o_{13}	couvert	chaud	normale	non	Oui
o_{14}	pluie	doux	haute	oui	Non

TABLE 2.1: Exemple de données pour la construction d'un arbre de classification

– Étiquetage des feuilles : Les nœuds construits sont alors évalués pour vérifier s'ils satisfont **le critère d'arrêt**. Ici, le nœud correspondant au test *Prévisions=couvert* ne contient que des objets appartenant à la classe *Jouer=Oui*, il satisfait le critère d'arrêt, il est alors considéré comme une feuille. Les deux autres nœuds ne satisfont pas le critère d'arrêt, **le processus de division est réitéré sur ces nœuds**.

– Choix de l'attribut de test : Soit le nœud correspondant au test *Prévisions=ensoleillé* et contenant les objets $\{o_1, o_2, o_8, o_9, o_{11}\}$. Selon le critère de division, l'attribut *Humidité* est choisi comme étant le meilleur pour séparer ces objets ;
– Construction des successeurs : Les valeurs possibles pour l'attribut *Humidité* selon les objets $\{o_1, o_2, o_8, o_9, o_{11}\}$ sont *haute* et *normale* donc deux successeurs sont construits, un pour chacune d'elle. Les objets $\{o_1, o_2, o_8, o_9, o_{11}\}$ sont ensuite répartis dans ces successeurs selon la valeur qu'ils vérifient pour l'attribut test ;
– Étiquetage des feuilles : Les nœuds construits sont évalués pour vérifier s'ils satisfont le critère d'arrêt. Ici, le nœud correspondant au test *Humidité=haute* contient des objets appartenant à la classe *Jouer=Non*, il satisfait le critère d'arrêt, il est alors considéré comme une feuille et étiqueté avec la classe *Jouer=Non*. Le second nœud correspondant à *Humidité=normale* contient des objets appartenant à la classe *Jouer=Oui*, il satisfait le critère d'arrêt, il est lui aussi considéré comme une feuille et étiqueté avec la classe *Jouer=Oui*. Les deux nœuds construits étant des feuilles, le processus de division ne peut être réitéré dans cette branche, il est maintenant appliqué aux nœuds qui n'ont pas encore été traités ; soit le nœud conte-

nant les objets $\{o_4, o_5, o_6, o_{10}, o_{14}\}$.

- Choix de l'attribut de test : Soit le nœud correspondant au test *Prévisions=pluie* et contenant les objets $\{o_4, o_5, o_6, o_{10}, o_{14}\}$. Selon le critère de division, l'attribut *Vent* est choisi comme étant le meilleur pour séparer ces objets ;
- Construction des successeurs : Les valeurs possibles pour l'attribut *Vent* selon les objets $\{o_4, o_5, o_6, o_{10}, o_{14}\}$ sont *oui* et *non* donc deux successeurs sont construits, un pour chacune d'elle. Les objets $\{o_4, o_5, o_6, o_{10}, o_{14}\}$ sont ensuite répartis dans ces successeurs selon la valeur qu'ils vérifient pour l'attribut test ;
- Étiquetage des feuilles : Les nœuds construits sont alors évalués pour vérifier s'ils satisfont le critère d'arrêt. Ici, le nœud correspondant au test *Vent=oui* contient des objets appartenant à la classe *Jouer=Non*, il satisfait le critère d'arrêt, il est alors considéré comme une feuille et étiqueté avec la classe *Jouer=Non*. Le second nœud correspondant à *Vent=non* contient des objets appartenant à la classe *Jouer=Oui*, il satisfait le critère d'arrêt, il est lui aussi considéré comme une feuille et étiqueté avec la classe *Jouer=Oui*. Les deux nœuds construits étant des feuilles, le processus de division ne peut être réitéré dans cette branche. Tous les nœuds ayant été traités, la phase de division est terminée.

Ainsi, la construction des arbres de classification dépend de fonctions simples présentées ci-dessous :

Fonctions basiques :
- créer_nœud(Ω') : construction d'un nœud contenant l'ensemble de données étiquetées Ω' ;
- créer_feuille($\mathcal{N}$,DT) : transforme le nœud $\mathcal{N}$ en feuille dans l'arbre DT en l'étiquetant avec la classe majoritairement représentée ;
- Meilleur_Arbre_Élagué(DT,BA,BV) : applique un processus de post-élagage à l'arbre DT, en utilisant la base d'apprentissage étiquetée BA et/ou la base de validation étiquetée BV et renvoie l'arbre élagué (cf. section 2.4) ;

L'algorithme générique de construction est décomposé en deux fonctions présentées dans les Algorithmes 2.1 et 2.2. L'Algorithme 2.1 correspond à l'appel initial utilisant une base d'apprentissage, une base de validation, un critère de division et un critère d'arrêt. Il crée le nœud racine avant d'appeler l'Algorithme 2.2 pour construire l'arbre, puis il appelle une méthode de post-élagage (ce dernier appel n'est pas obligatoire, *cf.* section 2.4). L'Algorithme 2.2 est le cœur du processus, *i.e.* la **génération récursive** des successeurs d'un nœud passé en paramètre, cet algorithme utilise aussi le critère de division et le critère d'arrêt.

Algorithme 2.1 Créer_Arbre(BA,BV,$gain$,$\mathcal{S}$)

Entrées :
- BA la base d'apprentissage, ensemble de données étiquetées $BA = (O,\mathcal{I},K)$;
- BV la base de validation (ensemble de données étiquetées) ;
- $gain$ le critère de division ;
- $\mathcal{S}$ le critère d'arrêt ;

Sorties :
- DT un arbre de classification avec pour racine le noeud contenant l'ensemble des données de la base d'apprentissage BA ;

DÉBUT

$\mathcal{N}$ = créer_noeud(BA) ;
DT = Construire_Arbre($\mathcal{N}$,$gain$,$\mathcal{S}$) ;

renvoyer Meilleur_Arbre_Élagué(DT,BA,BV) ; //post-élagage

FIN

2.2.3 Arbres binaires et M-aires

Lorsque tous les nœuds internes d'un arbre ont exactement deux successeurs, alors l'arbre est appelé **arbre binaire**. Notons que certains algorithmes imposent la génération d'arbres binaires. Dans ce cas, si un attribut qualitatif à plus de deux modalités est choisi pour la division, ses modalités sont regroupées en deux ensembles, chacun d'eux est associé à l'une des deux branches créées (*cf.* algorithme CART section 2.5).

Lorsque dans un arbre, il peut exister des nœuds avec plus de deux successeurs, cet arbre est appelé **arbre M-aire**.

Remarque 1. Comme illustré par notre exemple, il peut y avoir **plusieurs feuilles étiquetées par la même classe**. Cela est dû au fait qu'une classe peut être associée à différentes combinaisons 'attribut-valeur'. Chaque combinaison correspond à un chemin différent dans l'arbre. Le chemin de la racine à une feuille est unique et définit une combinaison particulière. Dans notre exemple, la classe *Jouer=Non* est associée à la combinaison (*i.e.* au chemin) *{Prévisions=ensoleillé, Humidité=haute},* mais aussi à la combinaison *{Prévisions=pluie, Vent=vrai},* etc.

Suite à cette présentation de l'algorithme générique de construction des arbres de classification, intéressons nous de manière plus formelle aux critères associés, de division et d'arrêt.

Algorithme 2.2 Construire_Arbre($\mathcal{N}$,*gain*,$\mathcal{S}$)

Entrées :
- $\mathcal{N}$ un nœud contenant un ensemble de données $\Omega_{\mathcal{N}} = (O_{\mathcal{N}},\mathcal{I}_{\mathcal{N}},K_{\mathcal{N}}) \subseteq BA$;
- *gain* le critère de division ;
- $\mathcal{S}$ le critère d'arrêt ;

Sorties :
- DT un arbre de classification ayant pour racine $\mathcal{N}$;

DÉBUT

$DT \leftarrow \mathcal{N}$;
si $\mathcal{N}$ satisfait $\mathcal{S}$ **alors**
  créer_feuille($\mathcal{N}$,DT) ;
sinon
  Sélectionner parmi $\mathcal{I}_{\mathcal{N}}$ le meilleur attribut $I^*_{\mathcal{N}}$ selon le critère de division *gain* ;
  pour chaque Modalité m_l de $I^*_{\mathcal{N}}$ **faire**
    Ω_l = sous-ensemble de données de $\Omega_{\mathcal{N}}$ vérifiant m_l ;
    $\mathcal{N}_l$ = créer_noeud(Ω_l) ;
    $sous\_arbre_l$ = Construire_Arbre($\mathcal{N}_l$,*gain*,$\mathcal{S}$) ;
    Ajouter dans DT le sous-arbre $sous\_arbre_l$ comme successeur de $\mathcal{N}$;
  fin pour
fin si

renvoyer DT ;

FIN

2.3 La division

La phase de division correspond à la *phase descendante* de la construction d'un arbre. Elle dépend de deux critères que sont le **critère de division** et le **critère d'arrêt.** Pour simplifier les notations et la présentation formelle des critères, **nous traitons les attributs qualitatifs dans cette partie, le problème des attributs quantitatifs sera abordé dans le chapitre** 5.

Les critères de division et d'arrêt sont calculés pour un ensemble de données et selon la répartition de ces données dans les classes. Voici quelques notations utiles à la définition de ces critères.

Soit un nœud $\mathcal{N}$ contenant un ensemble de données étiquetées $\Omega_{\mathcal{N}} = (O_{\mathcal{N}}, \mathcal{I}_{\mathcal{N}}, K_{\mathcal{N}})$ avec $\Omega_{\mathcal{N}} \subseteq \Omega = BA$, et $O_{\mathcal{N}} = \{o_1, o_2, \ldots, o_N\}$ est l'ensemble de N objets décrits par Z attributs qualitatifs $\mathcal{I}_{\mathcal{N}} = \{I_{.1}, I_{.2}, \ldots, I_{.Z}\}$. Les objets de $\Omega_{\mathcal{N}}$ appartiennent à différentes classes, l'ensemble $K_{\mathcal{N}} = \{q_1, q_2, \ldots, q_H\} \subseteq K$ couvrent ces classes, *i.e.* $\forall o_i \in O_{\mathcal{N}}\ \exists h \in \{1, \ldots, H\}$ tel que $k_{o_i} = q_h$. La Table 2.2 rappelle ces notations.

	$\mathcal{I}_{\mathcal{N}}$					$K_{\mathcal{N}}$
$O_{\mathcal{N}}$	$I_{.1}$	$\ldots$	$I_{.z}$	$\ldots$	$I_{.Z}$	
o_1	m_{11}		m_{1z}		m_{1Z}	k_{o_1}
$\vdots$			$\vdots$			$\vdots$
o_n	m_{n1}	$\ldots$	m_{nz}	$\ldots$	m_{nZ}	k_{o_n}
$\vdots$			$\vdots$			$\vdots$
o_N	m_{N1}		m_{Nz}		m_{NZ}	k_{o_N}

TABLE 2.2: Table des données associée à un nœud

Pour calculer ces critères de division *gain* et d'arrêt $\mathcal{S}$, il faut connaître la répartition en classes des objets de $O_{\mathcal{N}}$ selon les différentes valeurs de chaque attribut $I \in \mathcal{I}_{\mathcal{N}}$. Pour cela, une **table de contingence** est construite en croisant l'attribut classe $K_{\mathcal{N}}$ avec un attribut $I \in \mathcal{I}_{\mathcal{N}}$, aussi appelé attribut analysé. Notons que dans la littérature, la construction de cette table est aussi appelée *tri croisé*.

Considérons un attribut $I \in \mathcal{I}_{\mathcal{N}}$, prenant ses valeurs dans l'ensemble $\{m_1, m_2, \ldots, m_L\}$ (*i.e.* $\forall n \in \{1, \ldots, N\}\ m_{nz} \in \{m_1, m_2, \ldots, m_L\}$). En considérant les classes $K_{\mathcal{N}} = \{q_1, q_2, \ldots, q_H\}$ observées pour l'ensemble $O_{\mathcal{N}}$, alors la table de contingence croisant I et $K_{\mathcal{N}}$, présentée dans la Table 2.3, est définie comme suit :

- Chaque classe $q_h \in \{q_1, q_2, \ldots, q_H\}$ correspond à une ligne de la table de contingence ;

- Chaque valeur $m_l \in \{m_1, m_2, \ldots, m_L\}$ correspond à une colonne de la table de contingence ;
- Les cases internes de la table contiennent les effectifs croisant une classe et une valeur particulière de l'attribut I ; c'est à dire n_{hl} est le nombre d'objets de $O_{\mathcal{N}}$ vérifiant la valeur m_l et appartenant à la classe q_h ;
- La dernière ligne de la table, appelée marge en ligne, contient la somme des effectifs colonne par colonne ; c'est à dire $n_{.l}$ est le nombre d'objets de $O_{\mathcal{N}}$ vérifiant la valeur m_l ;
- La dernière colonne de la table, appelée marge en colonne, contient la somme des effectifs ligne par ligne ; c'est à dire $n_{h.}$ est le nombre d'objets de $O_{\mathcal{N}}$ appartenant à la classe q_h ;

$K_{\mathcal{N}}$	I					$\sum$
	m_1	$\ldots$	m_l	$\ldots$	m_L	
q_1	n_{11}		n_{1l}		n_{1L}	$n_{1.}$
$\vdots$			$\vdots$			
q_h	n_{h1}	$\ldots$	n_{hl}	$\ldots$	n_{hL}	$n_{h.}$
$\vdots$			$\vdots$			
q_H	n_{H1}		n_{Hl}		n_{HL}	$n_{H.}$
$\sum$	$n_{.1}$		$n_{.l}$		$n_{.L}$	N

TABLE 2.3: Table de contingence pour un attribut I qualitatif

A partir de cette table de contingence, les principaux critères de division et d'arrêt peuvent être définis. Afin de simplifier les notations, nous utiliserons les probabilités suivantes :

- $p_{h.} = Pr(K = q_h) = \frac{n_{h.}}{N}$, la fréquence de la classe q_h sur l'échantillon $\Omega_{\mathcal{N}}$;
- $p_{.l} = Pr(I = m_l) = \frac{n_{.l}}{N}$, la fréquence de la modalité m_l sur l'échantillon $\Omega_{\mathcal{N}}$;
- $p_{hl} = Pr(K = q_h / I = m_l) = \frac{n_{hl}}{n_{.l}}$, la probabilité conditionnelle de la classe q_h par rapport à la modalité m_l sur l'échantillon $\Omega_{\mathcal{N}}$;

2.3.1 Critère de division

A chaque nœud $\mathcal{N}$, le but est de construire des sous-groupes d'objets (nœuds successeurs) les plus homogènes possible selon l'attribut à expliquer. Ainsi à chaque étape de construction, **l'attribut le plus discriminant** $I^*_{\mathcal{N}}$ est recherché dans le nœud courant $\mathcal{N}$. L'attribut choisi est appelé variable de segmentation ou **attribut de test.**
Plus précisément, le critère de division, que nous notons *gain* de manière générique, permet de choisir **l'attribut le plus discriminant** $I^*_{\mathcal{N}}$ parmi l'ensemble des attributs

$\mathcal{I}_\mathcal{N}$, selon l'attribut à expliquer $K_\mathcal{N}$, et pour séparer l'ensemble d'objets $O_\mathcal{N}$. **Ce choix correspond à une optimisation locale c'est à dire :**

$$I^*_\mathcal{N} = \underset{I \in \mathcal{I}_\mathcal{N}}{argmax}\ gain(I, \Omega_\mathcal{N}) \tag{2.3.1}$$

Dans la littérature, il existe de nombreux critères de division. L'objectif de cette section n'est pas de présenter tous les critères existants, mais de donner quelques exemples parmi les plus connus. Nous nous intéresserons donc aux critères dits **statistiques** que sont **le test du** χ^2 ou sa normalisation **le t de Tschuprow**. Mais aussi aux critères issus de la théorie de l'information, c'est à dire **les gains informationnels**, tel que **le gain d'entropie** basé sur l'entropie de Shannon, ou encore sa normalisation **le gain ratio**. Et enfin à **un indice de concentration**, **l'indice de Gini** aussi appelé entropie quadratique.

Notons que de nombreux travaux de la littérature ont étudié les différents effets de ces critères sur les méthodes d'analyse de données [Bouroche 80, Saporta 90, Rakotomalala 97, Droesbeke 05, Saporta 06, Tufféry 10] ; nous renvoyons le lecteur à ces références pour plus de précisions.

Le χ^2 et sa normalisation le t de Tschuprow :

Le χ^2 est un critère statistique très connu qui a été développé par Pearson [Pearson 04]. Il permet de mesurer l'écart à l'indépendance *via* les tables de contingence, c'est à dire la différence entre la distribution observée et la distribution théorique qui aurait été obtenue si les deux attributs considérés (la classe $K_\mathcal{N}$ et l'attribut analysé I) étaient indépendants. Dans les tables de contingence, l'indépendance est vérifiée lorsque $\forall h \in \{1, \ldots, H\}\ \forall l \in \{1, \ldots, L\}\ n_{hl} = \frac{n_{h.} \times n_{.l}}{N}$. Si la valeur du χ^2 est significativement grande, alors les deux attributs sont considérés comme dépendants, cela indique que l'attribut permet de bien discriminer les objets de classes différentes.

Sur un nœud, le χ^2 (en fonction de la Table de contingence 2.3) est défini par la formule suivante [Droesbeke 05, Saporta 90, Saporta 06] :

$$gain_{\chi^2}(I, \Omega_\mathcal{N}) = \chi^2_{K_\mathcal{N}, I} = \sum_{h=1}^{H} \sum_{l=1}^{L} \frac{(n_{hl} - \frac{n_{h.} \times n_{.l}}{N})^2}{\frac{n_{h.} \times n_{.l}}{N}} \tag{2.3.2}$$

La **maximisation** du χ^2 permet de déterminer l'attribut le plus lié à la classe. En cas d'indépendance entre les deux attributs, le χ^2 est nul.

Le χ^2 présente deux inconvénients. Prenant des valeurs dans $[0, +\infty[$, il est difficile à analyser. De plus, il avantage les attributs ayant beaucoup de modalités.

Le t de Tschuprow est la normalisation du χ^2 introduite par Tschuprow [Tschuprow a, Tschuprow b]. Il permet de pallier les inconvénients du χ^2 en prenant en compte les degrés de liberté sur le nombre de classes possibles et le nombre de valeurs de l'attribut analysé. Le t de Tschuprow est défini par la formule suivante :

$$gain_{t^2}(I, \Omega_{\mathcal{N}}) = \frac{\chi^2_{K_{\mathcal{N}},I}}{N \times \sqrt{(H-1)(L-1)}} \tag{2.3.3}$$

Le t de Tschuprow est défini sur $[0, 1]$, 0 correspond à l'indépendance parfaite et 1 à la dépendance maximale.

Notons qu'une étude approfondie de ces deux critères est proposée par Saporta dans [Saporta 90].

Le gain informationnel :

Le gain informationnel est issu de la théorie de l'information développée initialement par Hartley en 1928 [Hartley 28] avant de connaître un vrai essor avec Shannon et Weaver en 1949 [Shannon 49]. Ce critère formalise la notion d'information en terme d'entropie. Dans le cadre des arbres de classification, il permet de mesurer le degré de mélange en termes de classes (*i.e.* d'attribut à expliquer) d'un ensemble d'objets. Ce critère repose sur l'entropie de Shannon définie pour l'ensemble $K_{\mathcal{N}}$. L'entropie de Shannon peut être vue comme le calcul de la quantité d'information nécessaire pour connaître $K_{\mathcal{N}}$. Elle est définie par :

$$E(K_{\mathcal{N}}) = -\sum_{h=1}^{H} p_{h.} log_2(p_{h.}) \tag{2.3.4}$$

L'entropie est alors maximale lorsque les objets sont uniformément répartis entre les différentes modalités de l'attribut à expliquer ; elle est minimale dans le cas contraire.

L'entropie conditionnelle est définie à partir de l'entropie de Shannon. Elle peut être vue comme le calcul de la quantité d'information nécessaire pour déterminer les valeurs de $K_{\mathcal{N}}$ sachant celles de I ; *i.e.* le degré de mélange des objets par rapport aux différentes modalités de l'attribut explicatif analysé. L'entropie conditionnelle est définie par :

$$E_c(K_{\mathcal{N}}/I) = -\sum_{l=1}^{L} p_{.l} \sum_{h=1}^{H} p_{hl}\, log_2(p_{hl}) \tag{2.3.5}$$

Le **gain d'entropie**, aussi appelé gain informationnel, est défini à partir de ces deux équations (2.3.4) et (2.3.5). Il correspond à la différence entre le degré de mélange d'un ensemble d'objets en termes de classes, et le degré de mélange de ces objets par rapport aux différentes modalités de l'attribut explicatif I. Il est défini par :

$$gain_{Entropie}(I, \Omega_{\mathcal{N}}) = E(K_{\mathcal{N}}) - E_c(K_{\mathcal{N}}/I) \tag{2.3.6}$$

Ce gain d'entropie augmente lorsque le partitionnement selon l'attribut I permet de mieux séparer les objets de classes différentes. Il vaut 0 lorsque ce partitionnement ne permet pas de séparer les objets de classes différentes. Il est maximal, lorsque les objets sont répartis uniformément sur les classes.

Dans le cadre des arbres de classification, Quinlan [Quinlan 86a] propose **une amélioration du gain d'entropie** qui prend en compte la « quantité d'information » portée par l'attribut explicatif I qui est fonction, entre autres, de son nombre de modalités. Ce gain d'entropie est normalisé pour obtenir le **gain ratio**, permettant une analyse plus aisée avec une valeur dans $[0, 1]$, *i.e.* il vaut 1 lorsque l'attribut I est parfaitement lié à l'attribut $K_{\mathcal{N}}$ (*i.e.* la répartition des objets suivant les modalités de l'attribut I coïncide exactement avec la répartition en classes, partitions pures), et 0 lorsqu'ils sont indépendants :

$$gain_{Ratio}(I, \Omega_{\mathcal{N}}) = \frac{gain_{Entropie}(I, \Omega_{\mathcal{N}})}{E(I)} \tag{2.3.7}$$

La quantité d'information portée par l'attribut explicatif I est alors définie par :

$$E(I) = -\sum_{l=1}^{L} p_{.l} log_2(p_{.l}) \tag{2.3.8}$$

Tout comme le critère du χ^2, le gain informationnel et le gain ratio ont fait l'objet de nombreuses études [Quinlan 86a, Breiman 96b, Rakotomalala 97, Zighed 07, Rokach 08].

L'indice de concentration :

Comme le gain précédent, l'indice de concentration, plus connu sous le nom d'**indice de Gini**, est un cas particulier de mesure d'entropie ; il est d'ailleurs aussi connu sous le nom d'entropie quadratique [Rakotomalala 97]. **L'indice de Gini** correspond à la répartition des classes dans $K_{\mathcal{N}}$, c'est un indice de **pureté** par rapport à $K_{\mathcal{N}}$. Il est défini par :

$$Gini(K_{\mathcal{N}}) = 1 - \sum_{h=1}^{H} p_{h.}^2 \tag{2.3.9}$$

L'indice de Gini conditionnel qui peut être vu comme la concentration/répartition des valeurs de $K_{\mathcal{N}}$ selon I, est défini par :

$$Gini_c(K_{\mathcal{N}}/I) = \sum_{l=1}^{L} p_{.l} \times (1 - \sum_{h=1}^{H} p_{hl}^2) \tag{2.3.10}$$

L'amélioration de la pureté lors d'une division correspond alors à la différence entre la répartition des classes $Gini(K_{\mathcal{N}})$ et la répartition des valeurs de $K_{\mathcal{N}}$ selon I (« diversité »

intra-classe) $Gini_c(K_{\mathcal{N}}/I)$; cette différence peut être vue comme la « diversité » inter-classe. **Le gain** est donc défini par :

$$gain_{Gini}(I, \Omega_{\mathcal{N}}) = Gini(K_{\mathcal{N}}) - Gini_c(K_{\mathcal{N}}/I) \tag{2.3.11}$$

Comme évoqué précédemment, dans un nœud à diviser, **la maximisation du critère** de division permet de définir l'attribut $I^*_{\mathcal{N}}$ permettant de séparer au mieux les objets $O_{\mathcal{N}}$ du nœud selon leur classe.

Un certain nombre de chercheurs s'accordent à dire que **le choix du critère de division n'influence pas les performances en classification des arbres de classification** [Breiman 84, Mingers 89b, Buntine 92, Breiman 96b, Rakotomalala 97] ; seules quelques propriétés de l'arbre divergeraient selon le critère choisi. Par exemple, selon Breiman [Breiman 84, Breiman 96b], l'indice de Gini favoriserait les divisions produisant au moins un nœud successeur pur, même si les effectifs sont déséquilibrés par rapport aux autres nœuds successeurs, tandis que l'entropie choisirait des divisions équilibrant les effectifs dans les nœuds successeurs. Selon Mingers [Mingers 89b], l'indice de Gini produirait les arbres les plus petits et le test du χ^2 produirait les arbres les plus grands.

Dans leur livre [Breiman 84], Breiman *et al.* indiquent clairement que ***le critère utilisé pour élaguer est beaucoup plus important,*** il permet, comparé au critère de division, d'améliorer les performances des arbres.

Avant de préciser ces critères, notons qu'il existe deux types d'élagage, **le pré-élagage** et le **post-élagage**. Le premier consiste à **stopper** la construction de l'arbre **au cours de la phase de division** ; cela coïncide avec la définition d'un critère d'arrêt (*cf.* paragraphe suivant 2.3.2). Le second, le post-élagage, consiste **après la phase de division,** à **réduire** la taille de l'arbre construit en retirant de ce dernier des nœuds ou branches. Le post-élagage est considéré comme **une phase ascendante**. Nous reviendrons sur le post-élagage dans la section 2.4.

2.3.2 Critère d'arrêt (ou de pré-élagage)

A chaque nœud, le critère d'arrêt permet de déterminer si un nœud doit être divisé ou non. Comme pour les critères de division, il existe plusieurs critères d'arrêt dans la littérature. Ces critères peuvent être des **critères empiriques**. Parmi les plus utilisés, se trouvent le critère de confiance, le critère de support, et la taille de l'arbre. Ces critères peuvent aussi être des **critères statistiques** avec en particulier l'absence de division pertinente. De nombreux travaux explicitent ces différents critères [Breiman 84, Rakotomalala 97, Rakotomalala 05a, Rokach 08], le lecteur pourra s'y reporter pour plus de précisions.

Ces critères d'arrêt permettent donc de stopper localement la phase descendante de construction de l'arbre. Ils s'appliquent à un nœud $\mathcal{N}$, et **lorsque ce nœud vérifie**

le critère d'arrêt, il est alors considéré comme une feuille, qui sera étiquetée avec **la classe** k^* **majoritairement représentée**. Cette classe majoritaire **respecte l'équation suivante :**

$$k^* = \underset{q_h \in K_{\mathcal{N}}}{argmax}\ p_{h.} \tag{2.3.12}$$

Le critère de confiance :

Le critère de confiance est plus connu sous le nom de **critère d'homogénéité** ou de pureté **du nœud en termes de classes.** Ce critère dépend d'un **seuil** η qu'il faut définir à l'avance. Un nœud $\mathcal{N}$ vérifiera ce critère si une proportion supérieure à η des objets du nœud appartient à la même classe.

Plus formellement, **soit un nœud** $\mathcal{N}$ **si** $\exists h \in \{1, \ldots, H\}$ *tel que* $p_{h.} \geq \eta$ **alors** $\mathcal{N}$ **est considéré comme une feuille**.

Lorsque l'ensemble des objets présents dans un nœud appartient à une même classe, *i.e.* $\exists h \in \{1, \ldots, H\}\ p_{h.} = 1$ alors le nœud est dit **pur**.

Le critère de support :

Le critère de support consiste à fixer un seuil η portant sur le nombre d'objets minimum qu'un nœud doit contenir. Ainsi, un nœud comptant moins d'objets que ce seuil sera considéré comme une feuille.

Plus formellement, **soit un nœud** $\mathcal{N}$ **si** $N \leq \eta$ **alors** $\mathcal{N}$ **est une feuille**.

Ce critère de support peut aussi être appliqué non pas sur le nœud courant $\mathcal{N}$ mais sur ses successeurs ; c'est à dire si l'un des successeurs du nœud courant ne respecte pas ce critère, alors la division de $\mathcal{N}$ est rejetée, et $\mathcal{N}$ est une feuille.

La profondeur de l'arbre :

Le critère portant sur la profondeur de l'arbre consiste à imposer *a priori* une taille maximale à l'arbre, *i.e.* un nombre maximal de nœuds par branche. Une fois cette taille atteinte, la phase de division est stoppée ; tous les nœuds en bout de branche sont considérés comme des feuilles.

L'absence de division pertinente :

Comme nous l'avons vu précédemment, le critère de division permet de sélectionner l'attribut de test le plus pertinent pour séparer les objets d'un nœud en fonction de leurs classes. Il est courant dans la littérature de s'appuyer sur les critères de division pour stopper la construction de l'arbre ; il s'agit alors de **considérer le critère de division comme critère d'arrêt lorsque la valeur du critère de division est inférieure à un seuil** η. Dans ce cas, si pour toutes les divisions existantes dans un nœud, la

valeur du critère de division est inférieure à ce seuil, alors la division du nœud est non pertinente ; **elle ne permettra pas de séparer les objets de classes différentes.** Le nœud est considéré comme une feuille. Plus formellement, **soit un nœud $\mathcal{N}$ si** $\forall I \in I_{\mathcal{N}}\ gain(I, \Omega_{\mathcal{N}}) \leq \eta$ **alors $\mathcal{N}$ est une feuille**.

Dans la littérature, η est fréquemment fixé selon la formulation d'un test statistique tel le test du χ^2 [Quinlan 86a], ou encore le test de Fisher [Fisher 35, Kent Martin 97]. Ces tests permettent de définir un seuil η de rejet de *l'hypothèse nulle* H_0 correspondant à l'indépendance de l'attribut de test et de l'attribut à expliquer. Valider H_0 revient à valider le critère d'arrêt d'absence de division ou de division pertinente. Ces tests portant sur une information locale, *i.e.* information contenue dans un nœud, ils ont souvent été critiqués car considérés comme ne reflétant pas réellement ce qui peut se passer dans les niveaux inférieurs (*i.e.* pas de prise en compte des informations qui pourraient être apportées par des successeurs) [Breiman 84, Kent Martin 97, Rakotomalala 97, Dong 01, Agresti 02].

De manière générale, **ces critères sont difficiles à utiliser** car ils impliquent tous la définition d'un seuil η [Breiman 84, Rakotomalala 97, Nakache 03, Rokach 08]. Si le seuil fixé est trop permissif alors l'arbre sera trop complexe (sur-dimensionné) avec le risque du sur-apprentissage (*cf.* section 1.3.1). Inversement, si le seuil fixé est trop restrictif alors l'arbre sera trop petit (sous-dimensionné) et ne représentera pas les informations contenues dans la base d'apprentissage de manière suffisante, *i.e.* sous-apprentissage (*cf.* section 1.3.1).

Les processus de post-élagage leur ont donc souvent été préférés.

2.4 Le post-élagage

Les arbres construits lors de la phase de division sont généralement *trop complexes* ; ils *collent* trop aux données d'apprentissage. La phase d'élagage sert alors à **optimiser le modèle** en termes de parcimonie et de capacité de généralisation. L'élagage a deux objectifs, le premier est de réduire la complexité structurelle des arbres et le second est de définir le sous-arbre de bonne taille pour éviter **le phénomène du sur-apprentissage** (*cf.* section 1.3.1). Ces deux objectifs sont bien entendu fortement liés.

Les premiers travaux sur les méthodes de post-élagage sont attribués à Breiman *et al.* [Breiman 84]. Dans leur livre, les auteurs motivent leur démarche par une étude empirique. Cette étude, illustrée par la Figure 2.4.1, compare l'évolution du taux d'erreur en resubstitution (*i.e.* calculé sur la base d'apprentissage) et celle du taux d'erreur en généralisation (*i.e.* calculé sur une base indépendante de la base d'apprentissage - base de test dans le livre de Breiman *et al.*) avec le nombre de feuilles de l'arbre.

Dans la Figure 2.4.1, la courbe bleue correspond à l'erreur en resubstitution, et la courbe rouge correspond à l'erreur en généralisation.

Sur cette Figure 2.4.1, nous voyons qu'à mesure que le nombre de feuilles (la taille de l'arbre) augmente, le taux d'erreur en resubstitution diminue constamment. En revanche, le taux d'erreur en généralisation montre d'abord une décroissance rapide, jusqu'à un arbre avec une dizaine de feuilles, puis nous observons que le taux d'erreur reste sur un plateau avant de se dégrader (*i.e.* croître) lorsque l'arbre est manifestement sur-dimensionné, *i.e.* **le modèle est en sur-apprentissage**. **La taille idéale de l'arbre** correspond au plateau dessiné par le taux d'erreur en généralisation. Un arbre de cette taille est obtenu par application d'un post-élagage à un arbre maximal (*i.e.* arbre construit en appliquant des critères d'arrêt très permissifs au cours de la phase de division). Cet arbre aura de bonnes performances en généralisation, tout en étant plus simple que l'arbre maximal et donc plus lisible.

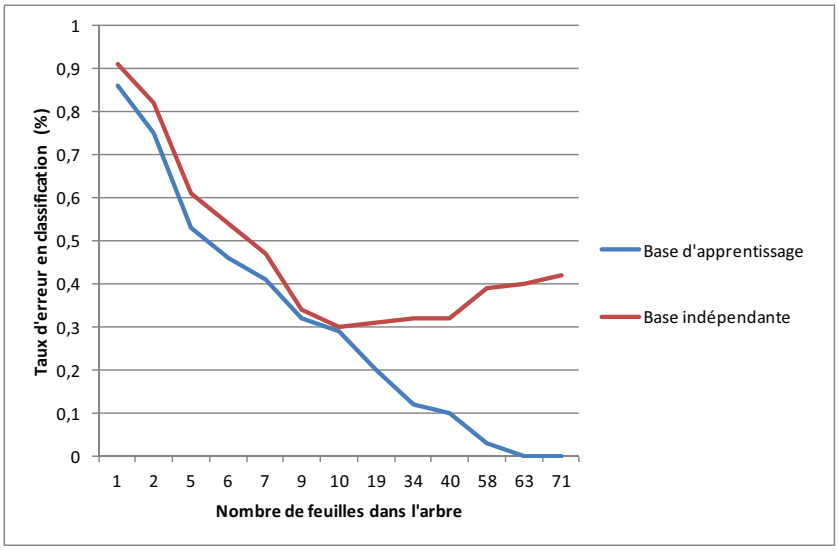


FIGURE 2.4.1: Évolution des taux d'erreur en fonction du nombre de feuilles d'un arbre

Il est difficile de prétendre à l'exhaustivité dans la présentation des méthodes de post-élagage, tant elles sont nombreuses dans la littérature. De plus, de nombreuses études ont été menées, celles-ci sont souvent contradictoires quant à l'efficacité de certaines méthodes. Les auteurs de ces études justifient généralement ces contradictions en mettant en avant des circonstances d'études différentes ou des domaines d'études différents [Quinlan 86b, Quinlan 87b, Esposito 97, Breslow 97, Rakotomalala 97]. Aussi **nous nous concentrons sur les méthodes les plus célèbres** que sont l'élagage par minimisation de la mesure de coût-complexité, l'élagage par minimisation de l'erreur et l'élagage basée sur l'erreur pessimiste.

La définition de ces différents post-élagages nécessite l'introduction de quelques notations :

L'arbre maximal, construit à partir d'une base d'apprentissage lors de la phase de division, sera noté DT_0. Les méthodes de post-élagage présentées par la suite reposent généralement sur la construction itérative d'une séquence d'arbres élagués ; cette **séquence de sous-arbres** sera notée $\{DT_0, DT_1, \ldots, DT_t\}$. L'arbre optimal DT^* est ensuite choisi dans cette séquence, généralement grâce à une base de validation. Nous verrons dans la suite que selon la méthode d'élagage choisie, la séquence de sous-arbres est parfois construite entièrement, *i.e.* $DT_t = t_1$ où t_1 représente le sous-arbre contenant un unique nœud, *i.e.* **la racine de** DT_0.

Le **nombre de nœuds d'un arbre** DT_i sera noté $|DT_i|$ tandis que son **nombre de feuilles** sera noté $L(DT_i)$. L'élagage d'un arbre DT_i consiste généralement à transformer un ou plusieurs de ses nœuds internes en feuilles. Nous verrons aussi, qu'en plus de cette transformation élémentaire, certaines méthodes proposent de remplacer un nœud interne par une branche ST_i de l'arbre (*i.e.* une branche regroupe l'ensemble des nœuds successeurs du nœud courant).

DT_{i+1} est le résultat de l'élagage de DT_i. Par construction, on a donc, $|DT_0| > \ldots > |DT_i| > |DT_{i+1}| > \ldots > |DT_t|$. La transformation d'un nœud interne $\mathcal{N}$ de DT_i se fait par évaluation de l'apport (en termes de taux d'erreur) de la branche ST_i ayant pour racine ce nœud interne. L'évaluation de l'apport d'une branche ST_i est propre à chaque méthode. Pour ces évaluations, **l'arbre** DT_i **élagué temporairement en** $\mathcal{N}$ **sera noté** $\tilde{DT}_i$**.**

Comme précisé précédemment, l'élagage a parmi ses objectifs, l'amélioration des performances des arbres en généralisation. A ces fins, les évaluations présentées dans les paragraphes suivants utilisent les **taux d'erreur** tels que définis dans la section 1.3.1. Rappelons que le taux d'erreur d'un arbre DT_i sur une base d'apprentissage BA contenant N objets est $\varepsilon(DT_i, BA) = \frac{MC(DT_i, BA)}{N}$ (*cf.* section 1.3.1).

Avant de présenter plus spécifiquement les méthodes d'élagage et les mesures associées, **nous présentons dans l'Algorithme 2.3, l'élagage d'un arbre** DT_0 par génération complète de la séquence de sous-arbres et **par choix du meilleur arbre dans la séquence selon son taux d'erreur sur une base de validation**.

Notons que le choix des branches à élaguer à chaque étape est propre à chaque modèle, nous présentons les algorithmes correspondant (Élague_Arbre(DT_i, BA XOR BV)) dans les Algorithmes 2.4, 2.5 et 2.6.

Exemple. La Figure 2.4.2 illustre un arbre DT_i, la Figure 2.4.3 représente une de ses branches ST_i ayant pour racine $\mathcal{N}$ et l'arbre élagué correspondant est présenté par la Figure 2.4.4.

Algorithme 2.3 Meilleur_Arbre_Élagué(DT_0,BA,BV) (avec génération complète de la séquence)

Entrées :
- DT_0 l'abre maximal construit lors de la phase de division à partir de la base d'apprentissage ;
- BA la base d'apprentissage ;
- BV la base de validation ;

Sorties :
- DT^* le meilleur arbre élagué ;

DÉBUT

Construire une liste vide Seq ;
Ajouter DT_0 à Seq ;
$DT_i \leftarrow DT_0$;
tant que $DT_i \neq t_1$ **faire**
  $DT_{i+1} \leftarrow$ *Élague_Arbre*$(DT_i, BA\ XOR\ BV)$;
  Ajouter DT_{i+1} à Seq ;
  $DT_i \leftarrow DT_{i+1}$;
fin tant que
$DT^* \leftarrow$ Choisir dans Seq le meilleur arbre sur BV ;
renvoyer DT^* ;

FIN

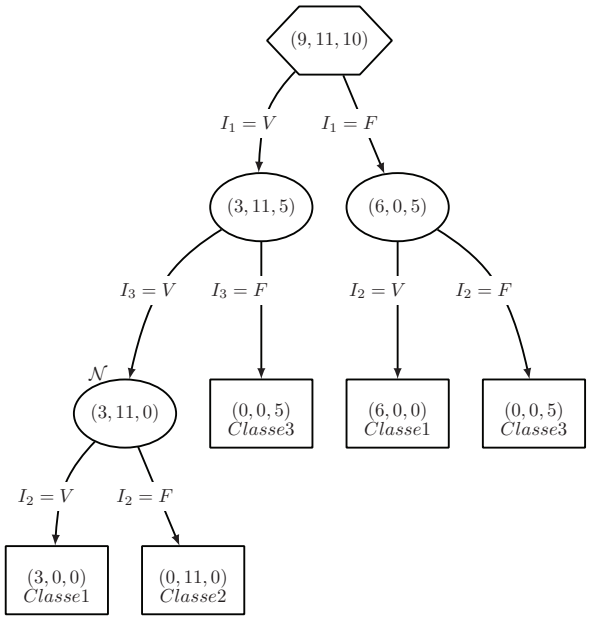


FIGURE 2.4.2: Un arbre DT_i

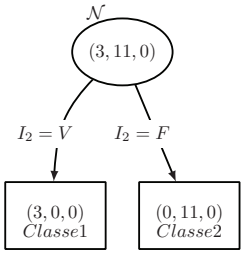


FIGURE 2.4.3: Une branche ST_i de DT_i (ayant pour racine $\mathcal{N}$)

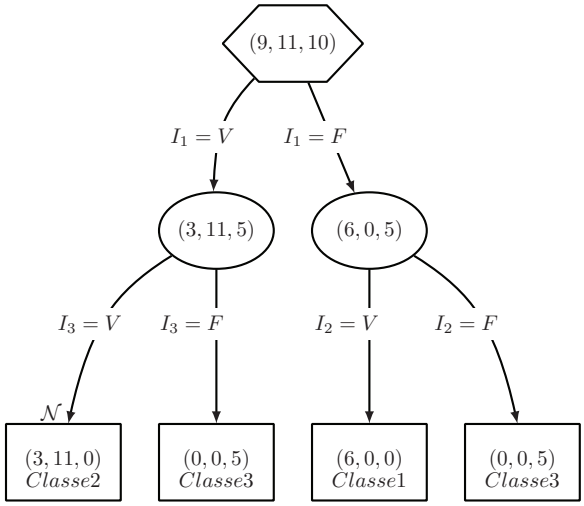


FIGURE 2.4.4: Sous-arbre DT_i élagué en $\mathcal{N}$

Élagage par minimisation de la mesure de coût-complexité (*CCP*)

L'élagage par minimisation de la mesure de coût-complexité a été introduit par Breiman *et al.* [Breiman 84] sous le nom *Cost-Complexity Pruning (**CCP**)*. Ce processus se déroule en **deux étapes** ; la première correspond à la construction d'une séquence complète d'arbres élagués $\{DT_0, DT_1, \ldots, t_1\}$, en **minimisant la mesure de coût-complexité** sur la **base d'apprentissage** (*cf.* paragraphe suivant). La seconde étape correspond au **choix du meilleur arbre** parmi ceux de la séquence construite ; ce choix dépend d'une **base de validation**.

Ainsi la séquence d'arbres contient tout d'abord l'arbre construit à partir de la base d'apprentissage DT_0, puis ce dernier est élagué selon le processus expliqué ci-dessous pour former l'arbre DT_1, et le processus est réitéré sur chaque arbre élagué jusqu'à obtenir l'arbre ne contenant que la racine t_1.

Minimisation de la mesure de coût-complexité

La mesure de coût-complexité définie par Breiman *et al.,* prend en compte d'un côté le taux d'erreur en resubstitution d'un arbre DT_i, *i.e.* **le coût**. De l'autre le nombre de feuilles de l'arbre $L(DT_i)$, *i.e.* **la complexité**.

Définition 5. La mesure de coût-complexité associée à un arbre DT_i est définie par la somme suivante :

$$\Theta_\sigma(DT_i) = \varepsilon(DT_i, BA) + \sigma \times L(DT_i) \tag{2.4.1}$$

Avec σ un coefficient de complexité utilisé pour la construction des arbres élagués, il permet de pénaliser les arbres contenant beaucoup de feuilles.

Pour élaguer DT_i, tous ses nœuds internes sont considérés un à un. A chaque nœud interne $\mathcal{N}$, correspond une branche ST_i, pour chacune d'elles, la mesure de coût-complexité de l'arbre temporairement élagué en $\mathcal{N}$ est évaluée, *i.e.* $\Theta_\sigma(\tilde{DT}_i)$. Puis la branche permettant de minimiser la différence entre $\Theta_\sigma(\tilde{DT}_i)$ et $\Theta_\sigma(DT_i)$ est définitivement élaguée pour former DT_{i+1}.

Comme présenté par Esposito *et al.* [Esposito 97], cela revient finalement à élaguer la ou les branches ST_i qui permettent de minimiser la valeur σ définie par :

$$\sigma = \frac{\varepsilon(\tilde{DT}_i, BA) - \varepsilon(DT_i, BA)}{(L(ST_i) - 1)} \tag{2.4.2}$$

Si plusieurs branches minimisent cette valeur, alors elles sont toutes supprimées de DT_i pour former DT_{i+1}.

Ainsi, la séquence complète d'arbres élagués est générée en suivant l'Algorithme 2.3 et l'Algorithme 2.4 pour le choix des branches à élaguer à chaque étape.

Algorithme 2.4 Élague_Arbre(DT_i,BA)~CCP

Entrées :
- DT_i un arbre à élaguer ;
- BA la base d'apprentissage ;

Sorties :
- DT_{i+1} l'arbre élagué ;

DÉBUT

$\sigma_{min} = +\infty$;
Liste_branches $\leftarrow$ liste vide de branches ;
pour chaque noeud interne $\mathcal{N}$ de DT_i **faire**
  $ST_i \leftarrow$ branche de DT_i de racine $\mathcal{N}$;
  $\tilde{DT}_i \leftarrow DT_i$ élagué en $\mathcal{N}$;
  $\sigma = \frac{\varepsilon(\tilde{DT}_i,BA)-\varepsilon(DT_i,BA)}{L(ST_i)-1}$;
  si $\sigma = \sigma_{min}$ **alors**
    Ajouter ST_i à *Liste_branches* ;
  sinon
    si $\sigma < \sigma_{min}$ **alors**
      Effacer *Liste_branches* ;
      Ajouter ST_i à *Liste_branches* ;
      $\sigma_{min} = \sigma$;
    fin si
  fin si
fin pour
$DT_{i+1} \leftarrow DT_i$;
pour chaque $ST_i \in$ *Liste_branches* **faire**
  Etiqueter le noeud racine de ST_i avec sa classe majoritaire ;
  $DT_{i+1} \leftarrow DT_{i+1}$ élagué au noeud racine de ST_i ;
fin pour

renvoyer DT_{i+1} ;

FIN

Choix du meilleur arbre

Une fois la séquence d'arbres construite, il faut choisir le meilleur en généralisation. Pour cela, Breiman *et al.* utilisent les taux d'erreur des arbres de la séquence commis sur une base de validation. **Le meilleur sous-arbre** DT^* de la séquence est celui possédant le moins de feuilles possibles et vérifiant la condition suivante :

$$\varepsilon(DT^*, BV) < \varepsilon^* + se(\varepsilon^*) \tag{2.4.3}$$

Avec
- $\varepsilon^* = \min_{DT_i \in \{DT_0,\ldots,t_1\}} \varepsilon(DT_i, BV)$: l'erreur minimale observée pour les arbres de la séquence
- $se(\varepsilon^*)$: l'erreur standard associée à ε^*, $se(\varepsilon^*) = \sqrt{\frac{\varepsilon^* \times (1-\varepsilon^*)}{N}}$

Ainsi dans l'Algorithme 2.3, la ligne *Choisir dans Seq le meilleur arbre* correspond à choisir DT^* possédant le moins de feuilles possibles et vérifiant l'équation 2.4.3.

Élagage par minimisation de l'erreur (*REP*)

L'élagage par minimisation de l'erreur a été introduit par Quinlan [Quinlan 86b, Quinlan 87b, Quinlan 93] sous le nom de *Reduced Error Pruning (**REP**)*. A la différence de l'élagage précédent, qui utilise la base d'apprentissage pour sa première étape, l'élagage par minimisation de l'erreur utilise **uniquement la base de validation**. Il coupe récursivement les branches en partant du bas, jusqu'à ce que le taux d'erreur constaté pour l'arbre courant DT_i sur la base de validation ne soit pas réduit, par au moins un des élagages possible de DT_i. La séquence d'arbres n'est donc pas construite entièrement et le meilleur arbre est alors le dernier arbre construit.

Plus précisément, pour toutes les branches ST_i d'un arbre DT_i, l'évolution du taux d'erreur sur la base de validation est analysée. Si le taux d'erreur pour l'arbre DT_i ($\varepsilon(DT_i, BV)$) est supérieur ou égal au taux d'erreur de l'arbre élagué temporairement $\tilde{DT}_i$ ($\varepsilon(\tilde{DT}_i, BV)$) et si la branche ST_i ne contient pas de sous-branche vérifiant cette même différence, alors ST_i doit être élaguée pour former DT_{i+1}. Si plusieurs branches vérifient la même différence, elles sont toutes élaguées. Ainsi, DT_{i+1} a un taux d'erreur équivalent ou plus faible que celui de DT_i, sur la base de validation. Le processus est réitéré à partir de DT_{i+1} et tant que le taux d'erreur sur la base de validation diminue.

L'algorithme de choix des branches à élaguer à chaque étape est présenté dans l'Algorithme 2.5.

Élagage basé sur l'erreur (*EBP*)

L'élagage basé sur l'erreur a été défini par Quinlan [Quinlan 93] sous le nom de *Error-Based Pruning (**EBP**)*. Cette méthode est une amélioration/reformulation de la mé-

Algorithme 2.5 Élague_Arbre(DT_i,BV)~REP

Entrées :
- DT_i un arbre à élaguer ;
- BV la base de validation ;

Sorties :
- DT_{i+1} l'arbre élagué ;

DÉBUT

$Liste\_branches \leftarrow$ liste vide de branches ;
$diff = 0$;
pour chaque noeud interne $\mathcal{N}$ de DT_i **faire**
  $ST_i \leftarrow$ branche de DT_i de racine $\mathcal{N}$;
  $\tilde{DT}_i \leftarrow DT_i$ élagué en $\mathcal{N}$;
  si $\varepsilon(\tilde{DT}_i, BV) - \varepsilon(DT_i, BV) = diff$ et $\nexists b \in Liste\_branches \mid b \subset ST_i$ **alors**
    Ajouter ST_i à $Liste\_branches$;
  sinon
    si $\varepsilon(\tilde{DT}_i, BV) - \varepsilon(DT_i, BV) < diff$ **alors**
      Effacer $Liste\_branches$;
      Ajouter ST_i à $Liste\_branches$;
      $diff = \varepsilon(\tilde{DT}_i, BV) - \varepsilon(DT_i, BV)$;
    fin si
  fin si
fin pour
si $Liste\_branches = \emptyset$ **alors**
  renvoyer DT_i ;
sinon
  $DT_{i+1} \leftarrow DT_i$;
  pour chaque $ST_i \in Liste\_branches$ **faire**
    $DT_{i+1} \leftarrow DT_{i+1}$ élagué au noeud racine de ST_i ;
  fin pour
fin si

renvoyer DT_{i+1} ;

FIN

thode de minimisation de l'erreur pessimiste (*Pessimistic Error Pruning ~ PEP*) définie elle aussi par Quinlan [Quinlan 86b, Quinlan 87b]. Quinlan a développé cet élagage en observant que l'utilisation d'une base de validation pouvait être contraignante, en particulier lorsque l'ensemble de données à disposition compte peu d'objets (*cf.* section 1.3.1). **Cet élagage ne nécessite donc pas de base de validation ; seule la base d'apprentissage est utilisée**.

Comme pour l'élagage précédent, l'arbre élagué DT_{i+1} est construit à partir de DT_i en analysant l'évolution du taux d'erreur en resubstitution. Pour éviter une estimation trop optimiste du taux d'erreur, Quinlan utilise un taux d'erreur corrigé $\varepsilon_{UB}(.)$ qui correspond à la borne supérieure de l'intervalle de confiance associé au taux d'erreur en resubstitution. $\varepsilon_{UB}(DT_i, BA)$ est alors défini selon la formule suivante :

$$\varepsilon_{UB}(DT_i, BA) = \varepsilon(DT_i, BA) + u_{cf} \times \sqrt{\frac{\varepsilon(DT_i, BA) \times (1 - \varepsilon(DT_i, BA))}{N}} \qquad (2.4.4)$$

Où
- $\varepsilon(DT_i, BA)$ correspond au taux d'erreur en resubstitution de l'arbre DT_i ;
- cf est un niveau de confiance ;
- u_{cf} est la limite/valeur critique associée à cf pour la distribution binomiale ;

A partir de cette erreur corrigée, Quinlan définit trois erreurs corrigées et les impute à chaque nœud interne $\mathcal{N}$ et à chaque branche associée ST_i :

1. $\varepsilon_{UB}(ST_i, \Omega_{\mathcal{N}})$: l'erreur commise par la branche ST_i sur l'ensemble des objets $\Omega_{\mathcal{N}}$ issus de la base d'apprentissage et contenus dans sa racine $\mathcal{N}$;
2. $\varepsilon_{UB}(\mathcal{N}, \Omega_{\mathcal{N}})$: l'erreur commise par la racine $\mathcal{N}$ de ST_i sur l'ensemble des objets $\Omega_{\mathcal{N}}$ qu'elle contient ;
3. $\varepsilon_{UB}(S^*, \Omega_{\mathcal{N}^*})$: l'erreur commise par **la branche spécifique** S^* de ST_i sur l'ensemble des objets $\Omega_{\mathcal{N}^*}$ contenus dans sa racine $\mathcal{N}^*$; Avec S^* la branche de ST_i dont la racine $\mathcal{N}^*$ est le successeur immédiat de $\mathcal{N}$ contenant le plus d'objets issus de $\Omega_{\mathcal{N}}$.

Si la valeur minimale entre ces trois erreurs corrigées est la première, alors la branche ST_i n'est pas élaguée, si c'est la seconde alors DT_i est élagué en $\mathcal{N}$ selon la méthode vue précédemment ; si c'est la troisième alors la branche S^* est greffée en lieu et place de ST_i. Si aucune branche n'est élaguée, alors le processus s'arrête. Notons que le parcours des noeuds interne pour l'évaluation se fait du bas vers le haut, *i.e.* en commençant par les prédécesseurs des feuilles.

L'algorithme de choix des branches à élaguer à chaque étape est présenté par l'Algorithme 2.6.

Le principe de la greffe est illustré dans la Figure 2.4.6, où la branche S^* en vert a été greffée en remplacement de la branche ST_i en rouge dans la Figure 2.4.5 (représentant

Algorithme 2.6 Élague_Arbre(DT_i,BA)~EBP

Entrées :
- DT_i un arbre à élaguer ;
- BA la base d'apprentissage ;

Sorties :
- DT_{i+1} l'arbre élagué ;

DÉBUT

$DT_{i+1} \leftarrow DT_i$;
pour chaque noeud interne $\mathcal{N}$ de DT_i **faire**
  $ST_i \leftarrow$ branche de DT_i de racine $\mathcal{N}$;
  $error_{ST_i} = \varepsilon_{UB}(ST_i, \Omega_{\mathcal{N}})$;
  $error_{\mathcal{N}}$=$\varepsilon_{UB}(\mathcal{N},\Omega_{\mathcal{N}})$;
  $error_{S^*} = \varepsilon_{UB}(S^*, \Omega_{\mathcal{N}^*})$;
  $error_{min} = min(error_{ST_i}, error_{\mathcal{N}}, error_{S^*})$;
  si $error_{min} = error_{\mathcal{N}}$ **alors**
    $DT_{i+1} \leftarrow DT_i$ élagué en $\mathcal{N}$;
  fin si
  si $error_{min} = error_{S^*}$ **alors**
    $DT_{i+1} \leftarrow DT_i$ greffé en $\mathcal{N}$ avec S^* ;
  fin si
fin pour

renvoyer DT_{i+1} ;

FIN

DT_i).

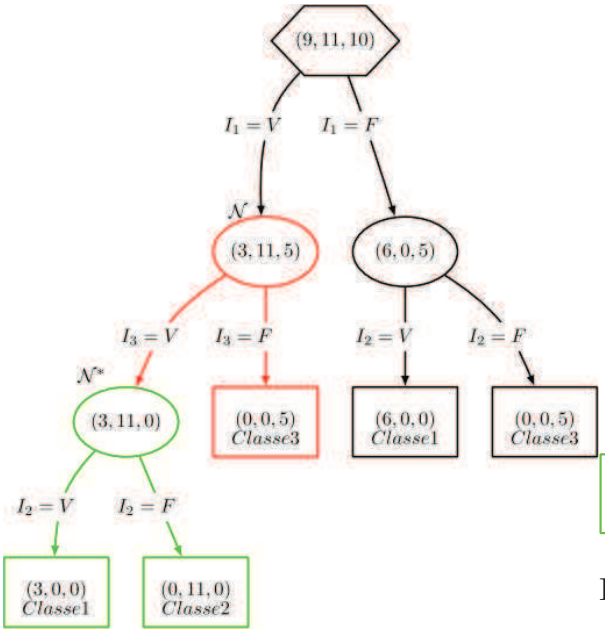


FIGURE 2.4.5: Arbre de référence DT_i

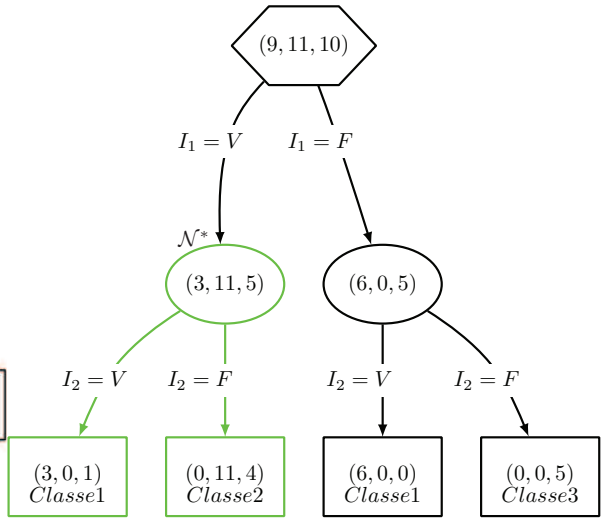


FIGURE 2.4.6: Arbre DT_{i+1} après la greffe au nœud $\mathcal{N}$ de la branche issue du nœud $\mathcal{N}^*$

Il existe dans la littérature de nombreuses méthodes d'élagage, il est difficile d'être exhaustif, nous avons présenté une fraction d'entre elles ci-dessus. Notons aussi parmi les plus citées, la méthode de la valeur critique (CVP) proposée par Mingers [Mingers 87], et celle de Niblett et Bratko basée sur l'erreur minimum ou erreur espérée (MEP) [Niblett 87]. Nous renvoyons le lecteur aux références associées pour plus de détails.

Les méthodes d'élagage **les plus réputées ont souvent été comparées.** Notons parmi les articles comparatifs réputés, l'article de Quinlan [Quinlan 86b, Quinlan 87b] qui rappelle le modèle CCP présenté ci-dessus avant de proposer les modèles REP et PEP puis de les comparer. Mais aussi celui de Mingers [Mingers 89a] qui pour sa part compare CCP, CVP, MEP, REP et PEP. Ou encore celui de Esposito *et al.* [Esposito 97] où les auteurs étendent leur article précédent [Malerba 96] ; ils rappellent et analysent les modèles REP, PEP, MEP, CVP, CCP et EBP, avant des les comparer *via* quelques 3375 expérimentations. Moins réputé mais plus récent, notons l'article de Rokach et Maimon [Rokach 05] qui propose entre autre une comparaison des trois méthodes présentées ci-dessus. La Table 2.4 présente quelques caractéristiques extraites de ces articles concernant les méthodes CCP, REP et EBP.

Notons enfin que selon Rakotomalala [Rakotomalala 05a], le post-élagage utilisé par CART est celui qui *intègre tous les "bons" ingrédients d'un apprentissage efficace* :

	#étapes	BV	Description	Analyse
CCP	2	Oui	- Tendance au sur-élagage - Compléxité quadratique selon le nombre de nœuds internes	- Sélectionne le meilleur arbre selon BV dans un ensemble d'arbres construits selon BA
REP	1	Oui	- Tendance au sur-élagage - Complexité linéaire	- Trouve le plus petit arbre ayant le plus petit taux d'erreur sur BV
EBP	1	Non	- Tendance au sous-élagage	- Améliore la méthode PEP - Processus de greffe en plus de celui de l'élagage - Selon [Esposito 97] c'est la méthode la plus stable/performante

TABLE 2.4: Comparaison des méthodes d'élagage

1. *une évaluation non biaisée de l'erreur pour déterminer le bon arbre ;*
2. *une réduction de l'espace des hypothèses avec le principe des séquences d'arbres rangés à coût-complexité décroissant, limitant ainsi le risque de sur-apprentissage sur l'échantillon de validation ;*
3. *une préférence donnée à la simplicité avec la règle de "l'écart-type" avant l'erreur minimale : l'idée est de se rapprocher du point d'inflexion dans l'évolution de l'erreur en fonction du nombre de feuilles de l'arbre (cf. Figure 2.4.1) ;*
4. *et lorsque la taille du fichier d'apprentissage est réduite, un système de validation croisée est proposé pour réaliser le post-élagage.*

2.5 Algorithmes de construction

Comme nous l'avons précisé en introduction de ce chapitre, il existe différents algorithmes de construction d'arbres de classification, ces algorithmes diffèrent les uns des autres principalement par les critères de division et d'arrêt qu'ils utilisent, mais aussi par leur mise en œuvre d'une méthode d'élagage. Nous nous intéressons principalement à des algorithmes connus qui utilisent les critères et les méthodes d'élagage présentées ci-dessus. Plus précisément, **nous nous intéressons au modèle ChAID** développé par Kass [Kass 80], à **CART** développé par Breiman *et al.* [Breiman 84], et à **C4.5** développé par Quinlan [Quinlan 93].

2.5.1 ChAID

ChAID *(Chi-squared Automatic Interaction Detection)* a été défini par Kass en 1980 [Kass 80]. C'est un successeur de l'algorithme AID défini par Morgan et Sonquist [Morgan 63, Sonquist 71].

ChAID utilise **le test du χ^2** comme critère de division et d'arrêt. Il ne propose **pas de post-élagage**. Lors de la division, ChAID construit autant de successeurs que de valeurs possibles pour l'attribut test, ainsi les arbres issus de cet algorithme sont des **arbres M-aires**.

L'une des particularités de ChAID est la définition d'une **option de fusion des nœuds successeurs**. Cette fusion est mise en place après la division d'un nœud et une fois les nœuds successeurs construits. Les profils de ces nœuds successeurs sont analysés deux à deux. Si, pour deux nœuds donnés, les profils sont proches au sens du test d'équivalence distributionnelle du χ^2 à $(K-1)$ degrés de liberté, alors **ces nœuds sont fusionnés** (ce test identifie les nœuds qui donneront des feuilles ayant la même distribution). Le processus est réitéré tant qu'une fusion est possible.

Pour illustrer ce processus de fusion, reprenons l'exemple de la Figure 2.2.1. Considérons la division de la racine ; trois nœuds successeurs ont été construits $\mathcal{N}_1 = \{o_1, o_2, o_8, o_9, o_{11}\}$, $\mathcal{N}_2 = \{o_3, o_7, o_{12}, o_{13}\}$, $\mathcal{N}_3 = \{o_4, o_5, o_6, o_{10}, o_{14}\}$. Le test d'équivalence distributionnelle du χ^2 à $(K-1)$ degrés de liberté consiste à comparer les distributions des nœuds deux à deux. C'est à dire, calculer à partir de la table de contingence associée aux deux nœuds, la valeur du χ^2. La table de contingence associée aux nœuds $\mathcal{N}_1$ et $\mathcal{N}_2$ est la Table 2.5. Ici, la valeur du χ^2 est 3.60.

En fixant le risque de première espèce à 10% par exemple, le seuil à ne pas dépasser pour valider la fusion est de 2.706 (*cf.* Annexe A table du χ^2 colonne $p = 0.10$ et ligne $K = 1$). Ainsi, $\mathcal{N}_1$ et $\mathcal{N}_2$ ne sont pas fusionnés. La Table 2.6 récapitule les différentes comparaisons entre $\mathcal{N}_1$, $\mathcal{N}_2$ et $\mathcal{N}_3$, les valeurs du χ^2 associées ainsi que la validité d'une fusion entre ces nœuds. Selon cette table, les fusions valides sont la fusion de $\mathcal{N}_1$ et $\mathcal{N}_3$ et la fusion de $\mathcal{N}_2$ et $\mathcal{N}_3$. Les profils les plus proches sont ceux de $\mathcal{N}_1$ et $\mathcal{N}_3$, ainsi ChAID fusionnerait ces deux nœuds pour obtenir, après la première division et la fusion, l'arbre est présenté dans la Figure 2.5.1. Après cette première fusion, la recherche d'une fusion valide serait réitérée pour $\mathcal{N}_2$ et le nœud issu de la fusion de $\mathcal{N}_1$ et $\mathcal{N}_3$. Le χ^2 associé étant supérieur au seuil attendu (*i.e.* synonyme de distributions différentes), ces nœuds ne seraient pas fusionnés et le processus de division reprendrait son cours.

2.5.2 CART

CART *(Classification And Regression Trees)* correspond à l'algorithme défini par Breiman *et al.* dans leur livre très connu [Breiman 84].

CART utilise **l'indice de Gini** comme critère de division et **le critère de support**, avec $\eta = 5$, comme critère d'arrêt. Il met en œuvre la **méthode d'élagage CCP**.

	$\mathcal{N}_1$	$\mathcal{N}_2$	Σ
Jouer = Oui	2	4	6
Jouer = Non	3	0	3
Σ	5	4	9

TABLE 2.5: Table de contingence des nœuds $\mathcal{N}_1$ et $\mathcal{N}_2$

	Distribution	χ^2	candidats à la fusion ?	Choix
$\mathcal{N}_1$,$\mathcal{N}_2$	$(2,3)$ et $(4,0)$	3.60	non	
$\mathcal{N}_1$,$\mathcal{N}_3$	$(2,3)$ et $(3,2)$	0.40	oui	sélectionnés pour la fusion
$\mathcal{N}_2$,$\mathcal{N}_3$	$(4,0)$ et $(3,2)$	2.06	oui	

TABLE 2.6: Calcul pour la fusion deux à deux des nœuds $\mathcal{N}_1$, $\mathcal{N}_2$ et $\mathcal{N}_3$

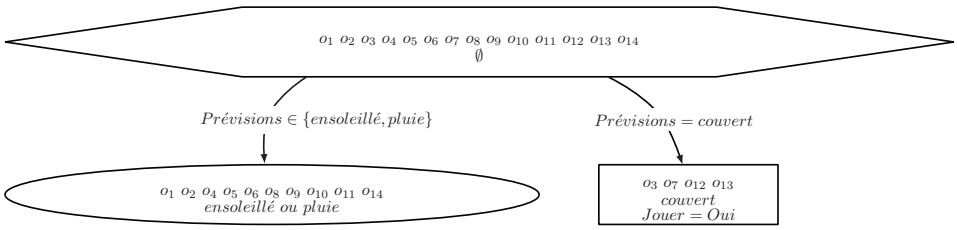


FIGURE 2.5.1: Arbre après fusion

Lors de la division d'un nœud, CART ne construit que deux nœuds successeurs ; ainsi les arbres issus de cet algorithme sont des **arbres binaires**. Pour effectuer le choix de l'attribut de test, CART regroupe en deux sous-ensembles les différentes valeurs possibles pour un attribut analysé ($2^{L-1} - 1$ combinaisons possibles).

CART est un algorithme souvent cité pour ses bonnes performances [Rakotomalala 97, Wu 07, Tufféry 10].

2.5.3 C4.5

C4.5 [Quinlan 93] est considéré comme une amélioration de l'algorithme ID3 [Quinlan 86a] *(Induction of Decision Tree)*.

Alors que ID3 utilisait l'entropie de Shannon comme critère de division, C4.5 utilise sa version normalisée, **le gain ratio**. De plus, ID3 ne traitait pas les attributs quantitatifs ; ces derniers devaient être transformés en attributs qualitatifs avant la construction de l'arbre. Ce n'est pas le cas pour C4.5 (*cf.* chapitre 5). Pour le critère d'arrêt, C4.5 utilise **le critère du support** avec un seuil $\eta = 2$. Puis il met en œuvre la méthode de **post-élagage EBP**. Comme ChAID, lors de la division, C4.5 construit autant de successeurs que de valeurs possibles pour l'attribut test, ainsi les arbres issus de cet algorithme sont des **arbres M-aires**.

C4.5 est un algorithme qui est aussi souvent cité pour ses bonnes performances

[Lim 00, Kotsiantis 07, Wu 07]. Mais, il produit de très grands arbres avec un risque de sur-apprentissage, il est conseillé de l'utiliser lorsque les données à traiter sont nombreuses.

L'une des particularités de C4.5 est sa mise en œuvre du **processus de greffe** *via* le post-élagage EBP.

Notons que le code de C4.5 est accessible sur internet [1], mais il n'est plus mis à jour et est remplacé par son successeur C5.0. Ce dernier a d'abord été proposé dans un logiciel commercial, avant que son code ne soit accessible lui aussi sur internet [2] depuis peu de temps (~début 2012). Les principales améliorations apportées entre ces deux versions sont la prise en compte de coûts de mauvaise classification associés à certaines classes, mais aussi la prise en compte de l'importance de certains objets par rapport à d'autres *via* un poids associé aux objets. Selon son auteur, C5.0 est plus facile à utiliser que la version C4.5.

La Table 2.7 présente un récapitulatif des algorithmes de construction d'arbre de classification présentés ci-dessus, avec leurs paramètres, leurs avantages et leurs inconvénients.

2.6 Méthode de classification

Une fois l'arbre de classification construit, il existe différentes manières de l'utiliser pour classer de nouveaux exemples. La plus connue correspond à la navigation dans la structure arborescente par validation successive des attributs de test. La seconde correspond à l'extraction de règles de classification.

2.6.1 Navigation

La navigation dans un arbre de classification pour classer un nouvel objet est une méthode intuitive. Connaissant les attributs explicatifs du nouvel objet, il s'agit de descendre de nœud en nœud dans l'arbre. A chaque étape, le nœud successeur est choisi par validation d'une valeur de l'attribut associé aux branches. Le parcours, à travers l'arbre, commence par la racine et se termine lorsqu'une feuille est atteinte. Le nouvel objet est alors classé dans la classe étiquetant la feuille atteinte.

La Figure 2.6.1 illustre ce processus de navigation. Le nouvel objet o_{15} est inséré au niveau de la racine, les attributs test sont validés un à un selon la modalité (valeur dans le cas quantitatif) vérifiée par o_{15}. La feuille atteinte est étiquetée par la classe *Jouer=Non* donc l'objet o_{15} est classé dans cette classe.

1. C4.5 : `http://www.rulequest.com/Personal/`
2. C5.0 : `http://rulequest.com/download.html`

	Algorithmes		
	ChAID [Kass 80]	CART [Breiman 84]	C4.5 [Quinlan 93]
Critère de division	χ^2	Indice de Gini	Gain ratio
Particularités	- M-aire - Méthode de fusion de nœuds	Binaire	- M-aire - Processus de greffe
Critère d'arrêt	Absence de division pertinente avec η fixé selon le risque de première espèce choisi	- Critère de support $\eta = 5$ (plus rarement $\eta = 1$) - Absence de division pertinente avec $\eta = 0$	Critère de support $\eta = 2$
Méthode d'élagage	Pas de post-élagage	Élagage basé sur la mesure de coût-complexité (CCP)	Élagage basé sur l'erreur (EBP) avec un niveau de confiance de 0,25
Avantages	Gestion des grandes bases de données	- Bonnes performances en classification - Pas de paramétrage complexe	- Bonnes performances en classification - Simplicité de calcul et d'interprétation
Inconvénients	- Performance en classification plus faible - Paramétrage du risque complexe		Arbre large

Table 2.7: Récapitulatif des algorithmes de construction d'arbres de classification

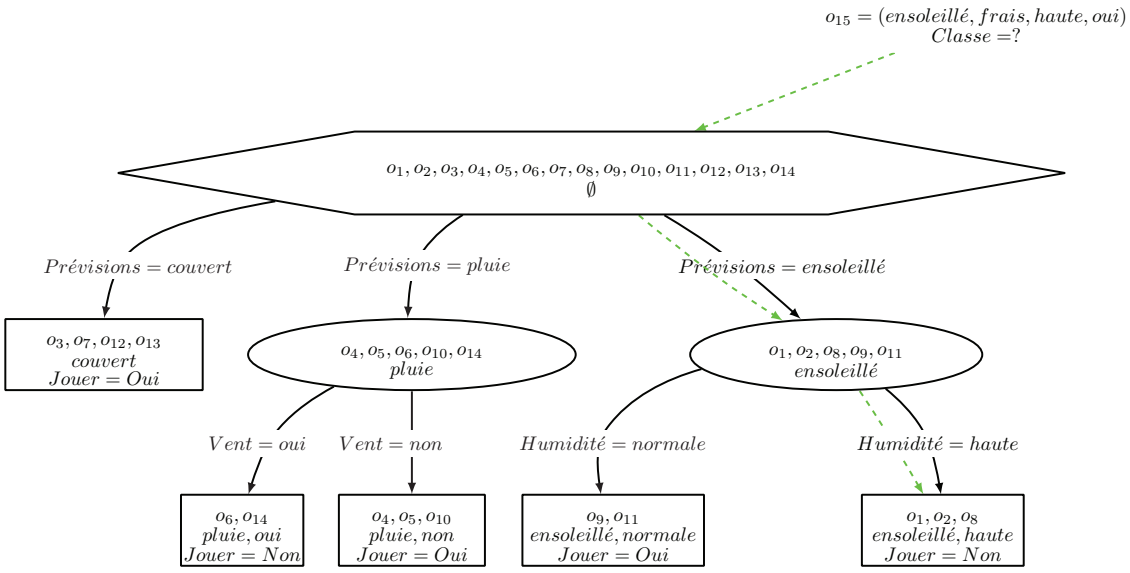


FIGURE 2.6.1: Navigation dans un arbre de classification

2.6.2 Extraction de règles de classification

L'extraction de règles de classification à partir d'un arbre de classification [Quinlan 87a, Quinlan 90], est aussi simple que la navigation. Il s'agit, pour chaque feuille de l'arbre, de générer une règle de classification du type *SI prémisse ALORS conclusion*. La prémisse de cette règle est la conjonction des différents tests composant le chemin entre la racine et cette feuille. La conclusion de la règle correspond à la classe étiquetant cette feuille.

Ainsi, à partir de l'arbre de la figure précédente, les règles sont :

- SI *Prévisions=couvert* ALORS *Jouer=Oui*
- SI *Prévisions=pluie* ET *Vent=oui* ALORS *Jouer=Non*
- SI *Prévisions=pluie* ET *Vent=non* ALORS *Jouer=Oui*
- SI *Prévisions=ensoleillé* ET *Humidité=normale* ALORS *Jouer=Oui*
- SI *Prévisions=ensoleillé* ET *Humidité=haute* ALORS *Jouer=Non*

Une fois les règles extraites, elles sont réunies dans une **base de règles**. Dans cette base, les règles sont liées par un OU logique. C'est cette base qui est utilisée pour classer un nouvel objet (*cf.* section 1.3.2.3). Les bases de règles ont certaines limites. En effet, une extraction brute telle que présentée précédemment, peut se heurter à la réplication de sous-arbres (*i.e.* un sous-arbre, avec le même attribut test et les même modalités, répliqué dans deux branches différentes) ou encore au faible effectif de certains sommets. Pour compenser ces limites, de nombreux travaux existent sur la simplification des bases de règles [Rakotomalala 96, Rakotomalala 97, Quinlan 93].

Le performances entre règles et navigation sont les mêmes. Mais, la navigation à travers

l'arbre est généralement considérée comme plus facile à comprendre que l'extraction de règles car plus visuelle. Nous reviendrons plus formellement sur la méthode de navigation dans le chapitre 4.

2.7 Discussion

2.7.1 Avantages

Parmi les avantages des arbres de classification, les plus cités sont :

1. La **lisibilité** de l'arbre de classification, qui permet d'appréhender directement la connaissance qu'il produit. De ce fait, un expert du domaine peut juger de la pertinence de l'arbre. Cet expert peut aussi intervenir dans la construction de l'arbre : choix des attributs de test, ...
2. L'arbre de classification sélectionne directement les **attributs les plus discriminants.**
3. Il est possible d'utiliser directement l'arbre pour de nouveaux objets. En effet, il n'est pas nécessaire de transformer les attributs des nouveaux objets pour pouvoir les classer.
4. L'arbre de classification peut être traduit en base de règles, facile à intégrer dans certains systèmes décisionnels.
5. Il est **rapide** en apprentissage et en classification.
6. L'arbre est non paramétrique. En effet, il ne nécessite aucune hypothèse *a priori* sur la distribution des données.
7. Il permet de traiter simultanément des attributs quantitatifs et qualitatifs (*cf.* chapitre 5).
8. Il permet une séparation linéaire par morceaux des classes, donc une résolution des problèmes non linéairement séparables. En effet, les feuilles de l'arbre correspondent à des regroupement d'objets pouvant être représentée comme un pavé dans l'espace des données.
9. Ils offrent des performances comparables aux autres méthodes supervisées [Lim 00, Rakotomalala 05a].

2.7.2 Inconvénients

Parmi les inconvénients des arbres de classification, les plus cités sont :

1. La **stabilité** de l'arbre, elle dépend des bases de données utilisées, plus la base est petite plus l'arbre est instable. Un changement même minime sur un attribut peut induire un arbre complètement différent.

2. Le passage à l'échelle : plus il y a de classes possibles, plus les performances chutent.
3. Les algorithmes sont généralement séquentiels : pas de remise en cause des étapes précédentes lors de la construction.
4. Les algorithmes sont généralement non-incrémentaux : obligation de recommencer la construction de l'arbre lors de l'ajout de nouvelles données.

2.8 Conclusion

Dans ce chapitre, nous avons présenté les arbres de classification de manière générique avant de nous intéresser plus précisément aux facteurs importants utiles à leur construction. Nous avons également présenté les algorithmes d'induction d'arbres les plus connus, avant de rappeler les points forts et les points faibles accordés aux arbres.

Les arbres de classification sont connus pour leur facilité d'interprétation, pour leur capacité à traiter différents types de données, et pour leur rapidité de fonctionnement, ce qui les rend populaires. Cependant, ils sont aussi connus pour leur instabilité vis à vis de modifications possibles sur l'ensemble de données qu'ils analysent. Notons que des études s'intéressent par exemple aux forêts d'arbres [Breiman 01] dans le but de limiter l'instabilité du modèle de classification construit, il s'agit alors de combinaisons de classifieurs.

Un certain nombre de chercheurs ont déjà évoqué le fait que la recherche pour la définition d'arbres de classification était un *problème résolu* ; d'autres s'opposent clairement à ce constat [Wu 07]. Il faut remarquer qu'il existe de nombreuses études analysant leurs performances et les différents critères qui permettent de les construire [Mingers 89a, Mingers 89b, Buntine 92, Breiman 96b, Malerba 96, Esposito 97, Breslow 97, Rakotomalala 97, Quinlan 99, Rokach 05, Patil 10]. Aujourd'hui, certains articles proposent des améliorations *via* la définition soit de nouveaux critères de division, soit de nouveaux modèles d'élagage [Do 10, Mansour 97, Loh 09]. Rakotomala précise dans son article [Rakotomalala 05a] *qu'aucune avancée significative n'a été produite en matière de taux de reconnaissance par rapport aux algorithmes de référence et qu'il paraît illusoire aujourd'hui de prétendre produire une nouvelle technique surclassant les autres.*

Comme nous le verrons dans la suite de cette thèse, nous nous intéressons aux arbres de classification en raison de leurs liens avec les treillis de Galois. Plus précisément, nous nous intéresserons à leur traitement des données quantitatives au chapitre 5, mais aussi à leur mise en œuvre de l'élagage, sur lequel nous reviendrons au chapitre 6. Avant cela présentons d'abord les treillis de Galois.

Points clés

Positionnement

- ❏ Nous avons présenté les paramètres importants de la construction d'arbres de classification.
- ❏ Les algorithmes les plus connus ont été mis en avant et différenciés.
- ❏ Les points forts et les points faibles des arbres de classification ont été rappelés.

Contributions

- ❏ Nous avons précisé les notations que nous utilisons afin de faciliter le comparatif que nous ferons avec les treillis de Galois.

Chapitre 3

Treillis

3.1 Introduction

La notion de treillis, sur laquelle reposent les treillis de Galois, date de la fin du 19éme siècle avec les ouvrages de Schröder et Dedekind [Schröder 90, Schröder 95, Dedekind 00]. Après un certain nombre d'années d'oubli, les treillis ont été le cœur de nombreux travaux de mathématiciens tels que Birkoff [Birkhoff 34a, Birkhoff 34b, Birkhoff 40, Birkhoff 48, Birkhoff 67, Birkhoff 49], Öre [Ore 44] ou Klein. En la matière, l'un des ouvrages de références, est celui de Birkhoff en 1940 [Birkhoff 40]. Dans cet ouvrage, **Birkhoff** définit **la structure de treillis** de manière *algébrique* à partir des opérations de bornes supérieure et inférieure. Quelques années plus tard, dans leur ouvrage [Barbut 70] qui fait lui aussi référence aujourd'hui, **Barbut et Monjardet** le définissent de manière plus *structurelle*, sous la forme d'un ordre possédant des éléments particuliers appelés bornes supérieure et borne inférieure. Ce livre introduit aussi la **définition des treillis de Galois**. Enfin en 1982, Wille introduit la terminologie de **treillis des concepts** [Wille 82], avant de produire un livre sur le sujet avec Ganter en 1999 [Ganter 99], livre considéré lui aussi comme une référence incontournable. Dans ce livre, **Ganter et Wille** s'intéressent plus précisément à **l'analyse formelle des concepts** (AFC - *Formal Concept Analysis FCA*), qui est une méthode de représentation des connaissances. Dès lors, les treillis de Galois sont utilisés pour l'analyse de données et de nombreux auteurs s'intéressent à cette structure, qui démontre des propriétés très intéressantes pour ce domaine de recherche [Oosthuizen 88, Liquière 90, Sahami 95, Polaillon 98, Njiwoua 99, Kuznetsov 01a, Kuznetsov 04, Guillas 07, Roth 08, Kaytoue 11, Visani 11, Domenach 12, Szathmary L. 12, Ganter 13, ...].

La structure de treillis de Galois, avec une complexité qui est dans le pire des cas, exponentielle en la taille des données analysées, **est plus complexe que celle des arbres de classification,** mais aussi plus robuste aux données détériorées ou bruitées [Guillas 07, Visani 11, ...]. Grâce à la montée en puissance des ordinateurs, ce domaine de recherche est en pleine émergence depuis quelques années.

Dans la suite de ce chapitre, nous nous intéressons tout d'abord aux différentes définitions liées aux treillis et treillis de Galois dans la section 3.2. Puis, la section 3.3 présente les algorithmes de construction de la structure de treillis ainsi que quelques outils logiciels de génération et de visualisation des treillis. Enfin, la section 3.4 s'intéresse plus particulièrement à l'utilisation des treillis de Galois en classification.

3.2 Définitions

La théorie des treillis repose principalement sur la notion d'ordre. Dans cette section, nous présentons, tout d'abord, les définitions et propriétés associées aux ordres, puis nous nous intéressons aux différentes définitions de treillis.

3.2.1 Les ordres

La notion d'ordre est définie sur un ensemble que nous noterons $\mathcal{X}$. Cet ensemble peut correspondre à différentes choses, en mathématiques par exemple, il est courant d'utiliser l'ensemble des nombres entiers $\mathbb{N}$. Définir un ordre sur un ensemble, c'est définir une relation d'ordre [1] $\leq$ entre les éléments de cet ensemble. Dans cette partie, nous donnons l'ensemble des définitions formelles associées aux ordres et que nous utilisons dans ce manuscrit, avant de donner un exemple reprenant ces dernières. Commençons par la définition 6 qui donne la définition formelle d'une relation d'ordre [Peirce 01, Hausdorff 78, Bourbaki 56].

Définition 6. Relation d'ordre : Une relation binaire $\leq$ sur un ensemble $\mathcal{X}$ est une **relation d'ordre** (aussi appelée ordre partiel) sur $\mathcal{X}$ si elle vérifie les propriétés de réflexivité, d'antisymétrie et de transitivité :
1- réflexivité : $\forall x \in \mathcal{X},\ x \leq x$.
2- antisymétrie : $\forall x, y \in \mathcal{X},\ si\ x \leq y\ et\ y \leq x\ alors\ x = y$.
3- transitivité : $\forall x, y, z \in \mathcal{X}\ si\ x \leq y\ et\ y \leq z\ alors\ x \leq z$.

Le couple $(\mathcal{X}, \leq)$, formé par l'ensemble $\mathcal{X}$ et la relation d'ordre $\leq$, est appelé **ensemble ordonné**.

A toute relation d'ordre $\leq$, il est possible d'associer sa **relation d'ordre strict** $<$, *i.e.* sa réduction réflexive (les relations réflexivité sont supprimées) et sa **relation de couverture** $\prec$, *i.e.* sa réduction réflexive et transitive (les relations de réflexivité et de transitivité sont supprimées). La relation de couverture est définie pour deux éléments x, y de l'ensemble $\mathcal{X}$, lorsque $x < y\ et\ \nexists z \in \mathcal{X}\ tel\ que\ x < z < y$. On dit alors que y

1. Les relations d'ordre sont parfois notées O ou R mais le signe *inférieur ou égal* $\leq$, hérité des relations entre les nombres, est le plus fréquemment utilisé.

couvre x ou que x *est couvert par* y.

La relation d'ordre sur un ensemble peut être reconstruite à partir de la relation de couverture et inversement [Goralčíková 79].

Un ensemble ordonné peut être représenté sous la forme d'un diagramme où les éléments de $\mathcal{X}$ sont les nœuds et les arcs représentent la relation d'ordre $\leq$ entre deux éléments, lorsqu'on représente seulement la relation de couverture, on parle de **diagramme de Hasse**, ce dernier est plus simple et plus lisible que le diagramme complet.

Définition 7. Diagramme de Hasse : Le diagramme de Hasse d'un ensemble ordonné est une réduction transitive et réflexive du graphe de sa relation d'ordre. C'est une représentation de son graphe de couverture.

Dans un ensemble ordonné, il existe des éléments particuliers tels que les **majorants** (ou *successeurs*) et les **minorants** (ou *prédécesseurs*), ces derniers sont définis comme suit :

Définition 8. Majorants et minorants : Soit un ensemble ordonné $(\mathcal{X}, \leq)$, avec $\leq$ est une relation d'ordre :
- Pour tout élément $x \in \mathcal{X}$, **l'ensemble des majorants** de x est l'ensemble $Sup_x = \{y \in \mathcal{X}; x \leq y\}$.
- Pour tout élément $x \in \mathcal{X}$, **l'ensemble des minorants** de x est l'ensemble $Inf_x = \{y \in \mathcal{X}; y \leq x\}$.

Exemple. Soit l'ensemble $\mathcal{X} = \{a, b, c, d, e, f, g\}$. La Figure 3.2.1 représente la relation d'ordre $\leq = \{(a, b), (a, c), (b, d), (b, e), (c, e), (d, f), (d, g), (e, f), (e, g)\}$ sur $\mathcal{X}$. Chaque élément de l'ensemble est représenté par un sommet, un nœud dans le graphe. Lorsque la relation d'ordre est vérifiée entre deux éléments de $\mathcal{X}$, celle-ci est représentée par un arc liant les deux nœuds représentant chaque élément. Les arcs en tirets gris représentent la relation de transitivité tandis que les arcs en pointillés noirs représentent la relation de réflexivité. De plus, pour tout couple d'éléments de $\mathcal{X}$, la relation d'antisymétrie est respectée.

Illustrons maintenant les majorants, les minorants et les bornes supérieure et inférieure. Prenons l'élément $b \in \mathcal{X}$; alors $Sup_b = \{b, d, e, f, g\}$ (*cf.* sous-graphe dans le rectangle noir de la Figure 3.2.1) et $Inf_b = \{a, b\}$. Prenons l'élément $c \in \mathcal{X}$; $Sup_c = \{c, d, e, f, g\}$ et $Inf_c = \{a, c\}$.

La Figure 3.2.2 représente le diagramme de Hasse associé à l'ordre de la Figure 3.2.1.

Remarque. Dans la littérature, il est courant de représenter les diagrammes de Hasse dans **le sens inverse** (*i.e.* arcs orientés vers le haut). **Nous choisissons de les représenter tel que dans la Figure 3.2.2, à des fins de comparaison avec les arbres de classification**.

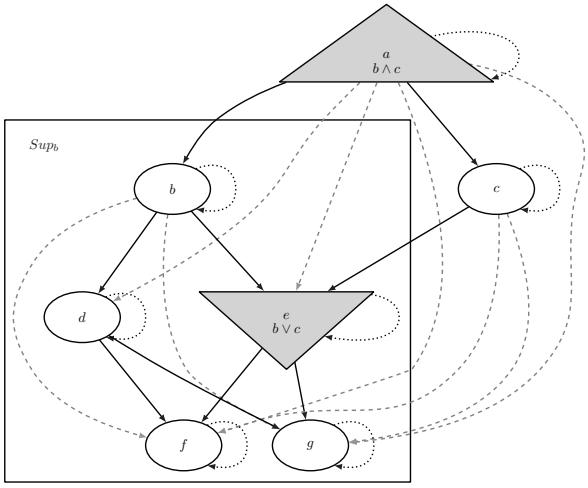


FIGURE 3.2.1: Exemple d'ensemble ordonné

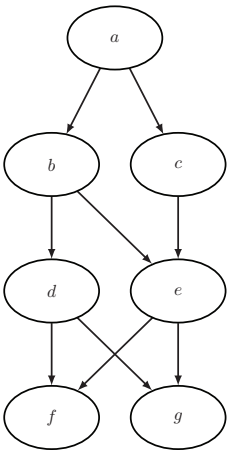


FIGURE 3.2.2: Diagramme de Hasse associé à l'ensemble ordonné de la Figure 3.2.1

Intéressons-nous maintenant aux définitions de la théorie des treillis, en lien avec les définitions précédentes.

3.2.2 Les treillis

Les treillis ont été définis de deux manières, ces deux définitions utilisent les notions de bornes supérieure et inférieure. En 1940, Birkhoff [Birkhoff 40] les définit *algébriquement* à partir d'opérateurs de bornes inférieure et supérieure (*cf.* definition 9). Puis en 1970, Barbut et Monjardet [Barbut 70] donnent une définition *structurelle* (*cf.* definition 10) sous forme d'un graphe possédant non seulement la propriété d'ordre (*cf.* définition 6) mais des éléments particuliers que sont la borne supérieure et la borne inférieure (*cf.* définition 10) :

Définition 9. Définition algébrique [Birkhoff 40] : Un treillis est un triplet $\mathcal{L} = (\mathcal{X}, \vee, \wedge)$ où $\vee$ et $\wedge$ sont deux opérateurs binaires sur l'ensemble $\mathcal{X}$ qui vérifient les propriétés suivantes :
- associativité : $\forall x, y, z \in \mathcal{X},\ (x \vee y) \vee z = x \vee (y \vee z)$ et $(x \wedge y) \wedge z = x \wedge (y \wedge z)$.
- commutativité : $\forall x, y \in \mathcal{X},\ x \vee y = y \vee x$ et $x \wedge y = y \wedge x$.
- idempotence : $\forall x \in \mathcal{X},\ x \vee x = x = x \wedge x$.
- loi d'absorption : $\forall x, y \in \mathcal{X},\ x \vee (x \wedge y) = x = x \wedge (x \vee y)$.

Définition 10. Définition structurelle [Barbut 70] : Un treillis est un ensemble $\mathcal{X}$ muni d'une relation d'ordre $\leq$, *i.e.* c'est une paire $\mathcal{L} = (\mathcal{X}, \leq)$ où toute paire $\{x, y\}$ d'éléments de $\mathcal{X}$ admet une borne inférieure $x \wedge y$ et une borne supérieure $x \vee y$ dans $\mathcal{X}$, avec :

- La borne supérieure de x et y**,** aussi appelée *supremum* et notée $\vee$, est l'unique élément minimal, de l'ensemble des majorants communs (*i.e.* le plus petit majorant commun) :

$$x \vee y = Min\,\{Sup_x \cap Sup_y\} = Min\,\{z \in \mathcal{X}; x \leq z \text{ et } y \leq z\}$$

- La borne inférieure de x et y**,** aussi appelée *infimum* et notée $\wedge$, est l'unique élément maximal, de l'ensemble des minorants communs (*i.e.* plus grand minorant commun) :

$$x \wedge y = Max\,\{Inf_x \cap Inf_y\} = Max\,\{z \in \mathcal{X}; z \leq x \text{ et } z \leq y\}$$

Remarque. Notons que les bornes inférieure et supérieure se généralisent aux parties d'un ensemble $\mathcal{X}$, notées $\mathcal{P}(\mathcal{X})$. Soit $X \subseteq \mathcal{P}(\mathcal{X})$ une partie de $\mathcal{X}$, sa borne inférieure, notée $\wedge X$, est l'unique élément maximal de l'ensemble des minorants de X. La borne supérieure de X, notée $\vee X$, est l'unique élément minimal de l'ensemble des majorants de X (*cf.* exemple ci-après).

Cette définition induit pour tout treillis l'existence **d'un unique élément minimal**, et **d'un unique élément maximal**. L'élément minimal aussi appelé bottom et noté $\perp$, vérifie $\forall x \in \mathcal{X}\ \perp \leq x$. Tandis que l'élément maximal aussi appelé top et noté $\top$, vérifie $\forall x \in \mathcal{X}\ x \leq \top$.

Exemple. Sur la Figure 3.2.1, considérons le couple d'éléments (c, b) alors leur borne supérieure est $b \vee c = e$ (*cf.* triangle inversé gris) et leur borne inférieure est $b \wedge c = a$ (*cf.* triangle gris). De plus, soit $X = \{a, b, c, e\}$ une partie de $\mathcal{X}$, alors $\vee X = e$ et $\wedge X = a$.

La Figure 3.2.2 précédente représente un ensemble ordonné qui n'est pas un treillis. En effet, la borne supérieure de f et g n'existe pas dans $\mathcal{X}$. La Figure 3.2.3 représente, quant à elle, le diagramme de Hasse d'un treillis.

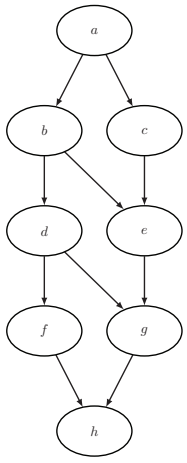


FIGURE 3.2.3: Exemple de treillis

Lorsque seule l'existence de la borne inférieure (resp. de la borne supérieure) est vérifiée pour un ensemble $\mathcal{X}$, alors $\mathcal{L}$ est appelé **inf-demi-treillis** (resp. **sup-demi-treillis**). Ainsi, un treillis est à la fois un inf-demi-treillis et un sup-demi-treillis.

Remarque. Notons une distinction importante faite entre treillis et **treillis complet** (ou fini) [Birkhoff 40, Birkhoff 49]. Un **treillis** $\mathcal{L} = (\mathcal{X}, \leq)$ est dit **complet** lorsque, pour tout sous-ensemble X d'éléments de $\mathcal{X}$, la borne inférieure $\wedge X$ et la borne supérieure $\vee X$ existent. **Les études présentées dans ce manuscrit se placent systématiquement dans le cas des treillis complets. Aussi, lorsque nous notons « treillis » nous désignons les treillis complets.**

Dans un treillis, il existe des **éléments particuliers**, parmi lesquels les éléments irréductibles et les atomes et coatomes. Ils sont définis comme suit.

Définition 11. Éléments particuliers d'un treillis $\mathcal{L}$

- Les **éléments irréductibles** d'un treillis sont les éléments qui ne sont ni des bornes inférieures, ni des bornes supérieures :

  - **les inf-irréductibles notés** M : ce sont les éléments qui ne sont pas bornes inférieures dans le treillis. Autrement dit, $m \in \mathcal{X}$ est un inf-irréductible (*meet-irreducible*) s'il n'est borne inférieure d'aucune partie ne le contenant pas, *i.e.* pour toute partie $X \subset \mathcal{X}$, $m = \wedge X$ implique $m \in X$.

  - **les sup-irréductibles notés** J : ce sont les éléments qui ne sont pas bornes supérieures dans le treillis. Autrement dit, $j \in \mathcal{X}$ est un sup-irréductible (*join-irreducible*) s'il n'est borne supérieure d'aucune partie ne le contenant pas , *i.e.* pour toute partie $X \subset \mathcal{X}$, $j = \vee X$ implique $j \in X$.

- Les **coatomes (resp. les atomes) :** Les coatomes (resp. les atomes) d'un treillis sont les éléments couverts par le top $\top$ (resp. couvrant le $\bot$).

Par définition, les inf-irréductibles sont couverts par un seul élément, *i.e.* ils **ont un unique successeur** m^+, et les sup-irréductibles couvrent un seul élément, *i.e.* ils ont un unique prédécesseur j^-. Notons en particulier que $\bot = \vee\emptyset$ et que $\top = \wedge\emptyset$, ainsi $\bot$ n'est pas un sup-irréductible et $\top$ n'est pas un inf-irréductible.

Exemple. La Figure 3.2.4 présente ces éléments irréductibles. Les inf-irréductibles sont les concepts grisés, les sup-irréductibles sont les concepts en formes rectangulaires et les coatomes sont les concepts en forme de losanges.

Remarque. Notons que, par définition, les coatomes sont des inf-irréductibles, mais l'inverse n'est pas toujours vrai, comme l'illustre la Figure 3.2.5.

De plus, tout élément d'un treillis peut être simultanément inf-irréductible et sup-irréductible. La Figure 3.2.6 présente le cas particulier du treillis M_3 pour lequel $M = J$.

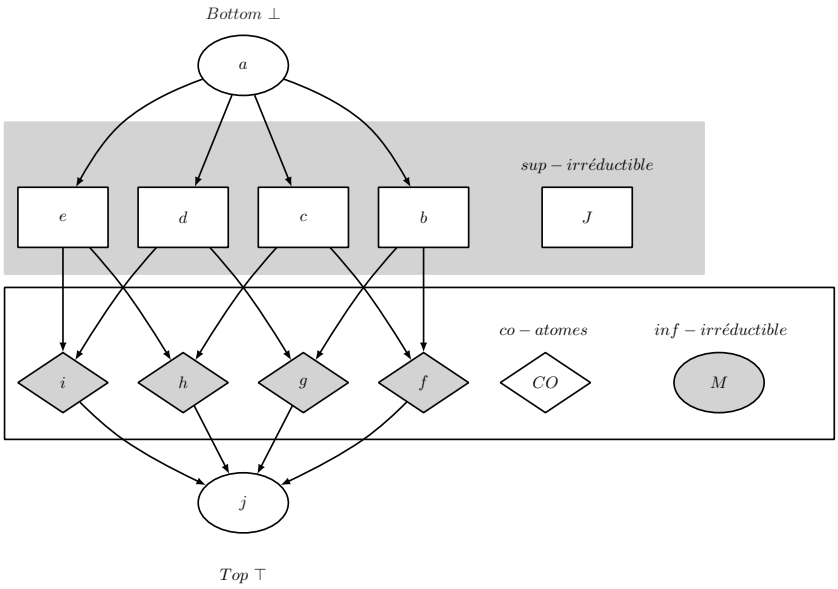


FIGURE 3.2.4: Diagramme de Hasse d'un treillis et ses éléments remarquables

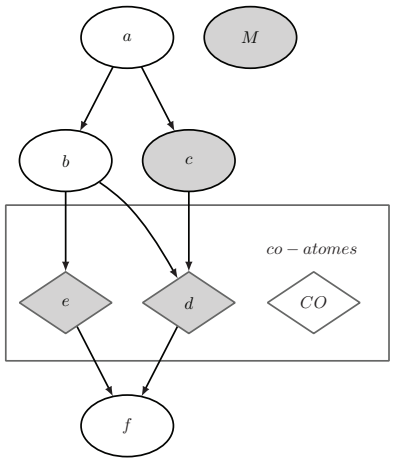


FIGURE 3.2.5: Différence entre les coatomes et les inf-irréductibles d'un treillis

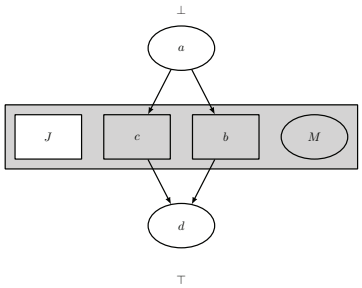


FIGURE 3.2.6: Treillis M_3

A partir de ces éléments particuliers, il est possible de construire **la table binaire** associée à un treillis, c'est une représentation compacte du treillis. Elle contient en ligne les éléments de M et en colonne ceux de J. Chaque cellule (m, j) de la table indique si il existe une relation d'ordre entre m et j, *i.e.* la cellule (m, j) vaut « vrai » si $j \leq m$, « faux » sinon.
La table 3.1 est la table binaire associée au treillis de la Figure 3.2.4.

M \ J	b	c	d	e
f	×	×		
g	×		×	
h		×		×
i			×	×

TABLE 3.1: Table binaire associée au treillis de la Figure 3.2.4

Treillis particuliers

Dans ce paragraphe, nous nous intéressons à quelques classes de treillis que nous utilisons dans la suite de ce manuscrit.

Les treillis complémentés

Définition 12. Treillis complémenté : Un treillis $\mathcal{L}$ est *complémenté* si tout élément $x \in \mathcal{X}$ admet au moins un *complément*, *i.e.* un élément $\bar{x} \in \mathcal{X}$ tel que $x \vee \bar{x} = \top$ et $x \wedge \bar{x} = \bot$.

De manière restrictive, un treillis $\mathcal{L}$ est *sup-complémenté* (aussi noté $\vee$-complémenté) lorsque pour tout élément $x \in \mathcal{X}$, il existe un *sup-complément*, *i.e.* un élément $\bar{x} \in \mathcal{X}$ tel que $x \vee \bar{x} = \top$.

Les treillis coatomistiques

Définition 13. Treillis coatomistique : Un treillis est coatomistique si tout élément inf-irréductible est un coatome.

Exemple. Dans la Figure 3.2.5, les inf-irréductibles n'étant pas tous des coatomes, le treillis n'est donc pas coatomistique. La Figure 3.2.7 présente un treillis coatomistique ; sa table binaire est présentée dans la table 3.2.

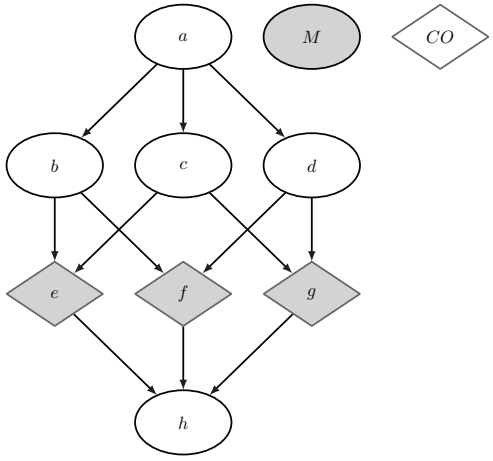


FIGURE 3.2.7: Exemple de treillis coatomistique

M \ J	b	c	d
e	×	×	
f	×		×
g		×	×

TABLE 3.2: Table associée au treillis coatomistique de la Figure 3.2.7

Cette première partie avait pour objectif de présenter les fondements mathématiques liés à la théorie des treillis. La partie suivante présente les treillis utilisés en analyse de données, à savoir les treillis de Galois ou treillis des concepts.

3.2.3 Treillis de Galois ou des concepts

Un treillis de Galois (aussi appelé treillis des concepts) est défini à partir d'une table de données binaire appelée **contexte formel**. Ainsi, ce type de treillis est défini à partir de deux ensembles. Dans le cadre de l'analyse de données, le premier ensemble correspond à

l'ensemble des objets noté O et le second correspond à un ensemble d'attributs binaires [2] $\mathcal{I}$. Les liens unissant les objets à leurs attributs sont définis par deux applications α et β. Le couple (α, β) forme une **correspondance de Galois** d'une relation binaire. La notion de correspondance de Galois a été introduite dès 1940 par Birkoff [Birkhoff 40], sous le nom de *polarité.* C'est en 1944 que Öre la renomme correspondance de Galois et, dans son étude approfondie [Ore 44], lui associe sa notation abstraite sous forme d'un couple d'applications. Plus formellement, ces applications et un contexte formel sont définis comme suit :

Définition 14. Contexte formel : Un contexte est un triplet $(O, \mathcal{I}, (\alpha, \beta))$:
- O : un ensemble d'objets,
- $\mathcal{I}$: un ensemble d'attributs,
- (α, β) : une correspondance de Galois avec
  - α est une application de $\mathcal{P}(O)$ vers $\mathcal{P}(\mathcal{I})$, elle associe à un ensemble d'objets, leurs attributs communs. Soit un ensemble d'objets $A \subseteq O$ et R une relation binaire entre O et $\mathcal{I}$:

$$\alpha(A) = \{b \in \mathcal{I} \mid aRb \ \forall a \in A\} \tag{3.2.1}$$

- β est une application de $\mathcal{P}(\mathcal{I})$ vers $\mathcal{P}(O)$, elle associe à un ensemble d'attributs, les objets qui les vérifient. Soit un ensemble d'attributs $B \subseteq \mathcal{I}$:

$$\beta(B) = \{a \in O \mid aRb \ \forall b \in B\} \tag{3.2.2}$$

Dans la littérature, le contexte est parfois noté $(O, \mathcal{I}, R)$. Dans les ouvrages tels que ceux de Ganter et Wille [Wille 82, Ganter 84, Ganter 97, Ganter 99], la notation la plus utilisée est (G, M, I). Dans ce cas, G correspond à O (pour *Gegenstände* qui signifie objets en allemand), M correspond à $\mathcal{I}$ (pour *Merkmale* qui signifie caractéristiques en allemand) et la relation binaire y est donc notée I. Dans ces ouvrages, les applications α et β, de la correspondance de Galois, sont représentées par un seul opérateur *prime*, *i.e.* $\alpha(A) = A'$ et $\beta(B) = B'$.

Exemple. La Table 3.3 donne un exemple de contexte formel. Il s'agit de la table de données utilisée dans le chapitre précédent (Table 2.1). Cette table a été *binarisée*, *i.e.* chaque modalité des attributs qualitatifs est considérée comme un attribut binaire. Nous avons retiré la classe, qui n'est pas utile pour le moment.

Illustrons les deux applications α et β. Soit l'objet o_1, alors $\alpha(o_1)$ =\{ensoleillé, chaud, haute\}. Pour l'ensemble d'objets $\{o_6, o_7\}$, $\alpha(\{o_6, o_7\})$ =\{frais, normale,Vent\} (représenté en gris clair dans la Table 3.3). Inversement, pour l'attribut binaire *pluie*, $\beta(pluie) = \{o_4, o_5, o_6, o_{10}, o_{14}\}$. Pour l'ensemble d'attributs $\{ensolleillé, chaud\}$, $\beta(\{$ensoleillé, chaud$\}) = \{o_1, o_2\}$(représenté en gris foncé dans la Table 3.3).

2. Tout attribut qualitatif peut être *binarisé*, *i.e.* chaque modalité devient un attribut booléen.

O \ $\mathcal{I}$	Prévisions			Température			Humidité		Vent
	couvert	pluie	ensoleillé	chaud	doux	frais	normale	haute	
o_1			×	×				×	
o_2			×	×				×	×
o_3	×			×				×	
o_4		×			×			×	
o_5		×				×	×		
o_6		×				×	×		×
o_7	×					×	×		×
o_8			×		×			×	
o_9			×			×	×		
o_{10}		×			×		×		
o_{11}			×		×		×		×
o_{12}	×				×			×	×
o_{13}	×			×			×		
o_{14}		×			×			×	×

TABLE 3.3: Exemple de contexte

Chaque correspondance maximale objets-attributs, définie via α et β, forme un nœud appelé **concept** ou concept formel dans le treillis correspondant. Un concept correspond à un rectangle maximal dans le contexte associé, *i.e.* un rectangle maximal formé de cases contenant toutes des croix. Les concepts sont définis formellement comme suit.

Définition 15. Concept : Un concept du contexte $(O, \mathcal{I}, (\alpha, \beta))$ est une paire (A, B) avec $A \subseteq O$, $B \subseteq \mathcal{I}$ et $A = \beta(B)$ et $B = \alpha(A)$. L'ensemble A est appelé **extension** du concept tandis que l'ensemble B est appelé **intension** du concept.

Exemple. Le couple $(\{o_1, o_2\}, \{ensolleillé, chaud\})$ n'est pas un concept. En effet, $\alpha(\{o_1, o_2\}) = \{ensolleillé, chaud, haute\}$. Le couple $(\{o_1, o_2\}, \{ensolleillé, chaud, haute\})$ est bien un concept, car $\beta(\{ensolleillé, chaud, haute\}) = \{o_1, o_2\}$ et réciproquement.

Ainsi, à partir d'un contexte, il est possible de définir un ensemble de concepts, que nous notons $\mathcal{C}$. Ces concepts respectent un ordre, selon la relation d'inclusion. Le treillis de Galois ou des concepts est défini à partir de cet ensemble et de cette relation.

Définition 16. Treillis de Galois : Le treillis de Galois du contexte $(O, \mathcal{I}, (\alpha, \beta))$ est le couple $\mathcal{L} = (\mathcal{C}, \leq)$ où :

- $\mathcal{C}$ est l'ensemble de concepts du contexte formel $(O, \mathcal{I}, (\alpha, \beta))$,
- $\leq$ est une relation d'ordre (extension/subsomption) entre concepts, *i.e.* soit $(A_1, B_1), (A_2, B_2) \in \mathcal{C}$ deux concepts :

$$(A_1, B_1) \leq (A_2, B_2) \Leftrightarrow A_2 \subseteq A_1 \Leftrightarrow B_2 \supseteq B_1 \tag{3.2.3}$$

- (A_1, B_1) est appelé **sur-concept** de (A_2, B_2)
- (A_2, B_2) est appelé **sous-concept** de (A_1, B_1)

Remarque. Dans les ouvrages de référence, tels que [Ganter 99] où les auteurs notent $\mathcal{L}$ sous la forme $\underline{\mathcal{B}}(G, M, I)$ ou encore [Caspard 07] où les auteurs notent $\mathcal{L}$ sous la forme $(Gal(O, \mathcal{I}, R), \leq)$, la relation d'ordre est généralement inverse à celle présentée dans la définition précédente, *i.e.* $(A_1, B_1) \leq (A_2, B_2) \Leftrightarrow A_1 \subseteq A_2 \Leftrightarrow B_1 \supseteq B_2$. L'équivalence entre ces deux relations ayant été démontrée entre autres dans [Caspard 07]. Nous utilisons **la relation d'ordre $\leq$ telle que présentée dans la définition 16**, à des fins de comparaison avec les arbres de classification (*cf.* chapitre 2).

Les éléments particuliers présentés dans la section précédente ont une formulation particulière lorsqu'ils sont définis à partir d'un treillis de Galois.

Tout d'abord, les bornes supérieure et inférieure dans un treillis de Galois $\mathcal{L}$ sont des concepts définis à partir de deux concepts $(A_1, B_1), (A_2, B_2) \in \mathcal{C}$, comme suit :

$$(A_1, B_1) \vee (A_2, B_2) = ((A_1 \cap A_2), \alpha(A_1 \cap A_2)) \tag{3.2.4}$$

$$(A_1, B_1) \wedge (A_2, B_2) = (\beta(B_1 \cap B_2), (B_1 \cap B_2)) \tag{3.2.5}$$

L'élément minimal d'un treillis de Galois est le concept $\perp = (O, \alpha(O))$, et l'élément maximal d'un treillis de Galois est le concept $\top = (\beta(\mathcal{I}), \mathcal{I})$.

Un même treillis de Galois peut s'obtenir à partir de plusieurs tables binaires. On dit alors qu'elles sont équivalentes. Parmi celles-ci se trouve un unique *contexte réduit* possédant un nombre minimal d'objets et d'attributs. Dans la littérature, c'est généralement le contexte réduit qui est utilisé pour introduire les différentes propriétés des treillis et contextes.

Le *contexte réduit* $(O, \mathcal{I}, (\alpha, \beta))^r$ est construit à partir d'une table binaire $(O, \mathcal{I}, (\alpha, \beta))$, en :

1. Supprimant tout objet $o_1 \in O$ possédant les mêmes attributs qu'un autre objet $o_2 \in O$ du contexte (*i.e.* $\alpha(o_1) = \alpha(o_2)$)
2. Supprimant tout attribut $I_1 \in \mathcal{I}$ vérifié par les mêmes objets qu'un autre attribut $I_2 \in \mathcal{I}$ du contexte (*i.e.* $\beta(I_1) = \beta(I_2)$)
3. Supprimant tout objet $o \in O$ possédant les mêmes attributs qu'un ensemble d'objets $O' \subset O$ du contexte (*i.e.* $\alpha(o) = \alpha(O')$)
4. Supprimant tout attribut $I \in \mathcal{I}$ vérifié par les mêmes objets qu'un ensemble d'attributs $I' \subset \mathcal{I}$ du contexte (*i.e.* $\beta(I) = \beta(I')$)
5. Supprimant tous les objets ne possédant aucun attribut
6. Supprimant tous les attributs vérifiés par aucun objet

Les éléments irréductibles s'organisent sous forme d'une table binaire qui est réduite. Ils décrivent la structure même du treillis et permettent sa reconstruction.

Proposition 1. *[Barbut 70] : Tout treillis est isomorphe au treillis de Galois de sa table binaire.*

L'isomorphisme entre le treillis de Galois de la table binaire (ou contexte formel) et le treillis de Galois du contexte réduit est illustrée par les Figures 3.2.8 et 3.2.9.

La proposition 1 permet donc de **mettre en relation les objets et attributs du contexte avec les éléments irréductibles** [Birkhoff 67] **:**

Proposition 2. *Chaque inf-irréductible est le plus grand concept par rapport à la relation d'ordre, contenant un (ou plusieurs) objet(s) (cet(ces) objet(s) appartien(nen)t à la différence ensembliste entre l'extension de cet inf-irréductible et l'extension de son unique successeur). Ces concepts sont aussi appelés objet-concept.*

Proposition 3. *Chaque sup-irréductible est le plus petit concept par rapport à la relation d'ordre, contenant un (ou plusieurs) attributs. Ces concepts sont aussi appelé attribut-concepts.*

Ce résultat important est l'une des bases de nos développements présentés dans le chapitre 5.

Exemple. La Table 3.4(iv) présente le contexte réduit issu de la Table binaire 3.4(i). Notons que la réduction peut se faire aussi uniquement sur les objets (*cf.* Table 3.4(ii)) ou sur les attributs (*cf.* Table 3.4(iii)).

Concernant la bijection liant les objets et les inf-irréductibles, prenons l'objet o_1 dans Table 3.4(iv) ; le concept $(\beta(\alpha(o_1)), \alpha(o_1)) = (\{o_1\}, \{I_1, I_2\})$ est l'unique plus grand concept contenant o_1 dans le treillis correspondant (Figure 3.2.8) et est un inf-irréductible. Prenons l'attribut I_2 dans Table 3.4(iv), le concept $(\beta(I_2), \alpha(\beta(I_2))) = (\{o_1, o_2, o_3\}, \{I_2\})$ est l'unique plus petit concept contenant I_2 dans le treillis correspondant (Figure 3.2.8) et est un sup-irréductible concept.

A tout treillis de Galois sont associés deux treillis de fermés particuliers. Intéressons nous à ces treillis particuliers.

3.2.4 Treillis des fermés

Tout treillis de Galois peut se décomposer en deux parties, la première sur l'ensemble des objets O et la seconde sur l'ensemble des attributs $\mathcal{I}$. Chacune des parties est une famille d'ensemble de parties définie sur O (resp. sur $\mathcal{I}$) et ordonnée par $\subseteq$ (resp. $\supseteq$). $\subseteq$ et $\supseteq$ sont donc des relations d'ordre. De plus, les compositions $\beta o \alpha$ et $\alpha o \beta$ sont des opérateurs de fermeture définis respectivement sur O et $\mathcal{I}$, et les deux treillis $\mathcal{L}_O = (O, \subseteq)$

O \ $\mathcal{I}$	I_1	I_2	I_3	I_4
o_1	×	×		
o_2		×		
o_3		×	×	×
o_4			×	×

(i) Table binaire (Contexte original)

O \ $\mathcal{I}$	I_1	I_2	I_3	I_4
o_1	×	×		
o_3		×	×	×
o_4			×	×

(ii) Contexte réduit sur les objets

O \ $\mathcal{I}$	I_1	I_2	I_3
o_1	×	×	
o_2		×	
o_3		×	×
o_4			×

(iii) Contexte réduit sur les attributs

O \ $\mathcal{I}$	I_1	I_2	I_3
o_1	×	×	
o_3		×	×
o_4			×

(iv) Contexte réduit

TABLE 3.4: Réduction de contexte

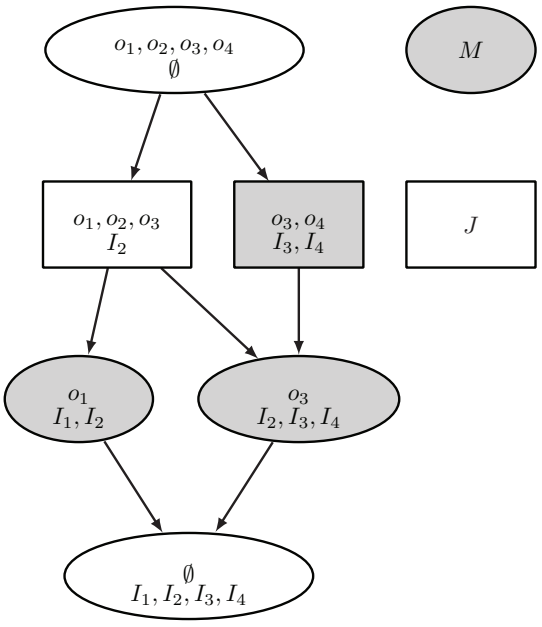


FIGURE 3.2.8: Treillis de la Table 3.4(i)

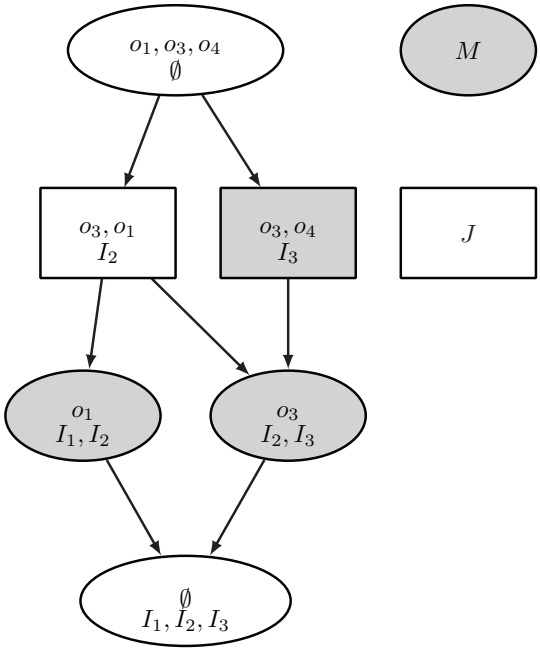


FIGURE 3.2.9: Treillis du contexte réduit (Table 3.4(iv))

et $\mathcal{L}_{\mathcal{I}} = (\mathcal{I}, \supseteq)$ sont composés de l'ensemble des fermés relativement à ces opérateurs, ces treillis sont appelés treillis des fermés, où :

Un **opérateur de fermeture** sur un ensemble $\mathcal{X}$, est une application sur $\mathcal{P}(\mathcal{X})$, notée φ, qui vérifie les propriétés suivantes :

1. φ est idempotent : $\forall X \in \mathcal{X}$, $\varphi^2(X) = \varphi(X)$,
2. φ est extensif : $\forall X \in \mathcal{X}$, $X \subseteq \varphi(X)$,
3. φ est isotone : $\forall X, X' \in \mathcal{X}$, $X \subseteq X' \Rightarrow \varphi(X) \subseteq \varphi(X')$.

A partir d'un opérateur de fermeture, on définit les **fermés** de $\mathcal{X}$ par :

Un sous-ensemble $X \in \mathcal{X}$ est dit fermé si et seulement si $X = \varphi(X)$.

Ainsi, si (A_1, B_1) est un concept du treillis de Galois, alors A_1 **est un fermé de** O (*i.e.* $\varphi(A_1) = \beta o \alpha(A_1) = \beta(B_1) = A_1$) et B_1 **est un fermé de** $\mathcal{I}$ (*i.e.* $\varphi(B_1) = \alpha o \beta(B_1) = \alpha(A_1) = B_1$).

Le **treillis des fermés** est alors défini à partir d'un ensemble de fermés, muni de la relation d'inclusion.

Définition 17. Treillis de fermés : Un treillis de fermés sur un ensemble $\mathcal{X}$ est **une paire** $(\mathcal{F}, \subseteq)$ où $\mathcal{F}$ est une famille de fermés de $\mathcal{X}$ pour un opérateur de fermeture , ordonnés par inclusion.

La famille $\mathcal{F}$ sur O (ou $\mathcal{I}$) possède la propriété de stabilité par intersection, *i.e.* $\forall F, F' \in \mathcal{F} \Rightarrow F \cap F' \in \mathcal{F}$. On parle aussi **de famille de Moore** [Moore 09].

Les Figures 3.2.10, 3.2.11 représentent $\mathcal{L}_O$ et $\mathcal{L}_{\mathcal{I}}$ du contexte de la Table 3.5. Le diagramme de Hasse du treillis de Galois correspondant est présenté dans la Figure 3.2.12.

O \ $\mathcal{I}$	I_1	I_2	I_3	I_4
o_1	×	×		
o_2		×		
o_3		×	×	
o_4			×	×

TABLE 3.5: Exemple de contexte formel

Après cette présentation des définitions liées à la théorie des treillis, la section suivante présente plus précisément des algorithmes de construction de treillis de Galois.

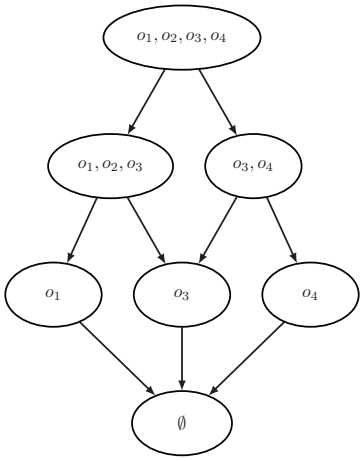


FIGURE 3.2.10: Diagramme de Hasse du treillis de fermés associé au contexte de la Table 3.5 - $\mathcal{L}_O = (O, \subseteq)$

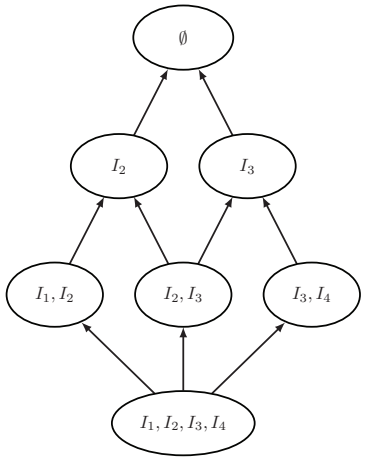


FIGURE 3.2.11: Diagramme de Hasse du treillis de fermés associé au contexte de la Table 3.5 - $\mathcal{L}_\mathcal{I} = (\mathcal{I}, \subseteq)$

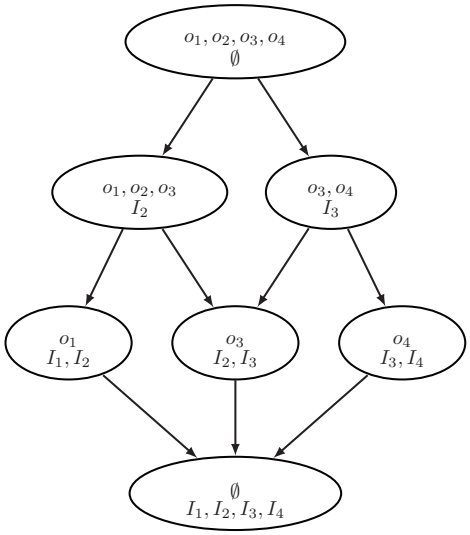


FIGURE 3.2.12: Diagramme de Hasse du treillis de Galois associé au contexte de la Table 3.5 - $\mathcal{L} = (\mathcal{C}, \leq)$

3.3 Algorithmes de génération et de visualisation

Comme nous l'avons mentionné précédemment, en analyse de données, ce sont les treillis de Galois ou des concepts qui sont utilisés pour représenter des données objets × attributs. **Le treillis de Galois** d'un contexte $(O, \mathcal{I}, (\alpha, \beta))$ peut avoir **une taille exponentielle en fonction des données utilisées**, cette taille est bornée dans le pire cas par $2^{|O+\mathcal{I}|}$ et dans le meilleur cas par $|O + \mathcal{I}|$. La génération du treillis est un problème dit $\mathcal{NP}$-difficiles en temps. Les algorithmes de génération des treillis de Galois ont une complexité algorithmique polynomiale par concept construit. **Dans la pratique, la taille du treillis reste raisonnable** [Kuznetsov 01b, Mephu-Nguifo 05]. De plus, la récente et constante montée en puissance des ordinateurs permet de construire les treillis de Galois contenant de plus en plus de concepts.

Dans la littérature, il existe différents algorithmes de construction d'un treillis de Galois. Parmi ces algorithmes se trouvent l'algorithme de Chein [Chein 69], celui de Ganter [Ganter 84] (NextClosure), celui de Kuznetsov [Kuznetsov 93, Kuznetsov 02] (CloseByOne), celui de Bordat [Bordat 86] ou encore celui de Nourine et Raynaud [Nourine 99], qui présente la meilleure complexité théorique (complexité quadratique par concept). Dans la liste des algorithmes existants, nous pouvons ajouter des algorithmes incrémentaux tels que celui de Norris [Norris 78], de Oosthuizen [Oosthuizen 88], de Godin *et al.* [Godin 91] ou encore de Carpineto et Romano [Carpineto 93].

Les paragraphes suivants présentent les deux plus connus que sont l'algorithme NextClosure développé par Ganter [Ganter 84] et l'algorithme de Bordat [Bordat 86] qui calcule directement le diagramme de Hasse. C'est ce dernier algorithme que nous utilisons dans les chapitres suivants et pour l'ensemble de nos expérimentations.

3.3.1 Algorithme NextClosure

NextClosure a été développé par Ganter en 1984. Le principe de base de cet algorithme est la construction itérative de fermés d'un ensemble $\mathcal{X}$, selon un opérateur de fermeture φ, et selon l'ordre lectique à partir du fermé minimal $\varphi(\emptyset)$, et jusqu'à ce que le fermé maximal $\mathcal{X}$ soit généré. L'ordre lectique, noté $<_i$ est défini pour l'ensemble $\mathcal{X} = \{x_1, x_2, \ldots, x_m\}$ d'éléments indicés, et pour deux fermés $X_1, X_2 \subseteq \mathcal{P}(\mathcal{X})$, par $X_1 <_i X_2$ si l'élément de plus petit indice qui distingue X_1 et X_2 appartient à X_2 :

$$X_1 <_i X_2 \textit{ ssi } \exists x_i \in X_2 \setminus X_1 \textit{ tel que } X_1 \cap \{x_1, \ldots x_{i-1}\} = X_2 \cap \{x_1, \ldots x_{i-1}\}$$

Cet ordre lectique étend l'ordre d'inclusion, en effet si $X_1 \subset X_2$ alors il existe $i \leq m$ tel que $X_1 <_i X_2$.

Exemple. Soit l'ensemble $\mathcal{X} = \{a, b, c\}$, $\{a, c\} <_1 \{b, c\}$ ou encore $\{a, b\} <_3 \{a, b, c\}$.

Dans l'article [Ganter 84], l'auteur décrit l'algorithme **NextClosure calculant le fermé suivant d'un fermé selon l'ordre lectique**, qu'il étend à **NextConcept** où les fermés sont complétés avec l'extension des concepts.

L'Algorithme 3.2 présente l'algorithme de base NextClosure prenant donc en paramètres un ensemble $\mathcal{X}$, un fermé X de $\mathcal{X}$, et un opérateur de fermeture φ. La génération complète du treillis de Galois d'un contexte $(O, \mathcal{I}, (\alpha, \beta))$ correspond à l'Algorithme 3.1. Ce dernier utilise NextClosure pour calculer les fermés sur l'ensemble des attributs, en commençant avec le plus petit fermé, *i.e.* $(\beta(\emptyset), B)$. Puis, il faut ensuite ordonner ces concepts selon la relation $\leq$ (*cf.* définition 16) en les comparant deux à deux. Une fermeture transitive permet d'obtenir le diagramme de Hasse [Goralčíková 79]. Dans [Godin 95b], Godin *et al.* combinent l'algorithme NextClosure à l'algorithme proposé par Alaoui [Alaoui 93] pour construire directement le diagramme de Hasse à partir de l'ensemble de fermés calculé par NextClosure.

Algorithme 3.1 Construire_Treillis$((O, \mathcal{I}, (\alpha, \beta)))$

Entrées :
$(O, \mathcal{I}, (\alpha, \beta))$: un contexte ;
Sorties :
$(\mathcal{C}, \leq)$: le treillis de Galois du contexte $(O, \mathcal{I}, (\alpha, \beta))$;

DÉBUT

$B \leftarrow \alpha(\beta(\emptyset))$;
Ajouter $(\beta(\emptyset), B)$ à $\mathcal{C}$;
//Calcul des fermés
tant que $B \neq \mathcal{I}$ **faire**
  $B \leftarrow$ NextClosure($\mathcal{I}$,B,$\alpha o \beta$) ;
  Ajouter $(\beta(B), B)$ à $\mathcal{C}$;
fin tant que
//Calcul de la relation d'ordre $\leq$
pour chaque $(A, B) \in \mathcal{C}$ non marqué **faire**
  pour chaque $(A_i, B_i) \in \mathcal{C}$ **faire**
    si $B \subset B_i$ **alors**
      Ajouter l'arc $(A, B) \leq (A_i, B_i)$;
    fin si
  fin pour
  Marquer (A, B) ;
fin pour
renvoyer $(\mathcal{C}, \leq)$;
FIN

Algorithme 3.2 NextClosure($\mathcal{X}, X, \varphi$)

Entrées :
- $\mathcal{X}$: un ensemble ;
- X : un fermé ;
- φ : un opérateur de fermeture ;

Sorties :
X^+ : le fermé suivant de X selon l'ordre lectique ;

DÉBUT

$Z \leftarrow \mathcal{X} \setminus$ X trié par ordre décroissant des indices des éléments de $\mathcal{X}$;
pour chaque $x_i \in Z$ **faire**
  $X^+ = \varphi(X \cap \{x_1, \ldots, x_{i-1}\} + x_i)$;
  si $X <_i X^+$ **alors**
    renvoyer X^+ ;
  fin si
fin pour
FIN

3.3.2 Algorithme de Bordat

L'algorithme de Bordat a été développé en 1986 [Bordat 86]. Il diffère de l'algorithme précédent car il construit directement le diagramme de Hasse du treillis de Galois à partir du concept minimal. Cet algorithme est du même type que les algorithmes de génération des arbres de classification (*cf.* Algorithme 2.2), il est dit *top-down* [Godin 95b, Kuznetsov 01b, Gély 04]. Pour construire le diagramme de Hasse du treillis de Galois, l'algorithme se base sur le théorème de Bordat [Bordat 86] (*cf.* théorème 1) et sur le corollaire associé (*cf.* corollaire 1) qui établit que les successeurs immédiats d'un concept (A, B) sont en bijection avec les sous-ensembles maximaux par inclusion de la famille $\mathcal{F}_A$ définie sur l'ensemble des objets par $\mathcal{F}_A = \{\beta(b) \cap A \ : \ b \in \mathcal{I} \setminus B\}$.

Théorème 1. *Soient (A_1, B_1) et (A_2, B_2) deux concepts d'un contexte $(O, \mathcal{I}, (\alpha, \beta))$. Alors, $(A_1, B_1) \prec (A_2, B_2)$ si et seulement si A_2 est maximalement inclus dans $\mathcal{F}_A$.*

Corollaire 1. *Soit (A, B) un concept. Il y a une correspondance exacte entre les successeurs immédiats de (A, B) et les sous-ensembles maximalement inclus de $\mathcal{F}_A$.*

L'algorithme de Bordat calcule récursivement les concepts successeurs immédiats d'un concept dans le diagramme de Hasse, en commençant avec ceux du concept minimal, *i.e.* le concept $(O, \alpha(O))$. Cela implique de vérifier lors de la construction d'un concept que ce dernier n'a pas déjà été construit en tant que successeur d'un autre concept. L'algorithme de Bordat est présenté dans l'Algorithme 3.3. Pour faciliter la lecture de l'algorithme, la génération des successeurs immédiats d'un concept est présentée dans

l'Algorithme 3.4, appelé par l'Algorithme 3.3. Différentes extensions de l'algorithme de Bordat ont été introduites dans [Bertet 07].

Algorithme 3.3 Algorithme de Bordat$((O, \mathcal{I}, (\alpha, \beta)))$

Entrées :
$(O, \mathcal{I}, (\alpha, \beta))$: un contexte ;
Sorties :
$(\mathcal{C}, \prec)$: le diagramme de Hasse du treillis de Galois du contexte $(O, \mathcal{I}, (\alpha, \beta))$;

DÉBUT

$\mathcal{C} \leftarrow \{(O, \alpha(O))\}$;
pour chaque $(A, B) \in \mathcal{C}$ non marqué **faire**
  $Succ_{(A,B)} \leftarrow$ Successeurs_Immédiats$((A, B), (O, \mathcal{I}, (\alpha, \beta)))$;
  pour chaque $(A_i, B_i) \in Succ_{(A,B)}$ **faire**
    si $(A_i, B_i) \notin \mathcal{C}$ **alors**
      Ajouter (A_i, B_i) dans $\mathcal{C}$;
      Ajouter l'arc de couverture $(A, B) \prec (A_i, B_i)$;
    fin si
  fin pour
  Marquer (A, B) ;
fin pour
renvoyer $(\mathcal{C}, \prec)$;
FIN

Les algorithmes de génération de treillis ou d'un ensemble de fermés sont étudiés dans différents articles [Ganter 84, Godin 86a, Guénoche 90, Godin 95b, Kuznetsov 01b, Kuznetsov 04, Gély 04, Fu 04b, Ben Tekaya 05, Choi 06, Bertet 11]. Ces articles étudient entre autre la complexité [3] algorithmique de différents algorithmes. Notons que l'algorithme NextClosure calcule l'ensemble des concepts d'un contexte $(O, \mathcal{I}, (\alpha, \beta))$ en $O(|O|^2|\mathcal{I}|))$ par fermé si l'opérateur de fermeture considéré est $\beta o\alpha$ et en $O(|\mathcal{I}|^2|I|))$ par fermé, si l'opérateur de fermeture considéré est $\varphi o\beta$. Pour l'algorithme de Bordat, on retrouve la même complexité en $O(|\mathcal{I}|^2|I|))$ par fermé. Pour plus de précisions à ce sujet, nous renvoyons le lecteur aux références précédentes.

Dans leurs articles [Kuznetsov 01b, Kuznetsov 02], Kuznetsov et Obiedkov indiquent que la variation du temps de calcul est liée à la variation des données. Les auteurs concluent d'ailleurs que *le choix de l'algorithme pour la construction de l'ensemble des concepts et du diagramme doit être basé sur les propriétés des données d'entrée.* Ils donnent alors comme règle générale : *l'algorithme de Godin doit être utilisé pour des pe-*

3. *cf.* [Lawler 80, Johnson 88] pour plus de précisions sur la notion de complexité.

Algorithme 3.4 Successeurs_Immédiats((A, B),$(O, \mathcal{I}, (\alpha, \beta))$)

Entrées :
(A, B) : un concept formel du contexte $(O, \mathcal{I}, (\alpha, \beta))$;
$(O, \mathcal{I}, (\alpha, \beta))$: un contexte ;
Sorties :
$Succ_{(A,B)}$: l'ensemble des concepts successeurs immédiats de (A, B) ;

DÉBUT

Calculer $\mathcal{F}_A$: $\mathcal{F}_A = \{\beta(b) \cap A | b \in \mathcal{I} \setminus B\}$;
Calculer F : les sous-ensembles maximalement inclus de $\mathcal{F}_A$;
pour chaque $A_i \in F$ **faire**
  Ajouter à $Succ_{(A,B)}$ le concept $(A_i, \alpha(A_i))$;
fin pour
renvoyer $Succ_{(A,B)}$;
FIN

tits contextes peu denses ; pour des contextes denses (*i.e.* ratio fort entre le nombre de croix du contexte et son nombre de case)*, les algorithmes tels que CloseByOne* [Kuznetsov 93, Kuznetsov 02]*, celui de Ganter, ou celui de Norris, devraient être utilisés. L'algorithme de Bordat fonctionne bien sur les contextes de densité moyenne, en particulier, lorsque le diagramme de Hasse doit être construit.*

Pour notre part, nous souhaitons utiliser le treillis pour de la classification supervisée. Nous souhaitons en particulier pouvoir naviguer dans le diagramme de Hasse d'un treillis de Galois et conserver la possibilité d'une génération à la demande telle que définie dans la méthode Navigala (voir section suivante et Navigala [Guillas 06b, Guillas 07, Bertet 07, Visani 11]), c'est pourquoi **nous utilisons dans toutes nos expérimentations l'algorithme de Bordat et donc le diagramme de Hasse**.

3.3.3 Visualisation

La structure de treillis de Galois est donc définie sur deux ensembles. Comme mentionné précédemment, les ordres et les treillis sont généralement représentés sous la forme d'un diagramme de Hasse. Il existe différents logiciels permettant de générer et de dessiner le diagramme de Hasse d'un treillis de Galois [Priss 08]. Les Figures 3.3.1, 3.3.2 et 3.3.5 présentent le diagramme de Hasse du treillis de Galois associé au contexte formel de la Table 3.3 et dessinés par différents logiciels que nous présentons dans les paragraphes suivants.

Génération et visualisation

Conexp

La Figure 3.3.1 présente le diagramme de Hasse du treillis de Galois associé au contexte formel de la Table 3.3, obtenu *via* le logiciel Conexp (Concept Explorer [4]) qui implémente l'algorithme *NextClosure*. Conexp a été développé par Yevtushenko [Yevtushenko 00, Yevtushenko 04]. Dans ce logiciel, la première étape consiste à saisir le contexte formel, puis différentes options peuvent être choisies, dont le dessin du diagramme de Hasse. Dans ce diagramme, les concepts (ou nœuds) du treillis sont les ronds. Les sup-irréductibles sont représentés avec la moitié haute du rond, en bleue. Les inf-irréductibles sont représentés avec la moitié base du rond, en noire. Chaque attribut est associé au plus petit sup-irréductible le contenant, tandis que chaque objet est associé au plus grand inf-irréductible le contenant (selon la correspondance présentée dans la section 3.2.3).

Cette représentation est appelée *représentation réduite du treillis* [Ganter 99, Yevtushenko 04]. En effet, tous les nœuds ne sont pas étiquetés ; cependant, chaque nœud contient bien un ensemble d'objets et un ensemble d'attributs. L'ensemble des attributs d'un concept est l'ensemble formé par les attributs contenus dans les sup-irréductibles qui le précédent. L'ensemble des objets d'un concept est l'ensemble formé par les objets contenus dans les inf-irréductibles qui lui succèdent. Les relations entre concepts ne sont pas orientées, mais dans le cas de la relation d'ordre que nous avons choisi à la définition 16, l'ordre s'oriente vers le bas, *i.e.* le plus petit élément (*bottom*) est le concept le plus haut de la Figure et le plus grand élément (*top*) est celui le plus bas.

Galicia

La Figure 3.3.2 présente le diagramme de Hasse du treillis de Galois associé au contexte formel de la Table 3.3, obtenu *via* le logiciel Galicia (Galois lattice interactive constructor [5]). Galicia a été développé entre différents laboratoires de Montréal, Nancy et Montpellier [Valtchev 03]. Dans ce logiciel, plusieurs algorithmes sont proposés, en particulier NextClosure et l'algorithme de Bordat. La première étape consiste à saisir le contexte formel, puis différents algorithmes peuvent être choisis dont ceux pour la génération du treillis. Une fois la génération du treillis terminée, il est possible de visualiser le dessin du diagramme de Hasse. Les concepts (ou nœuds) du treillis sont les ronds. Chaque concept est labellisé avec un numéro. Les ensembles d'attributs (I - intension) et d'objets (E - extension) sont visibles par un clic droit sur le concept souhaité dans l'interface du logiciel (*cf.* exemple sur la Figure 3.3.2 pour le concept n°13). Comme pour Conexp, les relations entre concepts ne sont pas orientées, mais dans le cas de la relation d'ordre que nous avons choisie à la définition 16, l'ordre s'oriente vers le bas, *i.e.* le plus petit

4. `http://conexp.sourceforge.net/`
5. `http://www.iro.umontreal.ca/~galicia/goals.html`

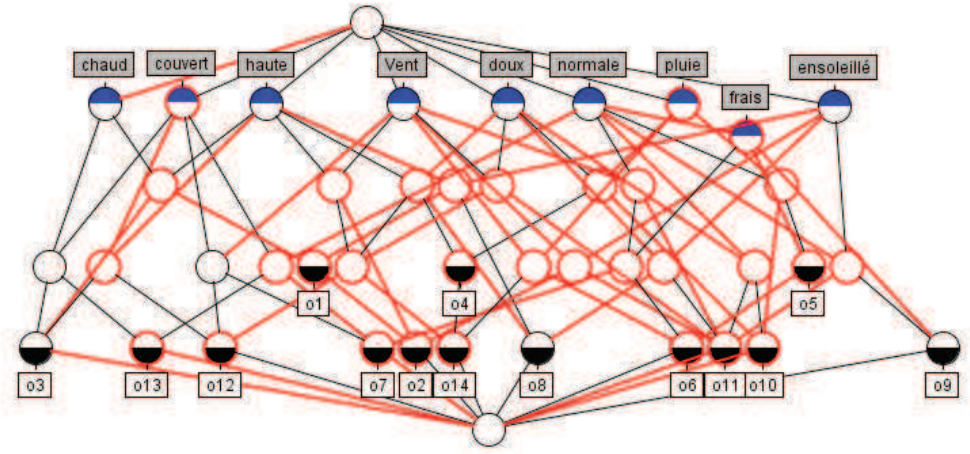


FIGURE 3.3.1: Diagramme de Hasse du treillis de Galois associé au contexte formel de la Table 3.3 - Conexp

élément (*bottom*) est le concept le plus haut de la Figure (le n° 1 en rouge) et le plus grand élément (*top*) est celui le plus bas (le n°44 dans cet exemple).

Bibliothèque lattice

La bibliothèque *lattice*[6] a été développée par K. Bertet [Bertet 11], en langage JAVA. **Les travaux présentés dans ce manuscrit ont été implémentés comme surcouche de cette bibliothèque (*cf.* Annexe E pour plus de précision sur le logiciel développé *Navigala2012*)**.

De manière générale, cette bibliothèque propose un ensemble de classes et de méthodes permettant de définir et de manipuler les systèmes de fermeture (*cf.* figures 3.3.3 et 3.3.4 extraites de [Bertet 11]). Plus particulièrement, il est possible d'instancier un contexte formel et de lui appliquer certaines opérations telle que la réduction sur les objets et sur les attributs (*cf.* Figure 3.3.3). Nous présentons ici uniquement les classes que nous avons utilisées pour le développement de nos travaux (*cf.* chapitres 5 et 6).

La classe principale est la classe abstraite *ClosureSystem*, elle contient les méthodes spécifiques à un système de fermeture (en particulier, l'opération de fermeture), et les méthodes génériques de manipulation d'un système de fermeture (en particulier, **génération du treillis des fermés avec l'algorithme Next-Closure ou l'algorithme de Bordat**). Cette classe définit les méthodes communes à un système de fermeture issu

6. La bibliothèque *lattice* est disponible sous licence LGPL.

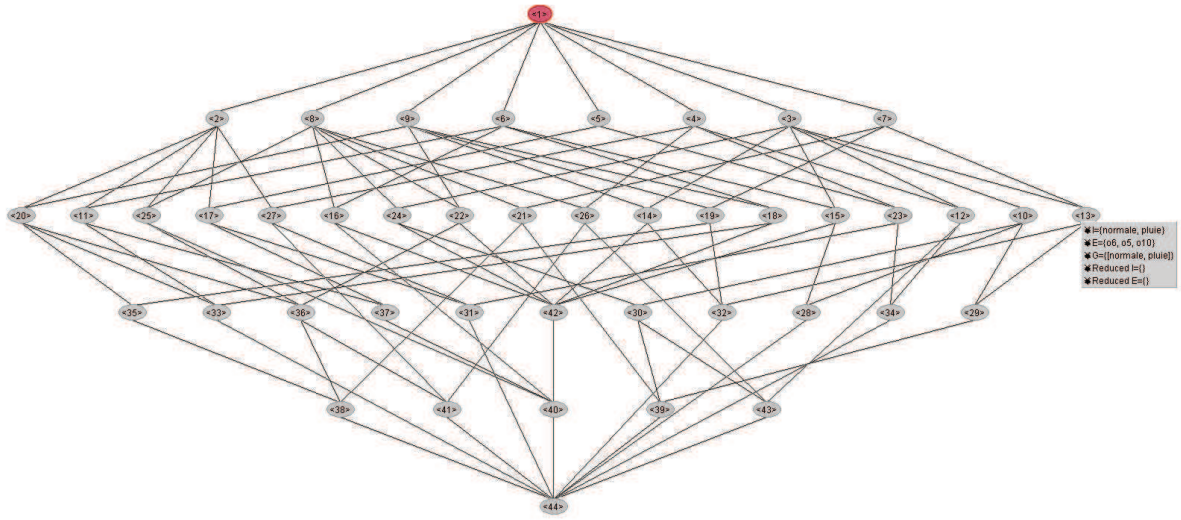


FIGURE 3.3.2: Diagramme de Hasse du treillis de Galois associé au contexte formel de la Table 3.3 - Galicia

d'un ensemble de règles (instanciable par la classe *IS)* ou d'une table binaire (instanciable par la classe *Context*).

La classe que nous utilisons principalement est la classe *Context.* Cette dernière permet d'instancier et de manipuler un contexte, l'instanciation pouvant se faire à partir d'un fichier texte, suivant le format suivant :

Exemple de Contexte :
Observations : o1 o2 o3
Attributes : I1 I2 I3 I4 I5
o1 : I1 I3
o2 : I1 I2
o3 : I2 I4 I5

Ensuite, dans le cadre de ces travaux, nous nous intéressons plus particulièrement à la classe *ConceptLattice* qui représente un treillis de Galois. Les instances de cette classe sont générées à partir d'un système de fermeture, et plus particulièrement d'un contexte. Les nœuds d'un treillis des Galois sont des instances de la classe *Concept.* Cette classe permet de représenter les deux ensembles définissant un concept formel (extension - ensemble d'objets et intension-ensemble d'attributs).

Tout graphe peut être sauvegardé dans un fichier au format *dot*, et visualisé à l'aide des outils proposés par le logiciel Graphviz, outil de visualisation de graphes.

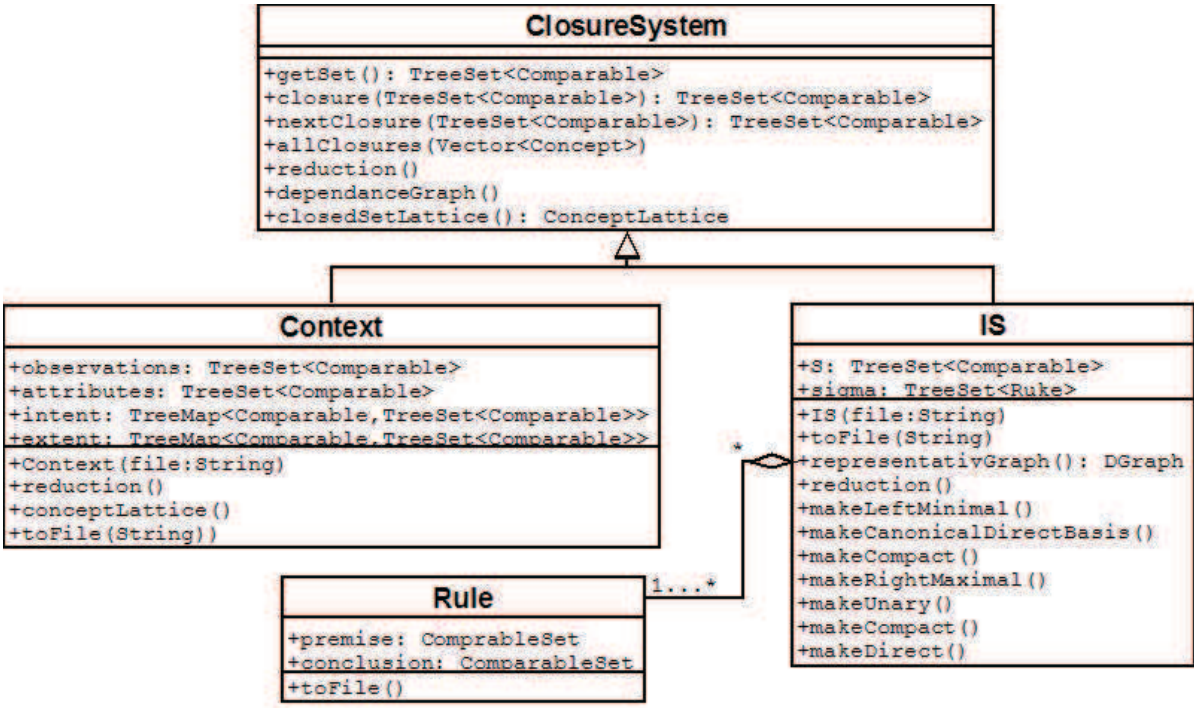


FIGURE 3.3.3: Bibliothèque *lattice* - Système de fermeture

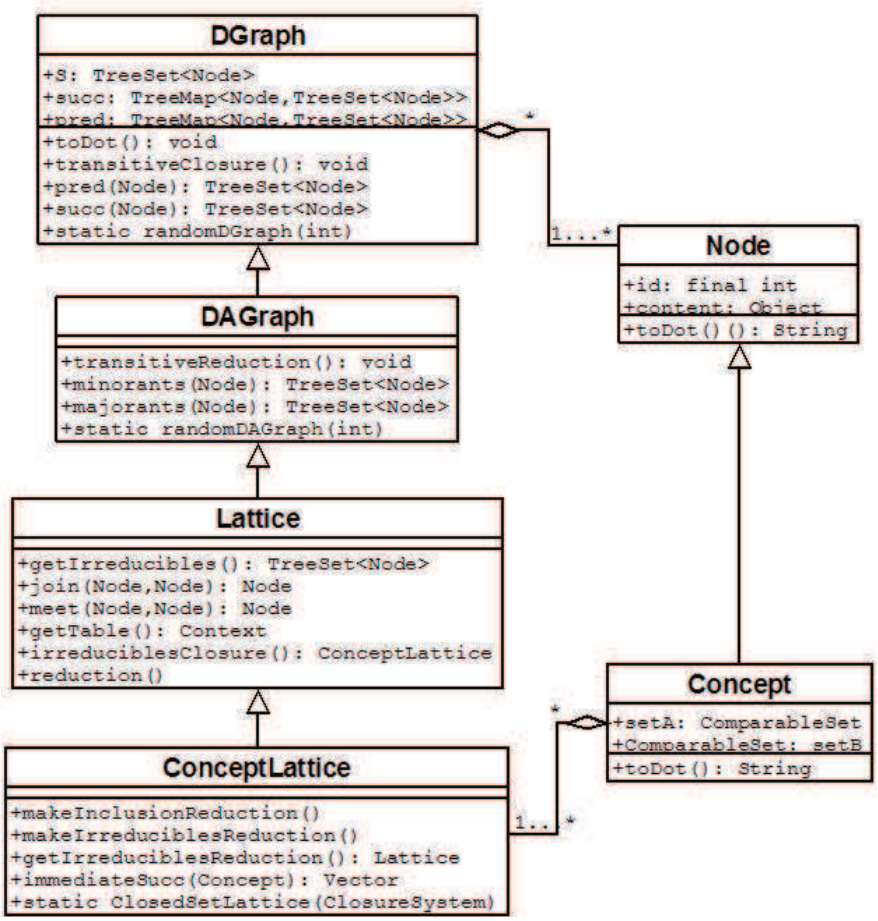


FIGURE 3.3.4: Bibliothèque *lattice* - Treillis

Visualisation

Graphviz & dotty

La Figure 3.3.5 présente le diagramme de Hasse du treillis de Galois associé au contexte formel de la Table 3.3, obtenu *via* le logiciel Graphviz (Graph vizualisation software [7]). Graphviz a été développé en particulier par le laboratoire AT&T Bell [Gansner 00]. Ce logiciel permet de dessiner tout type de graphe (arbres de classification, treillis,...). Il propose différents algorithmes de visualisation de graphe. En particulier, l'algorithme de visualisation de graphes sans cycle permet d'obtenir des dessins d'aussi bonne qualité que ceux obtenus avec Conexp et Galicia [Priss 08].

Graphviz utilise un fichier de description du graphe pour le dessiner au format *dot* (*cf.* exemple de fichier décrivant un treillis de Galois, Annexe B).

La bibliothèque *lattice* permet de générer ce type de fichier, elle y décrit les treillis en étiquetant les concepts avec les deux ensembles qui les définissent, *i.e.* l'ensemble d'objets et l'ensemble d'attributs. De plus, les concepts du treillis sont les ellipses et les relations entre concepts sont décrites comme orientées. Dans la Figure 3.3.5, l'ordre s'oriente vers le bas, *i.e.* le plus petit élément (*bottom*) est le concept le plus haut de la Figure et le plus grand élément (*top*) est celui le plus bas.

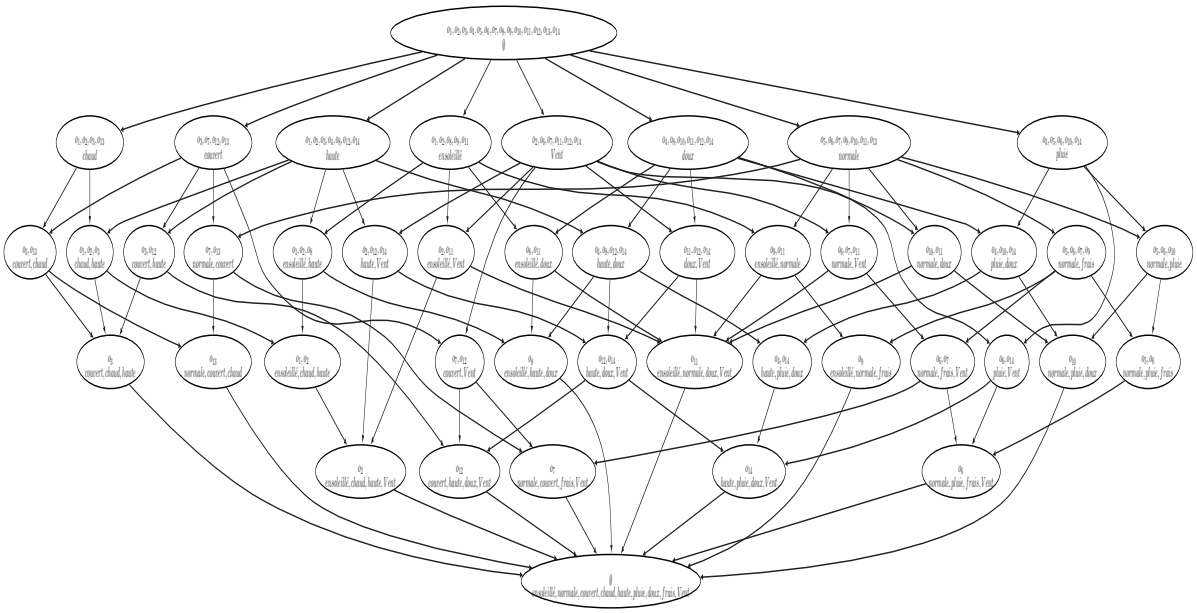


FIGURE 3.3.5: Diagramme de Hasse du treillis de Galois associé au contexte formel de la Table 3.3 - Graphviz

Les travaux présentés dans ce manuscrit s'appuyant sur la bibliothèque *lattice*, et celle-ci utilisant Graphviz, nous utilisons le logiciel Graphviz pour

7. `http://www.graphviz.org/Home.php`

le dessin et la visualisation des diagrammes de Hasse. De plus, ce logiciel permet un affichage complet du contenu des concepts, ce qui facilite la comparaison avec les arbres de classification.

3.4 Méthodes de classification

Les treillis de Galois, actuellement en pleine émergence dans le domaine de l'analyse de données, ont été *popularisés* en particuliers par les travaux de Ganter et Wille sur l'analyse formelle des concepts (AFC) [Ganter 99]. De nombreux auteurs s'intéressent à cette structure qui démontre des propriétés très intéressantes pour la classification supervisée ou non supervisée [Oosthuizen 88, Liquière 90, Sahami 95, Ganter 97, Njiwoua 99, Mephu-Nguifo 00, Mephu-Nguifo 05, Guillas 07, Poelmans 10, Visani 11].

L'ensemble des méthodes proposées se décompose en deux grands types de méthodes : **les méthodes orientées sélection et les méthodes orientées navigation**. De nombreux articles rappellent et comparent les différentes méthodes [Godin 86b, Godin 93, Guillas 07, Visani 11, Mephu-Nguifo 05, Fu 04a]. Vu le nombre croissant de ces méthodes, il est difficile d'être exhaustif à leur sujet. Dans un objectif de comparaison avec les arbres de classification, nous présentons succinctement ci-dessous une méthode orientée sélection permettant la construction de règles de classification. Puis nous présentons plus en détail, une méthode orientée navigation permettant la classification de nouveaux objets par parcours de la structure du diagramme de Hasse d'un treillis. C'est cette dernière méthode que nous utilisons dans les travaux présentés dans ce manuscrit.

3.4.1 Sélection de concepts

Les méthodes orientées sélection sont nombreuses, parmi les plus citées se trouvent la méthode GRAND (GRAph iNDuction) issue des travaux de Oosthuizen [Oosthuizen 88], la méthode LEGAL développée par Liquière et Mephu-Nguifo [Liquière 90] (améliorée *via* la méthode GLUE [Njiwoua 97, Mephu-Nguifo 00]), la méthode GALOIS de Carpineto et Romano [Carpineto 93], la méthode RULEARNER définie par Sahami [Sahami 95] ou encore la méthode CIBLe de Njiwoua et Mephu-Nguifo [Njiwoua 99].

Tandis que **les méthodes LEGAL, GALOIS et GLUE sont basées sur** la sélection de concepts pertinents suivie d'une phase de classification selon **le concept pertinent le plus similaire à l'objet à classer**, **les méthodes GRAND et RULEARNER procèdent à une extraction de règles de classification**, la classification d'un nouvel objet s'effectuant selon un vote majoritaire ou selon la première règle vérifiée. L'extraction des règles de classification étant un procédé mis en œuvre dans les arbres de classification, nous nous intéressons plus particulièrement à l'une des méthodes procédant à ce type d'extraction, *i.e.* la méthode GRAND.

GRAND

Comme la plupart des méthodes de classification, la méthode GRAND repose sur une phase d'apprentissage durant laquelle des règles de classification sont extraites du treillis, puis sur une phase de classification basées sur ces règles extraites.

Apprentissage

Pour la construction du treillis, Oosthuizen propose un algorithme de construction incrémentale [Oosthuizen 88, Mephu-Nguifo 05]. Une fois le treillis construit, vient la phase d'extraction des règles de classification, celle-ci est réalisée selon le principe suivant : pour une classe donnée, on recherche le concept le plus général permettant d'inférer la classe, *i.e.* la borne inférieure contenant des objets de cette classe. Les attributs du concept deviennent la prémisse de la règle et la classe devient sa conclusion.

Classification

Pour la classification d'un objet, toutes les règles de classification vérifiées par l'objet sont collectées. Puis un vote majoritaire est appliqué, *i.e.* l'objet sera alors classé dans la classe majoritaire sur l'ensemble des règles collectées.

3.4.2 Navigation

L'un des premiers articles utilisant la navigation dans les treillis de Galois est celui de Godin *et al. [Godin 86b],* qui propose une **navigation manuelle** selon la navigation dans une arborescence de fichiers. Un certain nombre de propositions ont été faites pour naviguer dans les treillis, comme par exemple Camelis [Ferré 07, Ferré 09] qui permet d'organiser des fichiers selon leur description et de les retrouver par construction incrémentale d'une requête, cette construction étant guidée par une navigation dans un treillis de Galois.

Dans [Guillas 06a], **une navigation automatique** est définie avec pour objectif la classification d'images. De cette étude, est née **la méthode Navigala** (NAVIgation into GAlois LAttice) [Guillas 07, Visani 11]. Initialement développée pour des images de symboles, elle a démontré son intérêt pour la classification de manière générale. **C'est cette dernière méthode qui nous intéresse**, du fait de ses liens avec les arbres de classification.

NAVIGALA

Pour mettre en œuvre la méthode Navigala, lorsque les données à traiter sont quantitatives, une étape de prétraitement est nécessaire pour transformer ces données en données qualitatives par **discrétisation des valeurs. Nous abordons ce point dans le chapitre 4**. Supposons dès lors que les données à traiter sont de type qualitatif.

Tout d'abord, une transformation en données binaires est réalisée (un attribut binaire est créé par modalité de chaque attribut qualitatif, *cf.* binarisation p 1.1.1). La phase d'apprentissage consiste, dans un premier temps, comme pour l'arbre de classification, à **construire le diagramme de Hasse du treillis de Galois** puis, dans un deuxième temps, à **étiqueter les concepts terminaux,** équivalents des feuilles dans un arbre. Un fois l'apprentissage achevé, il est possible d'utiliser la structure par navigation pour classer de nouveaux objets.

Apprentissage

Construction du diagramme de Hasse

La méthode Navigala utilise l'algorithme de Bordat [Bertet 07] pour la construction du diagramme de Hasse d'un treillis à partir d'une table de données binaires. Cet algorithme est présenté dans la section 3.3.2.

Étiquetage des concepts terminaux

Une fois le diagramme construit, les concepts contenant des objets appartenant à une même classe sont étiquetés avec cette dernière. De plus, pour garantir la possibilité d'une navigation, les coatomes du treillis sont aussi étiquetés avec leur classe majoritaire. Les concepts étiquetés sont appelés **concepts terminaux**.

Notons qu'en raison de la relation d'ordre existante entre concepts d'un treillis de Galois, lorsqu'un concept est un concept terminal, tous ses successeurs sont aussi des concepts terminaux, car ils contiennent un sous-ensemble d'objets d'un concept terminal.

Notons aussi qu'**une classe peut être représentée par plusieurs concepts terminaux.**

En outre, comme indiqué précédemment, **un résultat de la théorie des treillis [Birkhoff 67] établit une correspondance entre inf-irréductibles et objets** : chaque inf-irréductible correspond au plus grand concept $(\beta(\alpha(o)), \alpha(o))$ contenant un sous-ensemble d'objets ayant la même description $\{\alpha(o)\}$. En analyse formelle des concepts, ce concept est appelé concept-objet. Lorsque les données sont séparables, les objets de classes différentes sont généralement représentés par des descriptions différentes. Ainsi tout inf-irréductible contient des objets de même classe. Cela induit la propriété suivante :

Propriétés 1. Lorsque les données sont séparables, les inf-irréductibles d'un treillis sont des concepts terminaux.

Classification

Une fois le diagramme de Hasse étiqueté, la phase de classification de tout nouvel objet peut être réalisée. Cette dernière est basée sur la même idée que la navigation dans un arbre de classification (*cf.* section 2.6.1), *i.e.* le nouvel objet est inséré au niveau du

plus petit concept du diagramme $(O, \alpha(O))$ où toutes les classes des objets sont candidates et aucun attribut n'est validé, et descend dans la structure arborescente jusqu'à atteindre un **concept terminal** et à être étiqueté avec la classe de ce dernier. L'objet progresse étape par étape au sein du diagramme par validation de nouveaux attributs (et par conséquent par réduction de l'ensemble d'objets). Il s'agit d'un **processus itératif** avec une prise de décision locale, *i.e.* d'un concept courant vers l'un de ses concepts successeurs. Chaque étape de la navigation consiste à choisir un successeur immédiat. Pour faire le choix parmi tous les successeurs validés, différents critères de choix peuvent être utilisés. La méthode Navigala ayant été développée dans le cadre du traitement de données quantitatives, des critères de distance ont été définis [Guillas 07]. Mais, pour des données qualitatives binaires, ce critère de choix peut être par exemple « choisir le concept successeur ayant le moins (resp. le plus) d'objets dans son extension », *etc.*

La classification par navigation est réalisée à partir du diagramme de Hasse d'un treillis, *i.e.* à partir du treillis réduit. Cette dernière est stoppée dés qu'un concept terminal est atteint, de ce fait certains concepts ne peuvent être atteints par navigation, il est donc inutile de les conserver. C'est le cas par exemple du concept maximal du treillis, ce dernier ayant pour prédécesseurs uniquement des concepts terminaux (*cf.* propriété 1). Nous pouvons donc les supprimer des représentations. L'algorithme 3.5 présente cette suppression.

Exemple. La Figure 3.4.1 présente le diagramme de Hasse étiqueté du treillis de Galois du contexte de la Table 3.6. Les concepts terminaux sont les nœuds en forme octogonale, et sont étiquetés comme pour les feuilles d'un arbre de classification avec la classe majoritairement présente. Dans cet exemple, les concepts terminaux sont purs, de plus on remarque qu'une classe peut être représentée par plusieurs concepts terminaux, c'est le cas de la classe 4.

La classification de l'objet $o_{11} = \{I_1, I_4, I_6\}$ suit les flèches en tirets gris sur la Figure 3.4.2. Par navigation, le nouvel objet traverse les concepts grisés jusqu'à atteindre un concept terminal étiqueté avec la classe 2. Ce nouvel objet est donc étiqueté avec cette classe 2. Dans cet exemple, pour la première étape, deux concepts successeurs du bottom étaient valides, le critère de choix appliqué est « choisir le successeur contenant le moins d'objets ». Dans cet exemple, les flèches en pointillés gris clairs matérialisent les autres chemins valides et qui permettent d'aboutir au même concept terminal.

La méthode Navigala propose aussi la génération à la demande du diagramme de Hasse, le nombre de concepts pouvant être exponentiel en la taille des données dans le pire cas. La génération à la demande consiste à construire le diagramme de Hasse au fur et à mesure de la phase de classification, en ne construisant que les concepts nécessaires à la classification d'un nouvel objet, en utilisant l'algorithme de calcul des successeurs de l'algorithme de Bordat. Pour plus de précisions sur la génération à la demande, nous renvoyons le lecteur aux références suivantes [Guillas 07, Visani 11].

Algorithme 3.5 *Simplification_après_Concepts_Terminaux($\mathcal{L}$)*

Entrées :
- $\mathcal{L} = (\mathcal{C}, \prec)$ le diagramme de Hasse à simplifier ;

Sorties :
- $\mathcal{L}$ le diagramme de Hasse simplifié ;

DÉBUT
Supprimer $\top$ de $\mathcal{L}$;
$CT \leftarrow$ l'ensemble de concepts terminaux de $\mathcal{L}$;
pour chaque $C \in CT$ **faire**
  pour chaque successeur *succ_courant* de C **faire**
    Supprimer l'arc entre C et *succ_courant* ;
  fin pour
fin pour
pour chaque $C \in CT$ **faire**
  pred $\leftarrow$ ensemble des predecesseurs de C dans $\mathcal{L}$;
  si *pred* est vide **alors**
    Supprimer C de $\mathcal{L}$;
  fin si
fin pour
renvoyer $\mathcal{L}$;
FIN

O	$\mathcal{I}$						K
	I_1	I_2	I_3	I_4	I_5	I_6	
o_1	×		×			×	1
o_2	×		×			×	
o_3	×			×		×	2
o_4	×			×		×	
o_5	×			×		×	
o_6		×		×		×	3
o_7		×		×		×	
o_8		×		×		×	
o_9		×	×		×		4
o_{10}		×		×	×		

TABLE 3.6: Contexte exemple méthode Navigala

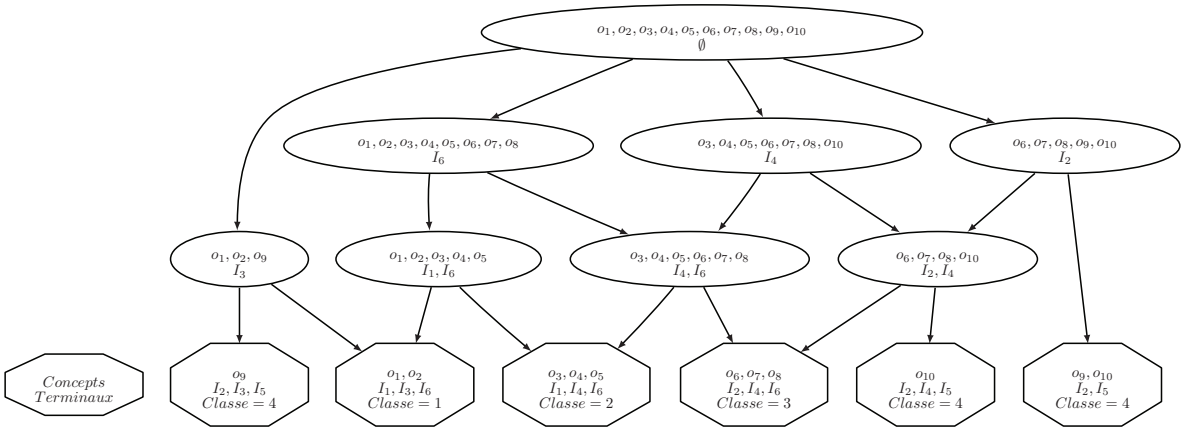


FIGURE 3.4.1: Diagramme de Hasse étiqueté du treillis de Galois de la Table 3.6

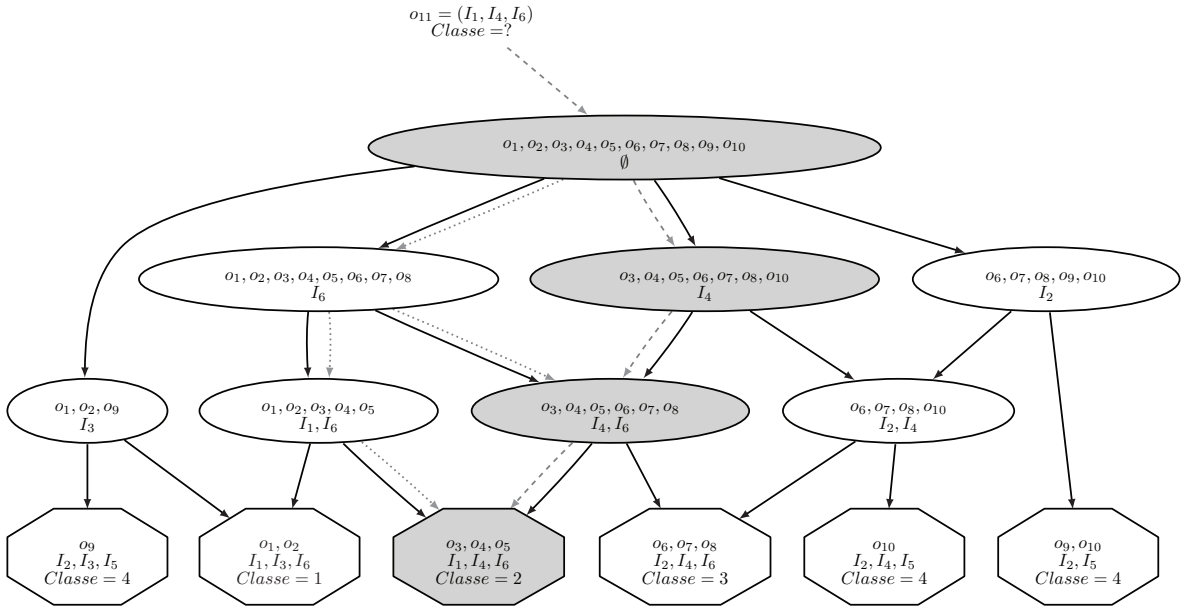


FIGURE 3.4.2: Classification de l'objet $o_{11} = \{I_1, I_4, I_6\}$

La navigation à travers le treillis de Galois est proche de la navigation dans un arbre de classification. Nous reviendrons plus formellement sur cette méthode dans le chapitre suivant, en particulier pour comparer la navigation dans un arbre et celle dans un treillis.

3.5 Discussion

3.5.1 Avantages

Parmi les avantages des treillis de Galois, les plus cités sont :

1. La **lisibilité** du treillis de Galois qui permet, comme pour l'arbre de classification, d'appréhender directement la connaissance qu'il produit.
2. La réduction du contexte permet de mettre en avant les liens entre attributs, *i.e.* les attributs équivalents, difficiles à voir sur la table complète.
3. Comme pour l'arbre, il est possible d'utiliser directement le treillis de Galois pour de nouveaux objets. Aucune transformation des attributs des nouveaux objets n'est nécessaire pour pouvoir les classer.
4. Tout comme l'arbre de classification, le treillis de Galois peut être traduit en une base de règles. Les règles de classification de la forme *SI ... ALORS ...* sont généralement très faciles à lire et à réutiliser dans d'autres systèmes, au contraire du résultat d'une méthode statistique par exemple.
5. Il est **robuste aux données bruitées**, en particulier grâce aux multiples chemins permettant de rejoindre un même concept terminal [Guillas 07].
6. La construction du treillis de Galois est **non paramétrique** : non seulement il ne nécessite aucune hypothèse *a priori* sur la distribution des données, mais en plus, sa construction n'est pas dépendante de la définition de seuils ou autre.
7. A l'image de l'arbre, il permet une séparation linéaire par morceaux des classes donc une résolution des problèmes non linéairement séparables. En effet, les concepts du treillis correspondent à des regroupement d'objets et à la conjonction de plusieurs attributs. Chaque concept peut donc être représentée comme un pavé dans l'espace des données.
8. Il offre des **performances comparables aux autres méthodes supervisées**. Plus robuste aux données bruitées que l'arbre de classification, il a de meilleures performances que l'arbre lorsque les deux structures sont définies à partir d'une même table de données binaire [Gély 04, Mephu-Nguifo 05, Guillas 07].

3.5.2 Inconvénients

Parmi les inconvénients des treillis de Galois, les plus cités sont :

1. Définis pour des tables de données binaires, **un prétraitement de discrétisation doit être mis en œuvre pour transformer les attributs quantitatifs** (*cf.* chapitre 5). Ce traitement engendre une perte d'information, *i.e.* une perte des valeurs exactes vérifiées par chaque objet.
2. La taille du treillis est exponentielle en la taille des données dans le pire cas. Même si dans la pratique celle-ci reste raisonnable, il est beaucoup **plus complexe que l'arbre de classification**. Rappelons cependant que, pour pallier à ce problème, un processus de génération à la demande a été développé.

3.6 Conclusion

Dans ce chapitre, nous avons présenté l'origine des treillis de Galois avec les ordres, la structure de treillis avant de nous intéresser plus précisément à la correspondance de Galois et au treillis de Galois ou des concepts. Nous avons également présenté les algorithmes de construction de treillis les plus connus, avant de présenter l'utilisation de cette structure dans le cadre de la classification supervisée. Enfin, nous avons rappelé les points forts et les points faibles accordés aux treillis de Galois.

Les treillis de Galois sont de plus en plus utilisés dans le cadre de la recherche d'information et de l'extraction de connaissances dans les bases de données. Les solutions proposées dans le cadre de la classification supervisée permettent une interprétation facile des résultats obtenus. Les treillis de Galois sont plus stables que les arbres de classification, et ont démontré des performances similaires à d'autres méthodes plus connues [Gély 04, Mephu-Nguifo 05, Guillas 07]. Cependant, ils sont aussi connus pour leur taille qui, dans le pire cas, peut être exponentielle en la taille de données, ce cas se réalisant cependant peu dans la pratique. A l'inverse de l'arbre, les treillis sont définis pour des données qualitatives (binaires ou binarisées), ils utilisent donc une transformation des données quantitatives en amont de leur construction.

A l'inverse des arbres de classification pour lesquels certains chercheurs considèrent le *problème comme résolu*, les treillis de Galois sont en pleine émergence dans le cadre de l'analyse de données. De nombreuses méthodes sont actuellement développées à des fins diverses [Carpineto 04, Kuznetsov 04, Choi 06, Roth 08, Kaytoue 11, Falk 11, Nguyen 12].

La définition de treillis de Galois à partir de données quantitatives est un domaine de recherche récent [Pfaltz 07, Kaytoue 11]. Les liens structurels existants entre les arbres de classification et les treillis de Galois, ainsi que le traitement des données quantitatives par les arbres de classification nous semblent deux notions importantes à étudier pour améliorer la définition des treillis dans un cadre de classification d'images. De plus, l'élagage existant pour l'arbre de classification nous semble être intéressant pour pallier le problème de taille des treillis de Galois. Aussi, dans la suite de ce manuscrit, nous

présentons différentes études permettant de définir une nouvelle méthode hybride de classification entre arbre de classification et treillis de Galois.

Plus précisément, nous nous intéresserons aux liens entre arbre de classification et treillis de Galois dans le chapitre 4. Puis nous nous intéressons à leur traitement des données quantitatives au chapitre 5. Enfin nous nous intéressons aux différentes simplifications de ces structures au chapitre 6.

Points clés

Positionnement

- ❏ Nous avons présenté les définitions liées à la théorie des treillis avant de nous intéresser plus précisément aux treillis de Galois.
- ❏ Les algorithmes les plus connus ont été présentés et différenciés.
- ❏ L'utilisation des treillis de Galois en classification supervisée a été présentée.
- ❏ Les points forts et les points faibles des treillis de Galois ont été rappelés.

Contributions

- ❏ Nous avons précisé les notations que nous utilisons et qui nous permettent de faire le comparatif avec les arbres de classification.

Vers un modèle hybride

Chapitre 4

Les liens entre arbres et treillis de Galois

4.1 Introduction

Après l'état de l'art réalisé dans les chapitres 2 et 3, ce chapitre a pour objectif de **présenter les liens forts qui unissent les arbres et les treillis de Galois**. De manière générale, des liens entre arbres de classification et treillis de Galois ont déjà fait l'objet d'un certain nombre d'études [Polaillon 98, Guillas 05, Nijssen 07, Belohlávek 07, Guillas 08, Bertet 09].

Pour notre part, nous nous intéressons aux **liens structurels** mais aussi aux **liens en usage pour les tâches de classification**. Dans les articles [Guillas 08, Bertet 09], nous avons défini et démontré les liens forts entre les arbres de classification et une classe particulière de treillis, à savoir **les treillis dichotomiques** dont nous avons proposé la définition.

Dans ce chapitre, **nous donnons cette définition formelle des treillis dichotomiques**, nous démontrons ses propriétés ainsi que ses liens avec les arbres de classification.

Ce chapitre constitue **une étude préliminaire à l'introduction d'un modèle hybride** qui préserve les points forts de chacun de ces deux modèles, tout en limitant les points faibles (*cf.* chapitres 5 et 6).

Comme nous l'avons introduit dans l'état de l'art, les arbres de classification et les treillis de Galois peuvent être utilisés de différentes manières pour les tâches de classification supervisée. En particulier, nous avons vu qu'il était possible d'extraire des règles de classification des deux structures, mais aussi de naviguer dans celles-ci. Dans la section 4.2, nous formalisons la navigation dans ces deux structures et comparons les deux méthodes. Nous rappelons que **la navigation dans le treillis de Galois généralise**

la navigation dans un arbre de classification [**Guillas 07**]. Pour ce comparatif, **nous nous concentrons sur les arbres construits avec des critères de division dit monovarié** ou statique, ces derniers étant généralement utilisés pour les données qualitatives. Dans le chapitre suivant, nous présenterons rapidement les critères multivariés ou dynamiques, pour autant nous ne les utiliserons pas, ces derniers étant peu utilisés dans les algorithmes courants de construction d'arbres.

A partir d'**une table de données binaires**, il est possible de définir **plusieurs arbres de classification** (selon l'algorithme de construction et les critères choisis). En revanche, il n'existe qu'**un seul et unique treillis de Galois**. Dans la section 4.3, nous présentons le **lien d'inclusion entre les différents arbres de classifications et le treillis de Galois**. Il s'agit d'une généralisation du lien d'inclusion défini uniquement pour les treillis dichotomiques dans [Guillas 08, Bertet 09]. Nous donnons la preuve de ce lien.

Ensuite dans la section 4.4, nous présentons **les treillis dichotomiques, nous les définissons de manière formelle et nous les caractérisons par les propriétés qu'ils vérifient** (la coatomicité et la $\vee$-complémentarité). **Puis nous rappelons le lien de fusion qui les unit aux arbres de classification.**

Notons que dans le cadre de l'analyse d'images, les tables de données sont généralement quantitatives. Les deux modèles étudiés utilisent une discrétisation (*cf.* chapitre suivant) pour transformer ces données en données qualitatives. Les tables de données qualitatives ainsi construites contiennent des modalités mutuellement exclusives. C'est pourquoi, **nous excluons de cette étude, les tables de données qualitatives dont les modalités sont non exclusives.**

Pour illustrer nos propos au cours de ce chapitre, nous utiliserons en particulier la table de données qualitatives $\Omega = (O, \mathcal{I}, K)$, présentée dans la Table 4.1. La table binaire correspondante est présentée dans la Table 4.2.

Pour simplifier les notations et pour faciliter les comparaisons, chaque attribut qualitatif est noté avec un indice unique, *i.e.* $I_{.z} = I_z$ **; et nous associons à chaque attribut** I_z **un ensemble de modalités fixées.** Plus précisément, l'attribut qualitatif I_1 a deux modalités notées m_1 et m_2, l'attribut qualitatif I_2 a aussi deux modalités notée m_3 et m_4 et l'attribut qualitatif I_3 a trois modalités notées m_5, m_6 et m_7.

4.2 Lien en classification

Nous avons présenté dans les chapitres 2 et 3 des méthodes de classification basées soit sur l'arbre de classification, soit sur le treillis de Galois. En particulier, nous avons présenté pour chaque modèle **l'étape d'apprentissage** qui consiste en la construction

O	$\mathcal{I}$			K
	I_1	I_2	I_3	
o_1	m_1	m_3	m_7	1
o_2	m_1	m_3	m_7	
o_3	m_1	m_4	m_6	2
o_4	m_1	m_4	m_6	
o_5	m_1	m_4	m_6	
o_6	m_2	m_4	m_6	3
o_7	m_2	m_4	m_6	
o_8	m_2	m_4	m_6	
o_9	m_2	m_3	m_5	4
o_{10}	m_2	m_4	m_5	

TABLE 4.1: Table de données qualitatives

de la structure arborescente à partir d'une table de données (que nous supposons qualitatives, nous traitons les données quantitatives dans le chapitre 5) puis en l'étiquetage des feuilles ou concepts terminaux. Ensuite, **l'étape de classification** de nouveaux objets se fait soit par extraction d'un système de règles de classification, soit par navigation dans la structure.

4.2.1 Étapes d'apprentissage et de classification

Pour les deux modèles, la phase d'apprentissage est divisée en deux étapes, la construction de la structure à partir de la table de données (*cf.* sections 2.5 et 3.3 pour plus de précisions sur les algorithmes), **puis l'étiquetage des feuilles** (pour les arbres) **ou des concepts terminaux** (pour les treillis), cette dernière étape consiste à étiqueter par la classe la plus fréquente, les nœuds ou concepts qui satisfont une mesure de pureté.

Exemple. La Figure 4.2.1 présente un arbre de classification construit à partir de la Table 4.1 en utilisant l'algorithme C4.5 et la Figure 4.2.2 présente l'arbre de classification construit à partir de la même table en utilisant l'algorithme CART sans élagage. Les feuilles (de forme rectangulaire) sont étiquetées avec la classe majoritaire.

La Figure 4.2.3 présente le diagramme de Hasse du treillis de Galois de la Table binaire 4.2 (équivalente à la Table 4.1). Les concepts terminaux sont les concepts purs et les coatomes du treillis, de forme octogonale sur la figure, ils sont étiquetés avec la classe majoritaire.

Une fois les arborescences calculées et étiquetées, elles sont utilisées pour classer de nouveaux objets. **Nous nous intéressons plus particulièrement à la classification par navigation dans chaque structure.**

O	$\mathcal{I}$							K
	I_1		I_2		I_3			
	m_1	m_2	m_3	m_4	m_5	m_6	m_7	
o_1	×		×				×	1
o_2	×		×				×	
o_3	×			×		×		2
o_4	×			×		×		
o_5	×			×		×		
o_6		×		×		×		3
o_7		×		×		×		
o_8		×		×		×		
o_9		×	×		×			4
o_{10}		×		×	×			

TABLE 4.2: Table binaire correspondant à la Table 4.1

Dans un arbre de classification, pour un nouvel objet, la navigation est effectuée à travers l'arbre, de la racine jusqu'à une feuille. Le nouvel objet est classé dans la classe de la feuille atteinte. **Chaque étape de navigation consiste à valider une modalité de l'unique attribut de test associé aux branches allant du nœud courant vers ses successeurs**. Plus précisément, un nouvel objet atteint le successeur contenant la modalité qu'il vérifie pour l'attribut de test.

Dans le cas d'arbre binaire tel que CART (*cf.* Figure 4.2.2), une branche peut être associée à plusieurs modalités lorsque l'attribut test est défini par plus de deux modalités. Dans ce cas, l'objet valide une seule des modalités de l'ensemble associé à la branche.

Dans un treillis de Galois, la navigation est réalisée de manière similaire. Elle commence au concept minimal $\perp$ jusqu'à un concept terminal. **Chaque étape de navigation consiste à valider un ou plusieurs attributs binaires, *i.e.* une modalité par attribut qualitatif, pour rejoindre l'un des successeurs immédiats du concept courant**. Une fois un concept terminal atteint, le nouvel objet est classé dans la classe de ce concept terminal.

Pour illustrer cette navigation par validation d'attributs, **les arcs représentant la relation $\leq$ entre les concepts peuvent être étiquetés avec les modalités à valider ou encore comme pour l'arbre de classification, avec un attribut test et la modalité associée à ce test.** Nous choisissons cette dernière option telle que présentée dans la Figure 4.2.3 (où les concepts et arcs inutiles pour la classification par navigation ont été supprimés, *cf.* Algorithme 3.5).

Il faut noter que dans l'arbre de classification, un objet ne peut emprunter qu'un

chemin allant de la racine vers les feuilles alors que dans les treillis de Galois plusieurs chemins sont possibles entre le concept minimal et jusqu'au même concept terminal (c'est le cas par exemple des objets o_6, o_7, et o_8 qui peuvent emprunter deux chemins différents menant tous au même concept terminal et à la classe 3. Ces chemins diffèrent selon l'ordre des tests sur les attributs qualitatifs).

4.2.2 Lien de généralisation

Dans un treillis de Galois, chaque étape de navigation correspond à la **validation d'un ensemble d'attributs dans une famille de parties d'attributs**.

Plus formellement, soit le concept (A, B) correspondant à une étape de navigation (les attributs binaires (ou modalités) B ont déjà été validés et les objets A sont donc les objets « possibles »). On considère $(A_1, B_1), ..., (A_\eta, B_\eta)$, les η successeurs immédiats de (A, B). Chaque successeur correspond à un ou plusieurs attributs binaires $\tilde{B_j} = B_j \setminus B$ à valider. Les différentes possibilités de validation se formalisent donc par une famille de parties d'attributs :

$$E = \{\tilde{B_j}, j \in \{1, \ldots, \eta\}\}$$

Cette famille possède quelques propriétés intéressantes [Guillas 07] :

Tout d'abord, un même attribut binaire ne peut pas appartenir à deux concepts successeurs de (A, B) sans appartenir à B (sinon, la propriété de borne inférieure n'est pas vérifiée), d'où l'équation suivante :

$$\forall j,\ j' \in \{1, \ldots, \eta\}\ j \neq j'\ \tilde{B_j} \cap \tilde{B_{j'}} = \emptyset \tag{4.2.1}$$

De plus, tout successeur de (A, B) possède au moins un nouvel attribut, mais peut en posséder plusieurs, d'où l'équation suivante :

$$\forall j \in \{1, ..., \eta\}\ \ |\tilde{B_j}| \geq 1 \tag{4.2.2}$$

Cette notion de famille de parties d'attributs engendre deux différences avec la navigation dans un arbre.

Multiplicité des attributs qualitatifs tests : la première différence, entre les deux navigations, est l'existence de plusieurs attributs qualitatifs tests par étape de navigation dans le treillis contre un unique attribut qualitatif test par étape de navigation dans les arbres. Plus formellement, dans un treillis, pour deux arcs différents partant d'un concept père commun et associés respectivement à $\tilde{B_j}$ et $\tilde{B_{j'}}$ (tels que $|\tilde{B_j}| = |\tilde{B_{j'}}| = 1$), il peut exister deux attributs qualitatifs tests, *i.e.* $\exists z \neq z' \in \{1, \ldots, Z\}$ tels que $\tilde{B_j} = m_{jz}$, $\tilde{B_{j'}} = m_{j'z'}$ et $I_z \neq I_{z'}$.

Forme des tests associés aux branches : la seconde différence réside dans la forme des tests associés aux branches. Dans un arbre, lorsque plusieurs modalités sont associées à une même branche (cas fréquent dans les arbres binaires), elles sont

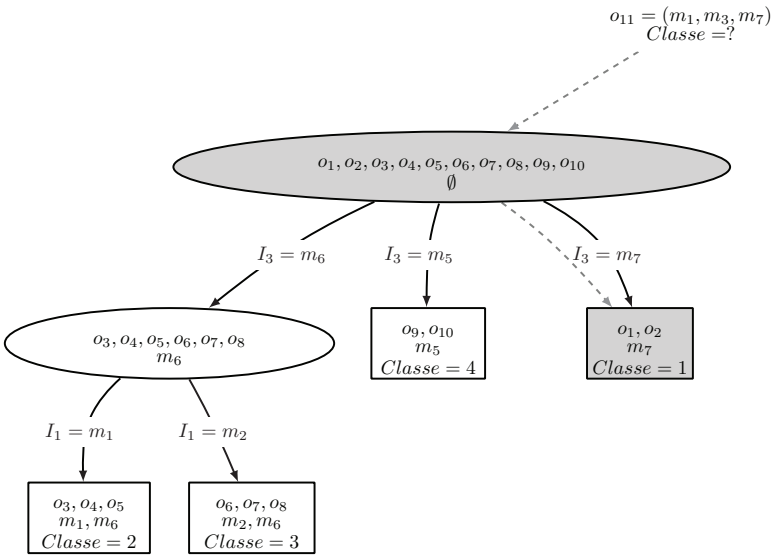


FIGURE 4.2.1: Classification de l'objet o_{11} par navigation dans l'arbre C4.5

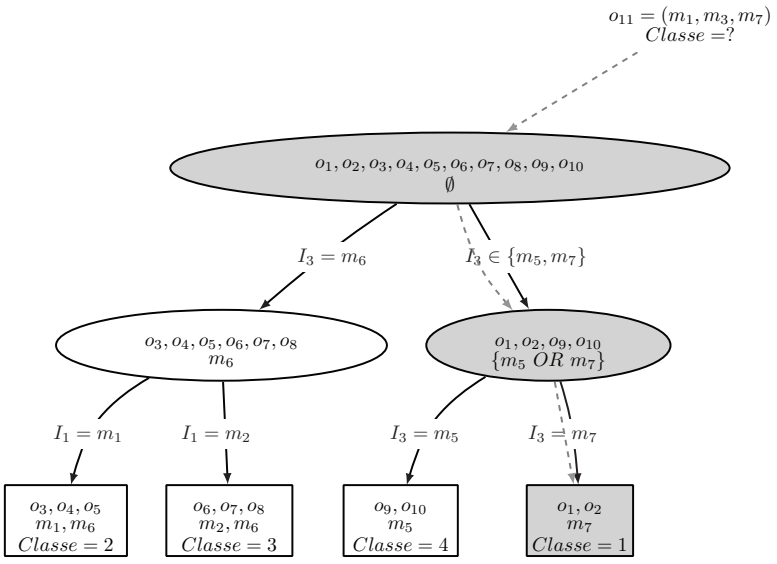


FIGURE 4.2.2: Classification de l'objet o_{11} par navigation dans l'arbre CART

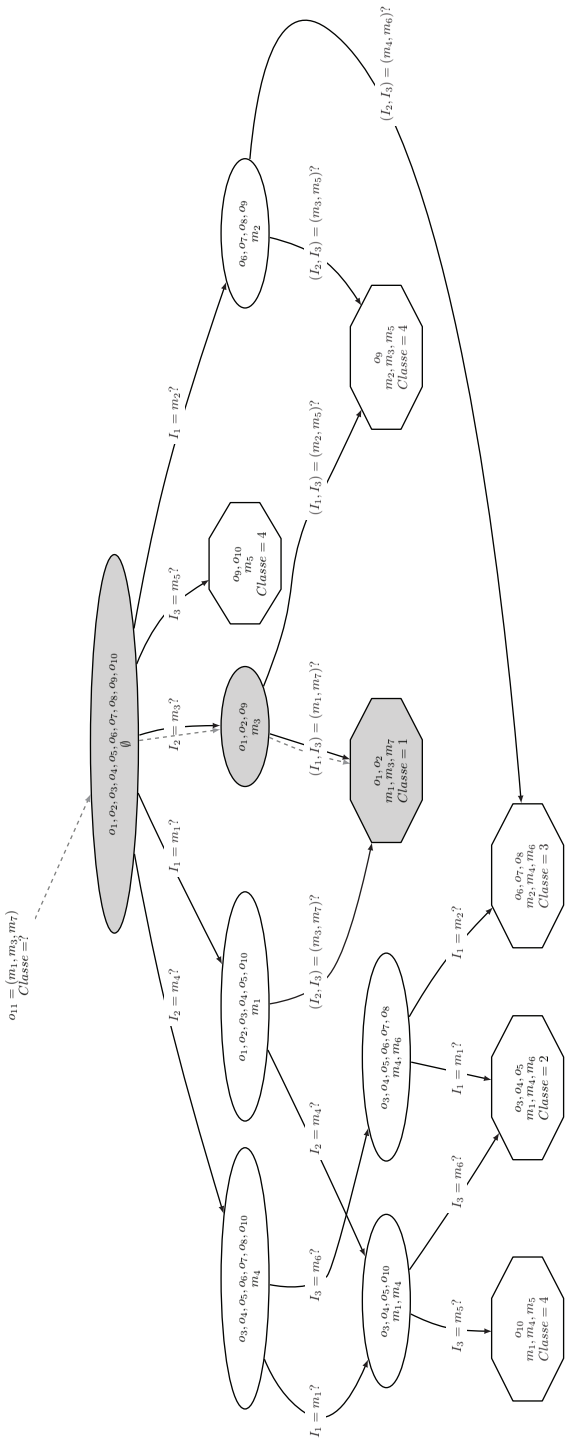


FIGURE 4.2.3: Classification de l'objet o_{11} par navigation dans le diagramme de Hasse

issues du même attribut qualitatif comme nous venons de le voir. Cependant, un objet n'en valide qu'une seule, **ainsi lorsqu'un ensemble de modalités est associé à une branche de l'arbre, le test correspond à un OU exclusif entre ces modalités**. Dans le treillis, lorsque plusieurs modalités sont associées à une même branche, elles sont issues d'attributs qualitatifs différents. Un objet doit valider l'ensemble de ces modalités de manière simultanée, **ainsi lorsqu'un ensemble de modalités est associé à une branche du treillis, le test correspond à un ET entre ces modalités.**

Exemple. Reprenons nos exemples (Figures 4.2.1, 4.2.2 et 4.2.3). Dans le treillis (Figure 4.2.3), à partir de $\perp$, un objet peut rejoindre trois successeurs chacun associé à une modalité m_i différentes, tous les attributs qualitatifs sont concernés, *i.e.* I_1, I_2, et I_3. Dans les arbres des Figures 4.2.1 et 4.2.2 toute étape de navigation est associée à un unique attribut qualitatif test I_z. D'où la différence 1.

De plus, dans le treillis, en considérant comme concept courant $(A, B) = (o_1o_2o_9, m_3)$, il existe deux concepts successeurs $(A_1, B_1) = (o_1o_2, m_1m_3m_7)$ et $(A_2, B_2) = (o_9, m_2m_3m_5)$. La famille de parties d'attributs est donc $E = \{\tilde{B_1}, \tilde{B_2}\} = \{B_1 \setminus B,\ B_2 \setminus B\} = \{\{m_1m_7\}, \{m_2m_5\}\}$. A partir de ce concept (A, B), il faut valider simultanément les attributs I_1 et I_3 respectivement avec les modalités m_1 et m_7 ou avec les modalités m_2 et m_5, ainsi pour cette étape de navigation, $|\tilde{B_1}| = |\tilde{B_2}| = 2 \geq 1$. Dans l'arbre de la Figure 4.2.2 en considérant le nœud racine $\mathcal{N} = (o_1 \dots o_9, \emptyset)$, il existe deux nœuds successeurs $\mathcal{N}_1 = (o_1o_2o_9o_{10}, m_5\ OU\ m_7)$ et $\mathcal{N}_2 = (o_3o_4o_5o_6o_7o_8, m_6)$. A partir de ce nœud $\mathcal{N}$, il faut valider l'attribut qualitatif I_3 avec l'une de ses modalités. Pour rejoindre $\mathcal{N}_2$, il faut valider soit m_5, soit m_7 et non les deux modalités simultanément. D'où la différence 2.

Ces différences sont illustrées dans ces mêmes figures, pour la classification de l'objet $o_{11} = (m_1, m_3, m_7)$, classé systématiquement dans la classe 1. Chaque étape de navigation correspond à un arc en tirets gris, les nœuds/concepts parcourus sont grisés. Les différentes étapes sont reprises dans la Table 4.3.

4.2.3 Discussion

Ainsi, une étape de navigation, dans un **arbre,** correspond à la validation d'une **unique modalité** parmi l'ensemble des modalités d'un **unique attribut qualitatif**. Tandis qu'elle correspond, dans un **treillis de Galois,** à la validation d'un **ensemble de modalités** parmi une **famille de modalités** issues de **différents attributs qualitatifs**.

Nous pouvons noter que, **dans les arbres de classification, le chemin de la racine à une feuille donnée est unique, dans un treillis de Galois, il y a plusieurs chemins à partir de l'élément minimal $\perp$ jusqu'à un même concept terminal. Cette propriété fournit au treillis de Galois une robustesse face aux données**

	Arbre (Figure 4.2.2)	
	Choix parmi	Validation de
Étape 1	l'ensemble des modalités de l'unique attribut qualitatif test I_3	l'ensemble de modalités $\{m_5, m_7\}$ par vérification de m_7
Étape 2	les modalités m_5 et m_7 de l'unique attribut qualitatif test I_3	la modalité m_7

	Treillis (Figure 4.2.3)	
	Choix parmi	Validation de\du
Étape 1	certaines modalités m_1, m_2, m_3, m_4, m_5 des trois attributs qualitatifs tests I_1, I_2, I_3	la modalité m_3 de I_2 *
Étape 2	les deux couples de modalités (m_1, m_7) et (m_2, m_5) associés au couple d'attributs qualitatifs tests (I_1, I_3)	couple de modalités (m_1, m_7)

*selon un critère de support, *i.e.* nombre d'objets du concept successeur le plus faible

TABLE 4.3: Comparaison de la navigation dans les deux structures

bruitées [Guillas 07, Bertet 11]**.** Ceci est une conséquence du choix fait sur plusieurs attributs qualitatifs tests lors d'une étape de navigation dans un treillis de Galois. Cependant, la présence de valeurs aberrantes dans ma base d'apprentissage peut entrainer l'existence de concepts et de chemins de navigation erronés.

La navigation dans les treillis de Galois généralise donc la navigation dans les arbres. Associés à ce lien en usage, des liens structurels transparaissent. Nous les étudions dans les sections suivantes.

4.3 Lien d'inclusion

4.3.1 Présentation

Le lien d'inclusion existant entre treillis de Galois et arbre de classification est le lien qui permet de définir des algorithmes d'extraction d'arbres de classification à partir de treillis de Galois [Belohlávek 07, Bertet 09].

En excluant, pour le moment, les arbres binaires, alors lorsque les deux structures sont définies à partir des mêmes données qualitatives, il est possible d'associer à chaque nœud de l'arbre un **unique concept** du treillis, comme suit :

– Considérons un nœud $\mathcal{N}$ de l'arbre, et l'ensemble des attributs (*i.e.* ensemble des

modalités) $I_{\mathcal{N}}$ contenus dans ce nœud, *i.e.* validés depuis la racine jusqu'à ce nœud. Au nœud $\mathcal{N}$, nous associons le plus petit concept contenant l'ensemble des attributs binaires $I_{\mathcal{N}}$ et les objets qui le vérifient, ce concept est défini comme suit :

$$(\beta(\varphi(I_{\mathcal{N}})), \varphi(I_{\mathcal{N}})) \tag{4.3.1}$$

De plus, la relation entre deux nœuds dans un arbre correspond à un raffinement sur les objets, *i.e.* l'ensemble des objets du nœud fils est inclus dans l'ensemble des objets du nœud père. Cette relation correspond à la relation d'ordre $\leq$ existante dans le treillis entre deux concepts.

A partir de ces deux correspondances, nous pouvons voir qu'un arbre est inclus dans le treillis lorsque les deux structures sont définies à partir des mêmes données qualitatives. Ce lien d'inclusion est formulé par la proposition suivante.

Proposition 4. ***Lien d'inclusion :*** *Tout arbre de classification est inclus dans le treillis de Galois, lorsque ces deux structures sont construites à partir des mêmes attributs qualitatifs.*

L'inclusion de l'arbre dans le treillis de Galois est une conséquence immédiate de la propriété de fermeture d'un treillis.

Démonstration. Considérons l'arbre de classification ainsi que le treillis de Galois issus des mêmes attributs qualitatifs binaires (il faut au préalable procéder à un codage disjonctif de la table de données, *cf.* Chapitre 1). Pour montrer que l'arbre est inclus dans le treillis de Galois, il s'agit alors de prouver les trois points suivants :

1- Deux nœuds différents d'un arbre de classification sont associés à des concepts différents :
Par l'absurde, si deux nœuds $\mathcal{N}_1$ et $\mathcal{N}_2$ sont associés au même concept, alors $\varphi(I_{\mathcal{N}_1}) = \varphi(I_{\mathcal{N}_2})$. Ceci signifie que ce sont les mêmes objets qui partagent les attributs de $I_{\mathcal{N}_1}$ et de $I_{\mathcal{N}_2}$, ce qui est en contradiction avec le fait que les deux nœuds $\mathcal{N}_1$ et $\mathcal{N}_2$ soient deux nœuds différents de l'arbre de classification.

2- Si deux nœuds sont ancêtres dans l'arbre de classification alors leurs concepts associés sont en relation dans le treillis :
Il est clair que si un nœud $\mathcal{N}_1$ est ancêtre d'un nœud $\mathcal{N}_2$ dans l'arbre de classification, alors $I_{\mathcal{N}_1} \subseteq I_{\mathcal{N}_2}$. L'opérateur φ étant isotone, on en déduit que $\varphi(I_{\mathcal{N}_1}) \subseteq \varphi(I_{\mathcal{N}_2})$, et par conséquent que les deux concepts $(\beta(\varphi(I_{\mathcal{N}_1})), \varphi(I_{\mathcal{N}_1}))$ et $(\beta(\varphi(I_{\mathcal{N}_2})), \varphi(I_{\mathcal{N}_2}))$ sont en relation selon $\leq$, *i.e.* $(\beta(\varphi(I_{\mathcal{N}_1})), \varphi(I_{\mathcal{N}_1})) \leq (\beta(\varphi(I_{\mathcal{N}_2})), \varphi(I_{\mathcal{N}_2}))$.

3- A l'inverse, si deux nœuds ne sont pas ancêtres dans l'arbre de classification alors leurs concepts associés ne sont pas en relation dans le treillis :
Si un nœud $\mathcal{N}_1$ n'est pas ancêtre d'un nœud $\mathcal{N}_2$, il s'agit alors de considérer les fils du

plus petit ancêtre commun à $\mathcal{N}_1$ et $\mathcal{N}_2$, et en particulier le fils $\mathcal{N}_1^{'}$ ancêtre de $\mathcal{N}_1$ et le fils $\mathcal{N}_2^{'}$ ancêtre de $\mathcal{N}_2$. Ces deux nœuds $\mathcal{N}_1^{'}$ et $\mathcal{N}_2^{'}$ existent par construction de l'arbre et il est clair que, $\mathcal{N}_1^{'}$ et $\mathcal{N}_2^{'}$ étant frères aucun objet ne peut avoir simultanément les attributs des deux concepts associés, à savoir $\varphi(I_{\mathcal{N}_1^{'}})$ et $\varphi(I_{\mathcal{N}_2^{'}})$. Ce qui se formalise par $\beta(\varphi(I_{\mathcal{N}_1^{'}})) \cap \beta(\varphi(I_{\mathcal{N}_2^{'}})) = \emptyset$. Ensuite, $\mathcal{N}_1^{'}$ étant ancêtre de $\mathcal{N}_1$, on en déduit que $I_{\mathcal{N}_1^{'}} \subseteq I_{\mathcal{N}_1}$, d'où $\varphi(I_{\mathcal{N}_1^{'}}) \subseteq \varphi(I_{\mathcal{N}_1})$ par isotonie de l'opérateur φ, et à l'inverse $\beta(\varphi(I_{\mathcal{N}_1^{'}})) \supseteq \beta(\varphi(I_{\mathcal{N}_1}))$ par définition de β. On aura de même $\beta(\varphi(I_{\mathcal{N}_2^{'}})) \supseteq \beta(\varphi(I_{\mathcal{N}_2}))$ car $\mathcal{N}_2^{'}$ est ancêtre de $\mathcal{N}_2$. On en déduit alors que $\beta(\varphi(I_{\mathcal{N}_2})) \cap \beta(\varphi(I_{\mathcal{N}_1})) = \emptyset$, ce qui prouve que les concepts associés aux nœuds $\mathcal{N}_1$ et $\mathcal{N}_2$ ne sont pas en relation selon $\leq$. □

Notons que un arc de l'arbre n'est pas toujours associé à la relation de couverture $\prec$. Il peut être associée à la relation $\leq$, soit à un arc de transitivité.

Concernant le cas des arbres binaires, il s'agit d'un cas particulier, lorsque l'attribut qualitatif test a plus de deux modalités alors plusieurs modalités sont regroupées sur une même branche. Dans ce cas, le nœud fils, correspondant à cette branche, contient un ensemble de modalités qui n'ont pas toutes été validées (validation par OU exclusif). Or, nous étudions le cas où les modalités des attributs qualitatifs sont mutuellement exclusives. Par définition des concepts, il n'existe pas de concept contenant deux modalités mutuellement exclusives issues du même attribut qualitatif. Aussi, ces nœuds particuliers regroupant plusieurs modalités mutuellement exclusives d'un même attribut qualitatif, ne peuvent pas être associés à un unique concept. Ils sont associés à autant de concepts que de modalités qu'ils contiennent. Ces concepts correspondent à ceux associés aux nœuds qui auraient été construits si l'arbre n'avaient pas été binaire, *i.e.* les nœuds associés à chacune des modalités.

Exemple. Reprenons les arbres des Figures 4.2.1 et 4.2.2 en étiquetant chaque nœud avec une étiquette $\mathcal{N}_i$, on obtient respectivement les arbres des Figures 4.3.1 et 4.3.2. Pour l'arbre CART, comme précisé précédemment, le nœud correspondant à l'ensemble de modalités $\{m_5, m_7\}$ correspond à deux nœuds si l'arbre n'avait pas été binaires. De ce fait nous l'avons étiquetés avec les deux $\mathcal{N}_i$ correspondant à ces nœuds dans l'arbre C4.5.

En considérant le treillis de Galois de la Figure 4.2.3 défini à partir de la même table de données qualitatives (Table 4.1). La Figure 4.3.3 illustre ce lien d'inclusion, les nœuds de l'arbre sont associés aux différents concepts grisés et étiquetés avec les $\mathcal{N}_i$.

Soit le nœud $\mathcal{N}_3 = (o_3o_4o_5o_6o_7o_8, m_6)$ de l'arbre, à ce nœud on associe le concept $(\beta(\varphi(m_6)), \varphi(m_6)) = (o_3o_4o_5o_6o_7o_8, m_4m_6) = (A_1, B_1)$ qui figure bien dans le treillis de Galois. De même, au nœud $\mathcal{N}_5 = (o_3o_4o_5, m_1m_6)$, on associe le concept $(\beta(\varphi(m_1m_6)), \varphi(m_1m_6)) = (o_3o_4o_5, m_1m_4m_6) = (A_2, B_2)$.

Les arcs de l'arbre sont représentés par les arcs en pointillés gris. L'arc reliant le nœud $\mathcal{N}_3$ à son fils $\mathcal{N}_5$ est associé à l'arc représentant la relation $\leq$ entre (A_1, B_1) et (A_2, B_2).

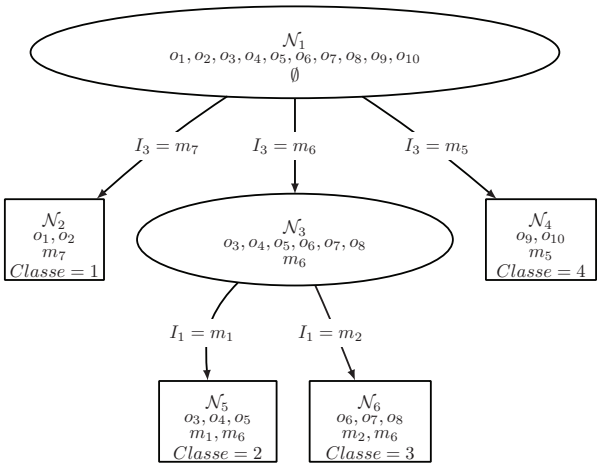


FIGURE 4.3.1: Arbre de classification C4.5

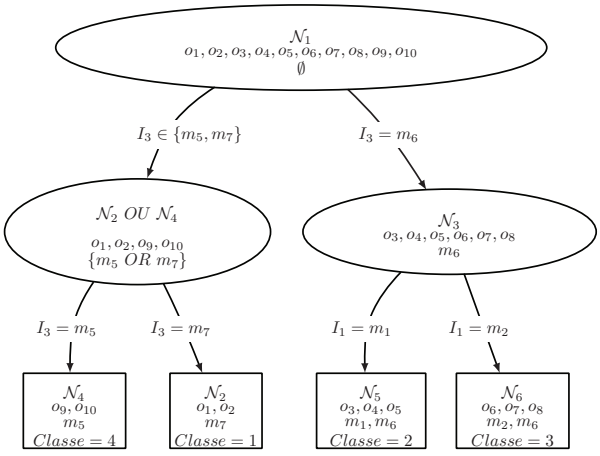


FIGURE 4.3.2: Arbre de classification CART

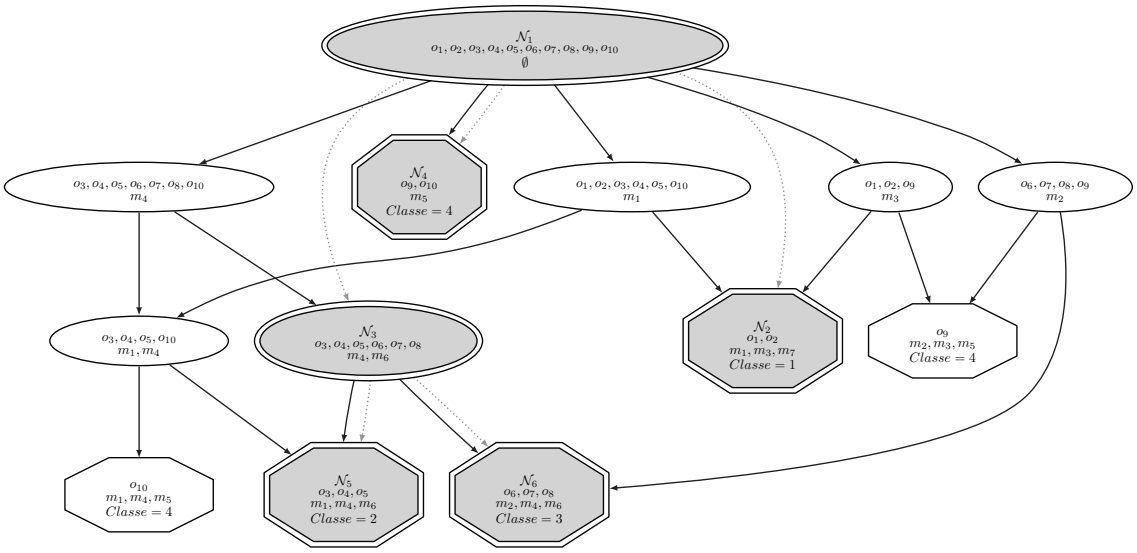


FIGURE 4.3.3: Inclusion de l'arbre de classification dans le treillis de Galois

4.3.2 Discussion

Ce lien d'inclusion est intéressant car il permet de mettre en avant le fait que **le treillis est plus complet que l'arbre**, *i.e.* il contient plus de chemins vers un même concept terminal. En effet, à partir d'une même table de données, un seul et unique treillis peut être défini, tandis que plusieurs arbres peuvent être construits (selon l'algorithme et les critères choisis). Tous ces arbres sont inclus dans l'unique treillis, ce dernier regroupe donc plus d'informations que les arbres.

Ce lien structurel d'inclusion des arbres dans le treillis est renforcé par un lien de fusion, lorsque le treillis est un treillis dichotomique. Nous nous intéressons à cette classe de treillis et au lien de fusion résultant dans la section suivante.

4.4 Contribution 1 : treillis dichotomiques et lien de fusion

Les treillis dichotomiques ont été introduits dans [Bertet 09] selon une propriété portée par la table. Cette propriété correspond à la complémentarité entre attributs d'une table binaire. Dans cette section, **nous donnons la définition formelle des treillis dichotomiques** (définition 18), puis nous étudions les propriétés de ces treillis. Enfin, nous nous intéressons au lien de fusion existant entre ces treillis et les arbres.

4.4.1 Définition des treillis dichotomiques

Les treillis dichotomiques sont des treillis définis à partir d'une table binaire où à tout attribut, il est possible d'associer un ensemble d'attributs complémentaires. Ce type de table est fréquemment rencontré en analyse de données, en particulier dans le cas où les attributs sont qualitatifs, et de modalités mutuellement exclusives. Nous verrons aussi dans le chapitre suivant que la discrétisation (transformation des données du type quantitatif au type qualitatif), génère ce type de table. Il en résulte des treillis ayant de propriétés structurelles fortes, qui permettent le développement d'algorithmes spécifiques à des fins d'amélioration des performances des treillis en classification (*cf.* chapitre 5). Intéressons nous tout d'abord à la définition formelle de ces treillis.

Définition formelle des treillis dichotomiques

Définition 18. Un treillis $\mathcal{L} = (\mathcal{C}, \leq)$ est dit dichotomique lorsqu'il est défini pour un contexte $(O, \mathcal{I}, (\alpha, \beta))$ où pour chaque attribut $I_j \in \mathcal{I}$, il existe une famille non vide $\bar{I}_j = \left\{\bar{I}_j^1, \bar{I}_j^2, \ldots, \bar{I}_j^\partial\right\} \subset \mathcal{P}(\mathcal{I})$ de parties d'attributs complémentaires (*i.e.* chaque $\bar{I}_j^i$ est un ensemble d'attributs binaires qui peut être un ensemble singleton), et tel que l'ensemble $\left\{\beta(I_j), \beta(\bar{I}_j^1), \ldots, \beta(\bar{I}_j^\partial)\right\}$ forme une partition de l'ensemble des objets O :
- Les objets vérifiant un ensemble $\bar{I}_j^i$ de $\bar{I}_j$ sont distincts des objets vérifiant I_j mais aussi des objets vérifiant tout autre ensemble $\bar{I}_j^k$ de $\bar{I}_j$:

$$\begin{array}{c} \forall i \in \{1, \ldots, \partial\}\ \beta(I_j) \cap \beta(\overline{I_j^i}) = \emptyset \\ \forall i, k \in \{1, \ldots, \partial\}\ (i \neq k)\ \beta(\bar{I}_j^i) \cap \beta(\overline{I_j^k}) = \emptyset \end{array} \tag{4.4.1}$$

- Chaque objet de O vérifie soit I_j soit un ensemble d'attributs de $\bar{I}_j$:

$$\beta(I_j) \bigcup_{i \in \{1,\ldots,\partial\}} \beta(\overline{I_j^i}) = O \tag{4.4.2}$$

Notons que dans cette définition, la notion de famille est nécessaire en raison de la possibilité de réduction du contexte, nous précisons ce point dans la section suivante.

Cette définition généralise la définition d'une table dichotomique introduite dans [Ganter 99]. Dans une table dichotomique, pour chaque attribut I_j, il existe un **unique** attribut complémentaire, *i.e.* $\bar{I}_j = \{\bar{I}_j^1\}$ et $\bar{I}_j^1$ est un singleton.

Le caractère unique de l'attribut complémentaire implique que chaque objet d'une table dichotomique vérifie soit I_j soit $\bar{I}_j^1$. Ainsi, **tout treillis construit à partir d'une table dichotomique est un treillis dichotomique**. Mais l'inverse n'est pas vrai, *i.e.* la table d'un treillis dichotomique n'est pas toujours une table dichotomique.

Exemple. Toute table binaire issue de la binarisation d'une table où les attributs qualitatifs ont exactement deux modalités est une table dichotomique. En effet dans ce cas, les modalités sont complémentaires deux à deux. Ainsi, le treillis associé à ce type de contexte est un treillis dichotomique.

En revanche, la Table 4.4(i) n'est pas dichotomique, en effet, l'attribut complémentaire de I_5 n'existe pas de manière unique, tel que présenté dans la Table 4.4(ii). Cependant, le treillis associé (Figure 4.4.1) est dichotomique car pour tout attribut du contexte de la Table 4.4(i), il existe une famille non vide de parties d'attributs complémentaires telles que présentées dans la Table 4.4(ii).

O	$\mathcal{I}$							K
	I_1	I_2	I_3	I_4	I_5	I_6	I_7	
o_1	×		×				×	1
o_2	×		×				×	
o_3	×			×		×		2
o_4	×			×		×		
o_5	×			×		×		
o_6		×		×		×		3
o_7		×		×		×		
o_8		×		×		×		
o_9		×	×		×			4
o_{10}	×			×	×			

(i) Table non dichotomique

$\mathcal{I}$	Complémentaires
I_1	$\bar{I_1} = \{\{I_2\}\}$
I_2	$\bar{I_2} = \{\{I_1\}\}$
I_3	$\bar{I_3} = \{\{I_4\}\}$
I_4	$\bar{I_4} = \{\{I_3\}\}$
I_5	$\bar{I_5} = \{\{I_6\}, \{I_7\}\}$
I_6	$\bar{I_6} = \{\{I_5\}, \{I_7\}\}$
I_7	$\bar{I_7} = \{\{I_5\}, \{I_6\}\}$

(ii) Attributs complémentaires

TABLE 4.4: Table non dichotomique associée au treillis dichotomique Figure 4.4.1

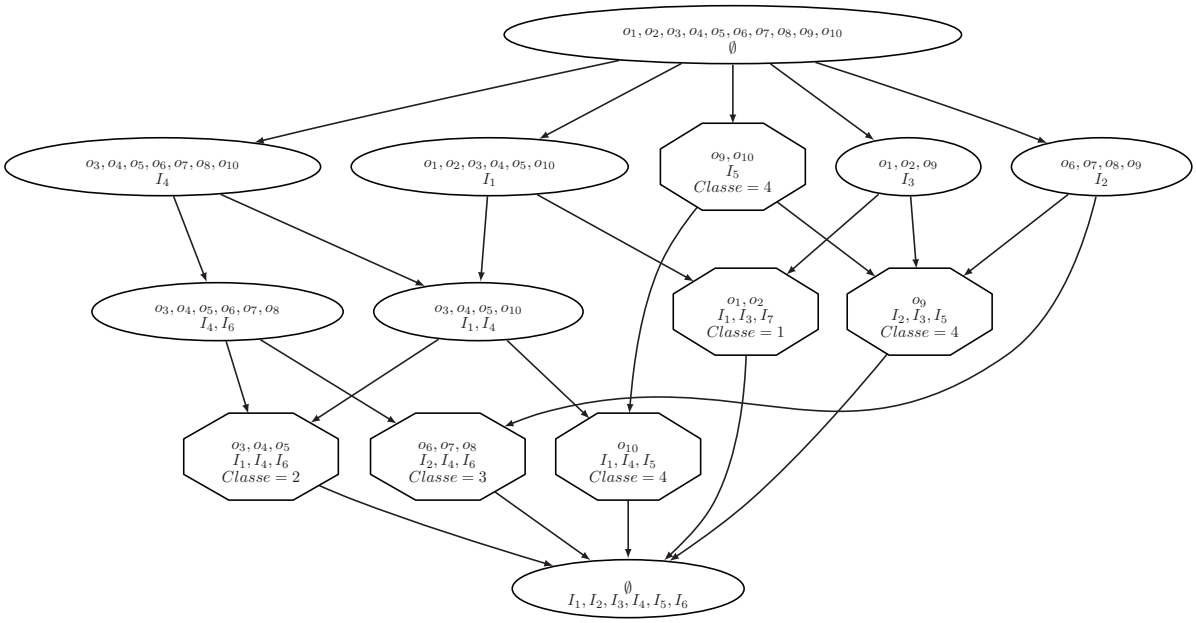


FIGURE 4.4.1: Treillis dichotomique issu de la Table 4.4 non dichotomique

4.4.2 Propriétés des treillis dichotomiques

Cette section a pour objectifs de présenter et d'étudier les propriétés de la classe de treillis dichotomiques que nous venons de définir. Cela nous permet, dans la suite de ce manuscrit, d'exploiter ces propriétés pour définir nos algorithmes de discrétisation locale (*cf.* chapitre 5).

Dichotomie et réduction

L'opération de réduction d'un contexte est généralement mise en œuvre avant toute génération du treillis de Galois pour la classification, cela afin de diminuer la redondance des informations présentes dans les concepts. En effet, la table réduite est la plus petite de toutes les tables possédant un treillis isomorphe, *i.e.* une même structure de classification. C'est pourquoi nous nous intéressons aux conséquences potentielles de cette opération sur la propriété de dichotomie et plus particulièrement à la réduction sur les attributs, ces derniers étant la base de la définition des treillis dichotomiques.

Par définition de la réduction (*cf.* section 3.2.3), pour tout contexte non réduit $(O, \mathcal{I}, (\alpha, \beta))$ vérifiant la définition 18, sa réduction vérifie également la définition 18 ; si le treillis du contexte non réduit est dichotomique alors le treillis du contexte réduit l'est aussi.

En effet, tout $I \in \mathcal{I}$ supprimé du contexte par l'opération de réduction, est équivalent à un sous-ensemble I' d'attributs de $\mathcal{I}$ dans le contexte réduit. Ainsi, si I était complémentaire d'un ou plusieurs attribut(s) dans le contexte non réduit, alors sa suppression est compensée par l'ensemble d'attributs équivalents I' car, par définition de la réduction, celle-ci ne modifie pas le treillis.

Exemple. Dans le contexte de la Table 4.4, après une opération de réduction, l'attribut I_7 serait supprimé car équivalent à l'ensemble $I' = \{I_1, I_3\}$ (*i.e.* $\beta(\{I_7\}) = \{o_1, o_2\} = \beta(\{I_1, I_3\})$), le contexte réduit correspond alors à la Table 4.5.

I_7 appartenait à l'ensemble des complémentaires de I_6 et de I_5. Par définition de la réduction, sa suppression ne modifie pas le treillis, et nous pouvons vérifier que les équations 4.4.1 et 4.4.2 sont toujours vérifiées. Les attributs I_1 à I_4 ayant conservé leurs complémentaires, vérifions que I_5 et I_6 ont bien des complémentaires dans le contexte réduit. Dans le contexte réduit, I_7 a été supprimé mais l'ensemble $I' = \{I_1, I_3\}$ sert d'équivalent. Ainsi la famille de parties d'attributs complémentaires de I_5 dans le contexte réduit est $\{\{I_1, I_3\}, \{I_6\}\}$. L'équation 4.4.1 est toujours vérifiée : $\beta(\{I_5\}) \cap \beta(\{I_1, I_3\}) = \emptyset$, $\beta(\{I_5\}) \cap \beta(\{I_6\}) = \emptyset$ et $\beta(\{I_6\}) \cap \beta(\{I_1, I_3\}) = \emptyset$. Tout comme l'équation 4.4.2 : $\beta(\{I_5\}) \cup \beta(\{I_1, I_3\}) \cup \beta(\{I_6\})) = O$

La famille de parties d'attributs complémentaires de I_6 dans le contexte réduit est $\{\{I_1, I_3\}, \{I_5\}\}$. L'équation 4.4.1 est toujours vérifiée : $\beta(\{I_6\}) \cap \beta(\{I_1, I_3\}) = \emptyset$, $\beta(\{I_6\}) \cap \beta(\{I_5\}) = \emptyset$ et $\beta(\{I_1, I_3\}) \cap \beta(\{I_5\}) = \emptyset$. L'équation 4.4.2 aussi : $\beta(\{I_6\}) \cup \beta(\{I_1 I_3\}) \cup \beta(\{I_5\})) = O$.

O	$\mathcal{I}$						K
	I_1	I_2	I_3	I_4	I_5	I_6	
o_1	×		×				1
o_2	×		×				
o_3	×			×		×	2
o_4	×			×		×	
o_5	×			×		×	
o_6		×		×		×	3
o_7		×		×		×	
o_8		×		×		×	
o_9		×	×		×		4
o_{10}	×			×	×		

TABLE 4.5: Contexte réduit de la Table 4.4

Notons que **la notion de famille utilisée dans la définition des treillis dichotomiques montrent là toute sont importance**. En effet, si ceux-ci avaient été définis en utilisant uniquement un ensemble d'attributs complémentaires, *i.e.* tous les $\bar{I}_j^i$ auraient du être des singletons, alors la propriété de dichotomie aurait alors été perdue après réduction du contexte.

Propriétés structurelles des treillis dichotomiques

Nous avons observé que les treillis dichotomiques ont des propriétés structurelles particulières telles que la coatomicité (*i.e.* les inf-irréductibles du treillis sont ses coatomes) **et la $\vee$-pseudo-complémentarité** (*i.e.* existence d'au moins un concept complémentaire pour tout concept du treillis ; *cf.* section 3.2.2 et propriétés 5 et 6 ci-après). **A partir de ces propriétés fortes, nous pouvons envisager le développement d'algorithmes propres à ces treillis.** En effet, ces propriétés permettent d'identifier les concepts terminaux du treillis, en particulier, les coatomes du treillis sont des concepts terminaux. Ces concepts sont utilisés pour généraliser aux treillis le processus de discrétisation locale existant dans les arbres, qui améliore les performances de ces derniers en classification.

Avant de présenter ces développements dans le chapitre suivant, nous donnons ci-dessous ces propositions et leurs preuves.

Proposition 5. *Tout treillis dichotomique est coatomistique.*

Démonstration. Par l'absurde, soit un treillis dichotomique $\mathcal{L} = (\mathcal{C}, \leq)$. Supposons que ce treillis ne soit pas coatomistique. Alors, il existe $M_1, M_2 \in M$ deux inf-irréductibles tels que $M_1 = (A_1, B_1) < M_2 = (A_2, B_2)$. Par définition de la relation $<$, on a $A_2 \subset A_1$ et $B_1 \subset B_2$.

Il existe donc $b \in B_2 \setminus B_1$ et $a \in A_1 \setminus A_2$ tels que $a \notin \beta(b)$ (et réciproquement $b \notin \alpha(a)$). Par définition des inf-irréductibles (*cf.* bijection entre objets et inf-irréductibles, proposition 2), $\beta(\alpha(a)) = A_1$. Selon la relation $<$ entre M_1 et M_2, $\beta(b) \cap \beta(\alpha(a)) \neq \emptyset$.

Soit $\bar{B} = \{\bar{B^i},\ i \in \{A, \ldots, \partial\}\}$, la famille de parties d'attributs complémentaires de b.

Si $\exists i \in \{1, \ldots, \partial\}$ tel que $\alpha(a) \subset \bar{B^i}$ alors il y a contradiction avec la définition des treillis dichotomiques : l'équation 4.4.1 n'est pas vérifiée, $\beta(b) \cap \beta(\alpha(a)) \neq \emptyset$. Donc $\forall i \in \{1, \ldots, \partial\}$, $\alpha(a) \not\subseteq \bar{B^i}$.

Ainsi $a \notin \bigcup_{i \in \{1,\ldots,\partial\}} \beta(\overline{B^i})$, de plus $a \notin \beta(b)$, il y a de nouveau une contradiction avec la définition des treillis dichotomiques : l'équation 4.4.2 n'est pas vérifiée, $a \notin \beta(b) \bigcup_{i \in \{1,\ldots,\partial\}} \beta(\overline{B^i}) \neq O$.

Ainsi, si un treillis est dichotomique alors il est coatomistique. □

Un résultat établit que tout treillis coatomistique est $\vee$-pseudo-complémenté [Barbut 70, Crawley 73]. Nous pouvons en déduire directement la proposition suivante, cependant nous en donnons une preuve sans utiliser la coatomicité.

Proposition 6. *Tout treillis dichotomique est $\vee$-pseudo-complémenté.*

Démonstration. Pour montrer que tout treillis dichotomique est $\vee$-pseudo-complémenté, considérons un concept quelconque (A, B) d'un treillis dichotomique. Il s'agit de montrer l'existence d'un concept complémentaire.

Considérons un attribut binaire quelconque b de B et sa famille de parties d'attributs complémentaires $\bar{B} = \{\bar{B^i},\ i \in \{A, \ldots, \partial\}\}$. Il s'en suit que les objets possédant b, et ceux possédant $\overline{B^i}$ $(\forall i \in \{1, \ldots, \partial\})$ sont différents, ce qui se formalise par $\forall i \in \{1, \ldots, \partial\}$, $\beta(\{b\}) \cap \beta(\{\overline{B^i}\}) = \emptyset$. Soit $\bar{B^i}$ un élément de $\bar{B}$, on considère alors le plus petit concept contenant $\overline{B^i}$ qui, par définition, sera le concept $(\beta(\varphi(\{\overline{B^i}\})), \varphi(\{\overline{B^i}\}))$ dont l'ensemble des attributs est $\varphi(\{\overline{B^i}\})$. On déduit de la définition des applications α et β (*cf.* définition 15) que $\beta(\varphi(\{\overline{B^i}\})) = \beta(\{\overline{B^i}\})$, et que $A \subseteq \beta(\{b\})$. Étant donné que $\beta(\{b\}) \cap \beta(\{\overline{B^i}\}) = \emptyset$, on peut alors en déduire que $A \cap \beta(\varphi(\{\overline{B^i}\})) = \emptyset$. Par conséquent, $(A, B) \vee (\beta(\varphi(\{\overline{B^i}\})), \varphi(\{\overline{B^i}\})) = (\emptyset, \mathcal{I}) = \top$, et le concept $(\beta(\varphi(\{\overline{B^i}\})), \varphi(\{\overline{B^i}\}))$ est un concept complément de (A, B), ce qui prouve la $\vee$-pseudo-complémentarité du treillis. □

La propriété de $\vee$-pseudo-complémentarité garantit l'existence, pour chaque concept, d'un concept complémentaire par rapport à $\top$. Ce concept complémentaire n'est pas unique.

Notons que l'inverse n'est pas toujours vrai, *i.e.* un treillis $\vee$-pseudo-complémenté n'est pas toujours dichotomique.

Exemple. La Figure 4.4.2 donne un exemple de treillis $\vee$-complémenté (*i.e.* $\vee$-pseudo-complémenté) qui n'est pas un treillis dichotomique. Le contexte correspondant est donné dans la Table 4.6. On peut voir que I_3 n'a pas d'attributs complémentaires au sens de la définition 18.

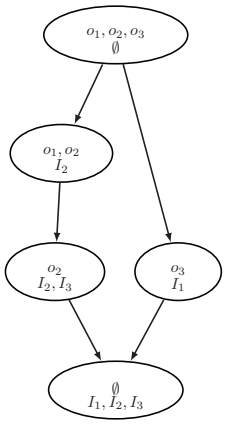


Figure 4.4.2: Un treillis $\vee$-complémenté non dichotomique

O \ I	I_1	I_2	I_3
o_1		$\times$	
o_2		$\times$	$\times$
o_3	$\times$		

TABLE 4.6: Contexte du treillis de la Figure 4.4.2

4.4.3 Lien de fusion

Les treillis dichotomiques possèdent un lien structurel fort avec les arbres de classification, ce lien est défini comme suit.

Proposition 7. ***Liens de fusion :*** *Un treillis dichotomique est la fusion (union ensembliste) de tous les arbres de classification lorsque ces structures sont construites à partir des mêmes attributs binaires.*

Démonstration. Nous avons déjà montré que tout arbre de classification était inclus dans le treillis dichotomique construit à partir des mêmes attributs binaires. Pour prouver que le treillis dichotomique est en fait la fusion de tous les arbres de classification, il suffit de montrer que tout concept appartient à un arbre de classification, ce que nous allons faire par construction.

Considérons un concept quelconque (A, B). On construit alors le sous-ensemble $\mathcal{C}$ de concepts du treillis contenant :
- le concept (A, B),
- un concept $(\beta(\bar{B}), \bar{B})$ complémentaire à (A, B),
- le concept minimal $\perp$,
- tous les concepts terminaux successeurs de (A, B) et de $(\bar{A}, \bar{B})$.

L'existence du concept complémentaire $(\beta(\bar{B}), \bar{B})$ se déduit de la propriété de $\vee$-pseudo-complémentarité du treillis dichotomique. De plus, elle induit que ce sous-ensemble $\mathcal{C}$ équipé de la relation $\leq$, forme un arbre. On ajoute alors dans l'ensemble $\mathcal{C}$ un nombre maximum de concepts du treillis dichotomique de telle sorte que la borne supérieure de deux éléments non comparables de l'ensemble ordonné $(\mathcal{C}, \leq)$ soit le top $\top$. Nous obtenons ainsi par construction un sous-arbre inclus dans le treillis dichotomique, contenant (A, B). Ses feuilles, qui sont des concepts terminaux, correspondent à des sous-ensembles d'objets qu'aucun attribut binaire ne peut séparer, *i.e.* des sous-ensembles d'objets appartenant majoritairement à la même classe. Cet arbre est donc un arbre de classification, ce qui termine cette preuve. □

La propriété de fusion se déduit de la $\vee$-pseudo-complémentarité. Elle est par conséquent propre aux treillis dichotomiques, en effet, la complémentarité sur l'ensemble des attributs, base de la définition des treillis dichotomiques, garantit l'existence de concepts complémentaires dans le treillis, ces concepts complémentaires correspondant à des nœuds frères dans les arbres. Cette propriété de fusion renforce ainsi les liens structurels entre treillis dichotomiques et arbres. La preuve précédente montre que tout sous-arbre d'un treillis de Galois est un arbre séparant les objets de classes différentes, *i.e.* un arbre de classification.

4.5 Conclusion

Les treillis de Galois et les arbres de classification sont deux structures arborescentes qui ont été définies dans des cadres différents. Alors que le treillis de Galois a été défini dans le cadre de la théorie des graphes, l'arbre a été défini par les statisticiens pour l'analyse de données. L'utilisation du treillis de Galois pour l'analyse de données est arrivée plus tard, les développements associés n'ont généralement pas été liés au développement des arbres de classification.

Ce chapitre nous a permis de mettre en évidence les **liens structurels très forts** qui unissent les structures hiérarchiques et en particulier les arbres de classification avec les treillis de Galois.

Nous avons **présenté et formalisé la définition des treillis dichotomiques**. Nous avons présenté et démontré ses propriétés. **Ces propriétés structurelles fortes jouent un rôle très important dans la suite des travaux** présentés dans ce manuscrit et **tout particulièrement la propriété de coatomicité** (proposition 5).

Les liens présentés nous ont amené à développer un modèle hybride entre arbres et treillis de Galois pour les tâches de classification. En effet en analyse de données, et

particulièrement en analyse de données issues d'images, nous avons à traiter des données quantitatives. Afin de pouvoir traiter ces données, les deux structures symboliques utilisent une discrétisation afin de transformer ces données en type qualitatif. Comme nous le verrons dans le chapitre suivant, cette transformation rend le treillis construit dichotomique. Cependant, les deux structures n'utilisent pas le même type de discrétisation, c'est l'étude de cette différence présentée dans le chapitre 5 qui nous a permis de développer le modèle hybride.

Points clés

Positionnement

- ❏ Nous avons rappelé les liens existants entre treillis de Galois et arbres de classification.
- ❏ Après avoir défini formellement la classe des treillis dichotomiques, nous avons présenté ses propriétés et ses liens avec les arbres.

Contributions

- ❏ Nous avons défini formellement les treillis dichotomiques (définition 18).
- ❏ Nous avons démontré leur coatomicité (proposition 5).
- ❏ Nous avons démontré leur $\vee$-pseudo-complémentarité (proposition 6).

Chapitre 5

Transformation des données quantitatives - Discrétisation

5.1 Introduction

Comme nous l'avons précisé lors de la présentation des différents modèles de classification (section 1.2), ces derniers sont souvent définis pour un seul type de données. Pour pallier à cette limitation, il existe des processus permettant de transformer le type des données, du type quantitatif au type qualitatif et inversement. La transformation du type qualitatif au type quantitatif peut être réalisée par exemple au moyen d'une analyse des correspondances multiples [Tufféry 10], ou encore par construction de concepts [Njiwoua 99] et la transformation du type quantitatif au type qualitatif est réalisée *via* un processus de **discrétisation** [Catlett 91, Weiss 91, Dougherty 95, Muhlenbach 05].

Dans le cadre de la classification d'images, les signatures extraites des images sont généralement constituées de descripteurs quantitatifs. En raison en particulier de leur lisibilité, nous utilisons des modèles de classification symboliques. Initialement, ces derniers ont été définis pour le traitement de données qualitatives, voire qualitatives binaires pour les treillis de Galois. **Nous nous intéressons donc dans ce chapitre à la transformation du type quantitatif au type qualitatif, *i.e.* à la discrétisation.**

La section 5.2 présente la définition générale de la discrétisation. Nous présentons les différents types de discrétisation et nous nous intéressons particulièrement à **la différence entre discrétisation globale et discrétisation locale**. En effet, nous verrons qu'une des différences existantes entre arbre de classification et treillis de Galois réside dans leur traitement des données quantitatives : le premier utilise généralement une discrétisation locale tandis que le second utilise généralement une discrétisation globale.

Dans la section 5.3, **nous présentons la méthode de discrétisation locale que**

nous avons définie pour les treillis de Galois. Différentes études montrent la supériorité d'une discrétisation locale pour les arbres de classification [Dougherty 95, Quinlan 93, Quinlan 96b] sur la discrétisation globale. Nous montrons qu'il en est de même pour les treillis tant du point de vue de la structure du treillis de Galois que de ses performances en classification.

5.2 Principe de la discrétisation

La discrétisation a pour objectif de **transformer les attributs quantitatifs en attributs qualitatifs dont les modalités sont des intervalles de valeurs des attributs quantitatifs initiaux** [Fayyad 93, Dougherty 95, Frank 99, Liu 02, Muhlenbach 02]**.**

La discrétisation fait partie des transformations dites avec perte d'informations [Celeux 93]. En effet, une fois la transformation réalisée, il n'est plus possible, à partir des intervalles construits, de retrouver les valeurs initiales associées à chaque objet.

L'étape de discrétisation est donc une étape importante dans la définition des modèles de classification symboliques à partir de données quantitatives [Rakotomalala 97]. Si le découpage des attributs quantitatifs en intervalles est mal réalisé, cela peut faire diminuer les performances du modèle construit [Celeux 93].

La discrétisation consiste donc à créer des intervalles de valeurs à partir d'une variable quantitative. Plus formellement, soit un attribut quantitatif $V \in \mathcal{V}$, **discrétiser** V **c'est couper son domaine de définition** $[min_V, max_V]$ en plusieurs intervalles de valeurs. Pour définir les intervalles, il faut **trouver des valeurs particulières du domaine de** V **qui serviront de bornes aux intervalles**, ces valeurs sont appelées **points de coupe** (ou encore points frontières).

Plus précisément, soit $(v_1, \ldots, v_N)$ les valeurs observées et triées de l'attribut $V \in \mathcal{V}$. Chaque v_i ($\forall i \in \{1, \ldots, N\}$) correspond à une valeur vérifiée par un objet de l'ensemble de données considéré et $\forall i < j \in \{1, \ldots, N\}$ $v_i < v_j$. Un ensemble de points de coupe candidats est défini à partir de ces valeurs, nous notons cet ensemble $C_V = \{c_V^1, \ldots, c_V^{N-1}\}$.

Dans la littérature, nous avons rencontré deux types de discrétisation, qui définissent les points de coupe candidats de manières différentes. Le premier type est le plus usuel, il correspond à **la discrétisation en intervalles disjoints**, le second rencontré dans le cadre de l'analyse formelle des concepts [Ganter 99] correspond à **la discrétisation en intervalles non disjoints**.

Remarque. Notons que pour accélérer les processus de discrétisation (en intervalles disjoints ou non), il est courant dans le cas supervisé, de considérer comme points de coupe candidats uniquement les points de coupe situés sur les frontières entre les classes différentes [Rakotomalala 05a], il existe aussi des heuristiques permettant de sélectionner les points à évaluer (*i.e.* d'éliminer certains points de C_V), voir par exemple [Fayyad 92].

Dans la suite de cette présentation, nous considérons tous les points de coupe possible comme candidats.

5.2.1 Intervalles disjoints ou non disjoints

Discrétisation en intervalles non disjoints

Dans le cadre de l'analyse formelle de concepts, Ganter et Wille [Ganter 99] ont proposé une discrétisation en intervalles non disjoints, aussi appelée échelonnage ou *scaling*.

Dans ce contexte, **l'ensemble des points de coupe candidats** pour discrétiser/échelonner l'attribut quantitatif $V \in \mathcal{V}$, correspond à l'ensemble des valeurs observées, ainsi $C_V = \{v_1, \ldots, v_N\}$.

La discrétisation en intervalles non disjoints se fait en deux étapes. La première consiste à construire un échelonnage par attribut quantitatif à discrétiser. Puis la seconde consiste à regrouper dans une table binaire (*i.e.* un contexte), les informations issues des valeurs observées pour chaque objet de l'ensemble de données et celles issues de chaque échelonnage.

L'échelonnage d'un attribut quantitatif correspond à une table binaire, avec en lignes les valeurs observées pour cet attribut et en colonnes ces mêmes valeurs associées à un ou plusieurs opérateurs de comparaison. Plus précisément, pour l'échelonnage d'attributs quantitatifs, les opérateurs utilisés sont les opérateurs $\leq$ et $\geq$. Lorsque seul le premier opérateur est utilisé, l'échelonnage est dit ordinal, lorsque les deux opérateurs sont utilisés l'échelonnage est dit inter-ordinal [1].

Une fois l'échelonnage construit, chaque colonne de ce dernier (*i.e.* chaque valeur observée pour l'attribut échelonné) devient une colonne du contexte binaire. Les objets sont ensuite associés à ces colonnes en suivant les relations établies dans l'échelonnage précédent, pour la valeur vérifiée par chaque objet.

A partir du contexte formel obtenu, plusieurs intervalles non disjoints sont implicitement définis.

Pour le traitement des données quantitatives continues, l'échelonnage inter-ordinal est généralement utilisé [Ganter 99, Kaytoue 11].

Exemple. Illustrons nos propos avec un exemple simple. Soit la Table 5.1(i) (extraite de la Table 5.6), en lui appliquant l'échelonnage inter-ordinal, on obtient le contexte binaire présenté dans la Table 5.2. Pour exemple, l'échelonnage inter-ordinal de chaque

1. Notons qu'il existe aussi l'échelonnage nominal pour les attributs qualitatifs. Pour plus de précisions sur les différents échelonnages, *cf.* [Ganter 99] p. 36-43.

attribut $V_{.1}$ est donné dans la Table 5.1(ii). Les intervalles non disjoints définis implicitement, à partir des colonnes définies pour $V_{.1}$ par exemple, sont les intervalles $[0,0]; [0,1]; [0,3]; [1,1]; [1,3]; [3,3]$.

O	$\mathcal{V}$			K
	$V_{.1}$	$V_{.2}$	$V_{.3}$	
o_1	1	4	15	1
o_2	0	0	18	
o_3	1	12	16	2
o_4	0	16	17	
o_5	3	12	18	

(i) Ensemble de données quantitatives

	≤ 0	≤ 1	≤ 3	≥ 0	≥ 1	≥ 3
0	×	×	×	×		
1		×	×	×	×	
3			×	×	×	×

(ii) Échelonnage inter-ordinal de l'attribut $V_{.1}$

Table 5.1: Discrétisation en intervalles non disjoints

Discrétisation en intervalles disjoints

La discrétisation en intervalles disjoints est la discrétisation la plus utilisée dans la littérature [Fayyad 93, Dougherty 95, Frank 99, Liu 02, Muhlenbach 02]. Dans ce cas, **l'ensemble des points de coupe candidats** pour discrétiser l'attribut quantitatif $V \in \mathcal{V}$, correspond à l'ensemble des valeurs intermédiaires situées entre deux valeurs consécutives de V. Ainsi, le plus souvent, $C_V = \{c_V^1, \ldots, c_V^{N-1}\}$ où $c_V^i = \frac{v_i + v_{i+1}}{2}$ $\forall i \in \{1, \ldots, N-1\}$.

La discrétisation est réalisée par **une sélection** de certains points de coupe c_V^i de C_V. Les intervalles disjoints construits sont de la forme : $[min_V, c_V^{i_1}]$, $]c_V^{i_1}, c_V^{i_2}]$, $]c_V^{i_2}, max_V]$.

Remarque. Dans la littérature, il existe deux manières de définir les bornes des intervalles disjoints. Pour Rakotomalala [Rakotomalala 97], les points de coupe choisis sont inclus dans les intervalles supérieurs, *i.e.* les intervalles sont de la forme $[min_V, c_V^{i_1}[$, ..., $[c_V^i, max_V]$. Tandis que pour Fayyad et Irani [Fayyad 93], les points de coupe sont inclus dans les intervalles inférieurs, *i.e.* les intervalles sont de la forme $[min_V, c_V^{i_1}]$, ...,$]c_V^i, max_V]$. Nous utilisons les bornes telles que définies par Fayyad et Irani [Fayyad 93].

Par discrétisation, l'attribut quantitatif V est transformé en un attribut qualitatif $I \in \mathcal{I}$ dont les modalités sont les intervalles construits. **Chaque objet est décrit par**

	$V_{.1}$				$V_{.2}$							$V_{.3}$							
	≥ 3	≥ 1	≥ 0	≤ 3	≤ 0	≤ 4	≤ 12	≤ 16	≥ 0	≥ 4	≥ 12	≥ 16	≤ 15	≤ 16	≤ 17	≤ 18	≥ 15	≥ 16	≥ 17
o_1		×	×	×		×	×	×	×	×			×	×	×	×	×		
o_2			×	×	×	×	×	×	×							×	×	×	×
o_3		×	×	×			×	×	×	×	×			×		×	×	×	
o_4			×	×				×	×	×	×	×			×	×	×	×	×
o_5	×	×	×	×			×	×	×	×	×					×	×	×	×

TABLE 5.2: Contexte ordinal représentant la Table 5.1(i)

la modalité correspondant à l'intervalle contenant la valeur quantitative de V qu'il vérifie.

Notons que l'union des intervalles créés est un intervalle couvrant l'ensemble des valeurs observées dans le domaine de définition de V.

Exemple. Prenons l'exemple simple de la table de données quantitatives présentée dans la table gauche de la Table 5.3. La discrétisation de l'unique attribut quantitatif V_1 correspond au découpage de son domaine de définition $[0, 20]$ en intervalles. Dans notre exemple la discrétisation produit trois intervalles $[0-4[$, $[4, 11[$ et $[11, 20]$. Chacun de ces intervalles devient une modalité de l'attribut qualitatif I_1 construit par discrétisation, comme présenté dans la table droite de la Table 5.3.

O	V_1
o_1	1
o_2	3
o_3	3
o_4	2
o_5	0
o_6	5
o_7	8
o_8	20
o_9	14

→ Discrétisation

O	I_1
o_1	$[0, 4]$
o_2	$[0, 4]$
o_3	$[0, 4]$
o_4	$[0, 4]$
o_5	$[0, 4]$
o_6	$]4, 11]$
o_7	$]4, 11]$
o_8	$]11, 20]$
o_9	$]11, 20]$

→ Binarisation

O	I_1		
	$m_1 = [0, 4]$	$m_2 =]4, 11]$	$m_3 =]11, 20]$
o_1	×		
o_2	×		
o_3	×		
o_4	×		
o_5	×		
o_6		×	
o_7		×	
o_8			×
o_9			×

TABLE 5.3: Discrétisation en intervalles disjoints

Dans la littérature [Catlett 91, Fayyad 93, Dougherty 95, Quinlan 96b, Rabaséda 96, Liu 02, Kotsiantis 06, Dash 11], les méthodes de discrétisation les plus souvent utilisées et étudiées restent les méthodes de discrétisation en intervalles disjoints. Elles y sont généralement présentées selon trois axes : **univariée ou multivariée** (aussi appelée respectivement statique ou dynamique) **; supervisée ou non supervisée ; locale ou globale**. En considérant uniquement la discrétisation en intervalles disjoints, nous nous intéressons à ces différents axes dans les sections suivantes.

Notons que parfois les méthodes ascendantes sont distinguées des méthodes descendantes [Rakotomalala 97, Liu 02]. Les premières consistent à créer tous les intervalles possibles selon les points de coupe trouvés (*i.e.* initialisation des intervalles avec une seule valeur observée par intervalle), puis à fusionner ces intervalles selon un critère de fusion jusqu'à satisfaire **un critère d'arrêt**. Ce type de discrétisation est utilisé dans les méthodes de classification hiérarchique ascendante (CAH) par exemple. Les secondes, telles que présentées ci-dessus, évaluent chaque point de coupe, puis choisissent celui qui offre le meilleur découpage selon **un critère de coupe** et construisent deux intervalles selon ce point de coupe, enfin l'opération est répétée jusqu'à satisfaire **un critère d'arrêt**.

5.2.2 Univariée ou multivariée :

La discrétisation multivariée

Avec ce type de méthodes, les attributs sont découpés en prenant en compte les autres attributs, *i.e.* les attributs sont traités simultanément. Ce type d'approches permet de prendre en compte les corrélations existantes entre attributs. Cependant, se basant sur des calculs itératifs compliqués et très gourmands en ressources machine, ce type de discrétisation est peu utilisé en pratique.

La discrétisation univariée

Avec ce type de méthodes, les attributs sont traités indépendamment les uns des autres. Il s'agit du type de discrétisation le plus répandu car nécessitant beaucoup moins de ressources machine que le type dynamique [Rakotomalala 97, Dougherty 95].

Discussion

Les méthodes de discrétisation univariées étant les plus répandues et utilisées, en particulier pour la construction des arbres de classification, nous nous concentrons sur ces dernières. Pour plus de précisions sur les modèles multivariés, nous renvoyons le lecteur aux travaux suivants [Rakotomalala 97, Hwang 02, Muhlenbach 02, Muhlenbach 05, Kotsiantis 06].

5.2.3 Non supervisée ou supervisée :

La différence entre discrétisation supervisée et discrétisation non supervisée (aussi appelées discrétisation contextuelle et discrétisation non contextuelle) **réside dans l'information utilisée pour mener la discrétisation.**

Dans cette section, les exemples de discrétisation seront appliqués à l'ensemble de données de la Table 5.4. Pour l'unique attribut quantitatif de cet ensemble, le nombre d'objets considérés est 20, la valeur minimale observée est 0 et la valeur maximale est 12. Tels que définis précédemment, les points de coupe possibles sont : $C_V = \{0.5,\ 1.25,\ 1.75,\ 2.25,\ 2.75,\ 3.25,\ 3.75,\ 4.25,\ 4.75,\ 5.25,\ 5.75,\ 6.25,\ 6.75,\ 7.5,\ 8.25,\ 8.75,\ 9.25,\ 9.75,\ 11\}$.

Discrétisation non supervisée

La discrétisation non supervisée est menée exclusivement en utilisant les informations portées par l'attribut analysé. Les classes des objets ne sont pas prises en compte même si ces dernières sont disponibles. **Ce type de discrétisation peut générer beaucoup de points de coupe,** en particulier lorsque les objets sont dispersés sur le domaine de l'attribut. Les méthodes correspondantes sont assez populaires car elles

O	V	K
o_1	0	1
o_2	1	1
o_3	1.5	1
o_4	2	1
o_5	4	1
o_6	5	1
o_7	5.5	1
o_8	6	1
o_9	6.5	1
o_{10}	7	1
o_{11}	8	1
o_{12}	2.5	2
o_{13}	3	2
o_{14}	3.5	2
o_{15}	4.5	2
o_{16}	8.5	2
o_{17}	9	2
o_{18}	9.5	2
o_{19}	10	2
o_{20}	12	2

TABLE 5.4: Ensemble de données à discrétiser

ne nécessitent que peu d'informations. **Cependant, il est nécessaire que l'utilisateur définisse un nombre k d'intervalles à construire.** Notons qu'il existe des formules permettant de définir automatiquement le bon nombre d'intervalles en prenant en compte le nombre d'objets de l'ensemble de données considéré (formules de Brooks-Carruthers, de Huntsberger, de Sturges [Sturges 26], de Scott [Scott 79] ou encore de Freedman-Diaconis [Freedman 81]).

Dans la littérature [Dougherty 95, Rakotomalala 97, Liu 02, Muhlenbach 05, Dash 11], les critères de coupe non supervisés les plus souvent cités sont le découpage en intervalles de largeurs égales (*Equal Width Interval*), le découpage en intervalles d'effectifs égaux (ou de fréquences égales - *Equal Frequency Interval*), ou encore le critère de coupe de distance maximale, ou le critère de contraste monothétique (existant aussi en version supervisée - *Monothetic Contrast Criterion*) [Van de Merckt 93]. Nous les détaillons dans les paragraphes suivants.

Les premiers critères sont simples, le dernier a été développé dans le cadre de l'induction de graphe et pallie à quelques défauts des premiers. Lorsqu'ils sont appliqués pour la discrétisation des données de la Table 5.4, et lorsque 4 intervalles doivent être construits, il en résulte les intervalles présentés sur les Figures 5.2.1, 5.2.2, 5.2.3 et 5.2.4. Sur ces figures, l'axe correspond au domaine de définition de l'attribut quantitatif, les carrés et losanges correspondent aux valeurs observées pour les objets (deux classes possibles losange-bleue ou carré-rouge, non utilisées dans le cas non supervisé) et les points de coupe sont représentés par des triangles noirs coupant l'axe.

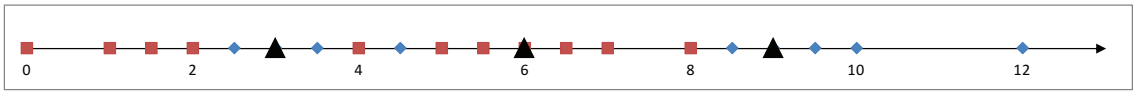


FIGURE 5.2.1: Discrétisation non supervisée - intervalles de largeurs égales

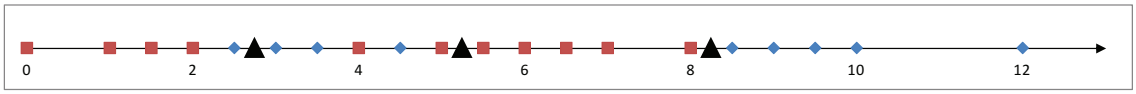


FIGURE 5.2.2: Discrétisation non supervisée - intervalles d'effectifs égaux

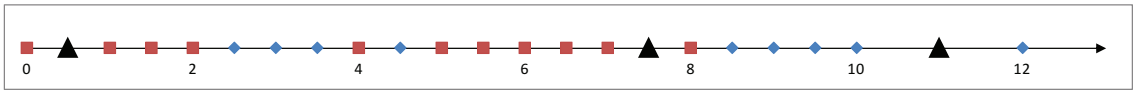


FIGURE 5.2.3: Discrétisation non supervisée - distance maximale

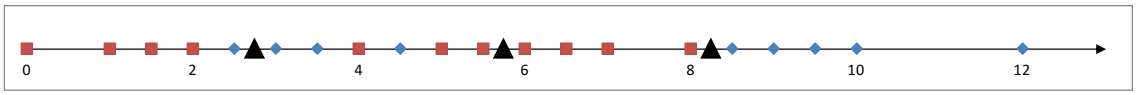


FIGURE 5.2.4: Discrétisation non supervisée - critère de contraste monothétique

Le critère de coupe en intervalles de largeurs égales

Ce critère coupe le domaine de l'attribut quantitatif en k intervalles de **même largeur**. De ce fait, ce critère a sa propre définition des points de coupe C_V (*i.e.* différente de celle mentionnée précédemment) et cette définition ne prend pas en compte de la répartition des objets dans les intervalles construits. Les points de coupe sont calculés en fonction de la valeur minimale min_V et de la valeur maximale max_V observées pour les données considérées, ils sont définis comme suit :

$$c_V^i = min_V + \frac{max_V - min_V}{k} \times i \ \forall i \in \{1, \ldots, k-1\}$$

Ainsi, chaque intervalle construit a une largeur de $\frac{max_V - min_V}{k}$.

Simple à programmer, ce critère est très sensible au nombre d'intervalles k et aux valeurs aberrantes (*outliers*) [Dougherty 95, Muhlenbach 05, Dash 11], celles-ci induisant des intervalles presque vides.

Le critère de coupe en intervalles d'effectifs égaux

Ce critère coupe le domaine de l'attribut quantitatif en k intervalles selon **la fréquence des objets** sur ce domaine. Les points de coupe sont calculés en fonction du nombre d'objets N considérés et des valeurs minimale min_V et maximale max_V observées pour ces données. Chaque intervalle contiendra approximativement $\frac{N}{k}$ valeurs.

Par exemple, selon ce critère, la discrétisation en 4 intervalles de l'attribut quantitatif représenté sur la Figure 5.2.2, génère les intervalles $[0, 2.75]$, $]2.75, 5.25]$, $]5.25, 8.25]$ et $]8.25, 12]$, chacun contenant 5 objets.

La mesure de distance maximale

Cette mesure coupe le domaine de l'attribut quantitatif en k intervalles selon la distance entre les valeurs vérifiées par les objets. Il s'agit de mesurer les distances entre deux valeurs consécutives, et de sélectionner les points de coupe au milieu de deux valeurs consécutives pour lesquelles les distances sont les plus grandes. Cette mesure permet de prendre en compte la répartition des objets sur le domaine de l'attribut quantitatif. Comme le critère de coupe en intervalles réguliers, cette mesure est très sensible aux valeurs aberrantes.

Par exemple, selon ce critère, la discrétisation en 4 intervalles de l'attribut quantitatif représenté sur la Figure 5.2.3, génère les intervalles suivants $[0, 0.5]$, $]0.5, 7.5]$, $]7.5, 11]$ et

$]11, 12]$.

Le critère de contraste monothétique

Ce critère prend en compte la proximité des objets sur le domaine de l'attribut analysé. L'objectif est de respecter au mieux la similarité au sein des groupes d'objets regroupés dans les différents intervalles. Le critère est défini selon un point de coupe $c_V^i \in C_V$, par :

$$mcc(\Omega, c_V^i) = \frac{N_1 \times N_2}{N}(m_{\Omega_1} - m_{\Omega_2})^2$$

Avec :
- Ω_1 et Ω_2 les deux sous-ensembles de données définis à partir de l'ensemble de données initial Ω : $\Omega_1 = \{(o_n, \vec{o_n}) \in \Omega |\ v_n \leq c_V^i\}$ et $\Omega_2 = \{(o_n, \vec{o_n}) \in \Omega |\ v_n > c_V^i\}$;
- $N_i = |\Omega_i|$ le nombre d'objets appartenant à chaque sous-ensemble ($i \in \{1, 2\}$) ;
- m_{Ω_i} la moyenne des valeurs observées pour les objets de l'ensemble Ω_i ($i \in \{1, 2\}$).

Ce critère correspond à l'inertie inter-groupes monovariée [Rakotomalala 97]. **Les intervalles sont construits en minimisant la distance intra-groupe** (distance entre les objets de chaque Ω_i) **et en maximisant la distance entre les centres de gravité des groupes**.

Pour ce type de critère, les points de coupe c_V^i utilisés parmi les points de coupe possibles sont ceux qui maximisent le critère $mcc(\Omega, c_V^i)$. Ainsi pour discrétiser un attribut $V \in \mathcal{V}$ en considérant un ensemble de données Ω, **le meilleur point de coupe** c_V^* est celui qui maximise le critère, c'est à dire :

$$c_V^*(\Omega) = argmax_{c_V^i \in C_V}(mcc(\Omega, c_V^i)) \tag{5.2.1}$$

Par exemple, selon ce critère, la discrétisation en 4 intervalles de l'attribut quantitatif représenté sur la Figure 5.2.4, génère les intervalles suivants $[0, 2.75]$, $]2.75, 5.75]$, $]5.75, 8.25]$ et $]8.25, 12]$.

Ce critère a des propriétés intéressantes (prise en compte de la répartition des valeurs sur le domaine de l'attribut, faible sensibilité aux valeurs aberrantes, ...), mais, comme tout critère de discrétisation non supervisé, il peut parfois engendrer des découpages non pertinents pour la classification [Rakotomalala 97, Celeux 93].

Le paramétrage des méthodes de discrétisation non supervisées est difficile, et tout particulièrement déterminer le bon nombre d'intervalles à construire reste complexe. Bien que simple à comprendre, ces méthodes ne sont pas adaptées au cadre supervisé dans lequel nous nous trouvons. La Figure 5.2.5 illustre la différence entre une discrétisation non supervisée et une discrétisation supervisée menées sur le même ensemble de données étiquetées. Les points triangulaires et carrés appartiennent à ceux classes différentes. Cet exemple simple illustre bien l'apport de l'utilisation d'un critère prenant en compte la classe des objets par rapport à un critère ne la prenant pas en compte, en particulier

lorsqu'un modèle de classification doit être construit. En effet, **un critère supervisé permet de sélectionner seulement le(s) point(s) de coupe séparant les objets de classes différentes.**

FIGURE 5.2.5: Discrétisation non supervisée (gauche) & discrétisation supervisée (droite)

Différentes études ont mis en avant les meilleurs résultats obtenus, entre autre en classification supervisée, lorsqu'une discrétisation supervisée est utilisée [Catlett 91, Dougherty 95, Liu 02, Muhlenbach 05, Dash 11].

Discrétisation supervisée

La discrétisation supervisée est menée sur un ensemble de données étiquetées. Dans ce cas, la classe de chaque objet est prise en compte lors du choix des points de coupe. Ainsi, pour un attribut particulier, un point de coupe séparant des objets de classes différentes sera préféré à un point de coupe séparant des objets de même classe. **Lorsque les données à traiter sont étiquetées, les méthodes supervisées sont privilégiées car la plupart peuvent déterminer automatiquement le nombre d'intervalles et les meilleurs points de coupe** [Dougherty 95, Muhlenbach 05]. Cela évite tout d'abord un éparpillement des objets de même classe dans un trop grand nombre d'intervalles différents, mais aussi la présence d'objets de classes différentes dans un même intervalle.

Il existe **de nombreux critères de coupe** permettant de choisir, selon la classe des objets, les meilleurs points de coupe. Nous pouvons citer, entre autres, les critères de division utilisés pour la construction des arbres, *i.e.* **les critères de Gini, d'Entropie, du χ^2, mais aussi le critère de contraste monothètique dans sa version supervisée** [Van de Merckt 93] ou encore **le critère** très connu défini par Fayyad et Irani [Fayyad 93], **MDLPC** (*Minimum Description Length Principle Cut*) basé sur la mesure d'entropie. Mentionnons aussi le critère de Hotelling [Hotelling 33a, Hotelling 33b] utilisé dans la méthode Navigala [Guillas 07].

Les Figures 5.2.6 et 5.2.7 présentent deux discrétisations supervisées obtenues respectivement avec le critère de Gini et le critère MDLPC pour l'ensemble de données de la Table 5.4, les carrés représentant la classe $K = 1$ et les triangles la classe $K = 2$.

Les critères de discrétisation MDLPC et MCC supervisé sont présentés dans de nombreuses références [Dougherty 95, Rabaséda 96, Fayyad 96, Rakotomalala 97, Dash 11]. Pour notre part, nous nous intéressons particulièrement aux critères mis en œuvre dans les arbres de classification (critère de Gini, d'entropie, du χ^2) et au critère utilisé dans

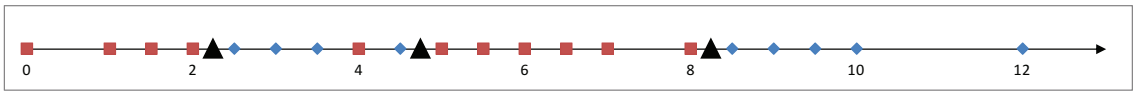


FIGURE 5.2.6: Discrétisation supervisée - critère de Gini

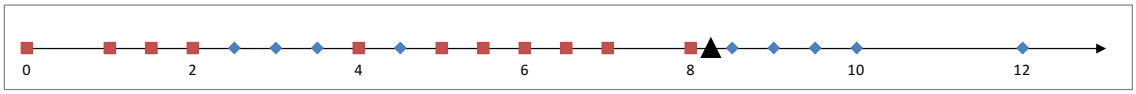


FIGURE 5.2.7: Discrétisation supervisée - MDLPC

Navigala (critère de Hotelling). Pour plus de précisions sur les modèles MDLPC et MCC supervisé, nous renvoyons donc le lecteur aux références précédentes.

De manière générale, la définition de ces critères supervisés repose, comme pour la définition des critères de division des arbres de classification (chapitre 2), **sur la définition d'une table de contingence**. Ainsi pour chaque point de coupe $c_V \in C_V$, une table de contingence est construite en croisant l'attribut qualitatif I (en colonnes) à la classe $K = \{q_1, q_2, \ldots, q_H\}$ des N objets de l'ensemble de données considéré $\Omega = (O, \mathcal{Y}, K)$ (en lignes). I est l'attribut qualitatif associé à c_V, *i.e.* l'attribut ayant pour modalités les deux intervalles $[min_V, c_V]$ et $]c_V, max_V]$. La table de contingence, présentée dans la Table 5.5, est définie comme suit :

- Les cases internes de la table contiennent les effectifs croisant une classe et l'un des deux intervalles ; n_{h1} (resp. n_{h2}) est le nombre d'objets de O vérifiant l'intervalle $[min_V, c_V]$ (resp. $]c_V, max_V]$) et appartenant à la classe q_h ;
- La dernière ligne de la table contient la somme des effectifs colonne par colonne ; $n_{.1}$ (resp. $n_{.2}$) est le nombre d'objets de O vérifiant l'intervalle $[min_V, c_V]$ (resp. $]c_V, max_V]$) ;
- La dernière colonne de la table contient la somme des effectifs ligne par ligne ; $n_{h.}$ est le nombre d'objets de O appartenant à la classe q_h ;

Afin de simplifier les notations, rappelons que nous utilisons les estimations de probabilités suivantes :

- $p_{h.} = \hat{Pr}(K = q_h) = \frac{n_{h.}}{N}$, c'est la fréquence de la classe q_h sur l'échantillon Ω ;
- $p_{.1} = \hat{Pr}(I = [min_V, c_V]) = \frac{n_{.1}}{N}$, c'est la fréquence de l'intervalle $[min_V, c_V]$ sur l'échantillon Ω ; $p_{.2}$ est définie de manière similaire pour $I =]c_V, max_V]$;
- $p_{h1} = \hat{Pr}(K = q_h / I = [min_V, c_V]) = \frac{n_{h1}}{n_{.1}}$, c'est la probabilité conditionnelle de la classe q_h par rapport à l'intervalle $[min_V, c_V]$ sur l'échantillon Ω ; p_{h2} est définie de manière similaire pour $I =]c_V, max_V]$.

Avant de présenter l'utilisation de ces critères de manière générique, nous présentons le critère de Hotelling, les autres critères auxquels nous nous intéressons (Gini, Entropie et

K	I		$\sum$
	$[min_V, c_V]$	$]c_V, max_V]$	
q_1	n_{11}	n_{12}	$n_{1.}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
q_h	n_{h1}	n_{h2}	$n_{h.}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
q_H	n_{H1}	n_{H2}	$n_{H.}$
$\sum$	$n_{.1}$	$n_{.2}$	N

TABLE 5.5: Table de contingence associée au point de coupe c_V

χ^2) ayant déjà été présentés dans la section 2.3.1. Notons que le découpage de l'attribut V en deux intervalles simplifie les formules de ces critères (*i.e.* seulement deux colonnes dans la table de contingence, *cf.* Annexe C pour les formules simplifiées).

Le critère de Hotelling prend en compte la distribution des objets sur le domaine de l'attribut à discrétiser V ainsi que leur répartition en classes. Il est défini comme suit :

$$gain_{Hotelling}(V, c_V^i, \Omega) = H(V) - (p_{.1}H(V_1) + p_{.2}H(V_2)) \tag{5.2.2}$$

Avec $V_1 = \{v_n \in V |\ v_n \leq c_V \ et\ o_n \in O\}$ et $V_2 = \{v_n \in V |\ v_n > c_V \ et\ o_n \in O\}$.

Et, soit g_h le centre de gravité de la classe q_h, g le centre de gravité de l'attribut V, v_{h_j} le j-ième élément de la classe q_h et $n_{h.}$ le nombre d'éléments de la classe q_h :

avec :

$$H(V) = \frac{VarInter(V)}{VarIntra(V)}$$

$$avec\ :\quad VarInter(V) = \sum_{h=1}^{H} p_{h.}(g_h - g)^2$$

$$VarIntra(V) = \sum_{h=1}^{H} p_{h.}(\sum_{j=1}^{n_{h.}}(v_{h_j} - g_h)^2)$$

Dans sa forme, ce critère est proche du critère de contraste monothètique puisqu'**il permet de construire des intervalles en minimisant la variance intra-classe et en maximisant la variance inter-classe**s.

Dans la suite du manuscrit, nous adoptons **la notation générique** *gain* **pour désigner ces différents critères de coupe**. Ainsi $gain(V, c_V^i, \Omega)$ peut être le critère du χ^2, de Gini, d'Entropie, ou de Hotelling, calculé sur l'attribut V au point de coupe candidat c_V^i.

Ces critères permettent donc d'évaluer la qualité d'un découpage, *i.e.* ils permettent de comparer les différents points de coupe possibles de C_V pour déterminer ceux qui engendrent la meilleure séparation des objets de classes différentes. **Un point de coupe**

est considéré comme le meilleur point de coupe c_V^* parmi les points de coupe possibles, pour l'attribut $V \in \mathcal{V}$ et sur l'ensemble de données Ω, s'il vérifie l'équation suivante :

$$c_V^*(\Omega) = argmax_{c_V^i \in C_V}(gain(V, c_V^i, \Omega)) \tag{5.2.3}$$

Lors de la discrétisation d'un attribut quantitatif, plusieurs points de coupe sont généralement sélectionnés, *i.e.* les points de coupe donnant les critères de coupe les plus forts, sont les points sélectionnés. **La discrétisation se fait de manière itérative, à chaque étape le meilleur point de coupe est sélectionné dans C_V selon l'équation 5.2.3, il est « retiré » de C_V et le processus est réitéré jusqu'à satisfaire un critère d'arrêt.**

Il existe différents **critères d'arrêt** dans la littérature, parmi les plus simples se trouve la définition d'un nombre maximum d'intervalles à construire, critère difficilement paramétrable et non supervisé. Parmi les plus élaborés se trouvent **les seuils définis pour les critères de coupe tels que le critère de Gini, d'Entropie ou du χ^2**, qui ont déjà été présentés dans la section 2.3.2. Comme les critères de Gini, d'Entropie ou du χ^2, le critère de Hotelling s'annule lorsque les objets de classes différentes sont séparés ou lorsque qu'une coupe n'est pas pertinente et en particulier lorsque les données ne sont pas séparables (*i.e.* des objets de classes différentes ont la même description). La discrétisation d'un attribut s'arrête donc lorsque pour tout point de coupe, le critère est nul ou inférieur à un seuil. L'utilisation d'un seuil non nul pour rejeter une coupe permet de limiter le nombre d'intervalles construits et aussi d'éviter le sur-apprentissage.

Dans le cadre des travaux présentés dans ce manuscrit, les données analysées sont étiquetées, c'est pourquoi nous privilégions les méthodes de discrétisation supervisées. Le choix du critère de coupe étant généralement associé au modèle utilisé et aux ensembles de données analysés, nous testerons plusieurs critères pour déterminer celui qui donne les meilleurs résultats en moyenne sur les ensembles de données analysés. Le critère d'arrêt sera lui aussi défini par rapport au modèle mis en œuvre.

5.2.4 Locale ou globale :

Le troisième axe de distinction entre les méthodes de discrétisation correspond à séparer les méthodes dites globales, des méthodes dites locales.

Discrétisation globale

La discrétisation globale considère à chaque étape l'ensemble de données complet, tant du point de vue des observations que des attributs. Ainsi lorsque plusieurs points de coupe doivent être trouvés pour séparer les objets d'un ensemble de données,

ils sont recherchés pour tous les attributs $V_{.i} \in \mathcal{V}$, parmi tous les points de coupe $C_{V_{.i}}$ définis pour chaque $V_{.i}$ et en considérant l'ensemble complet des objets O. Ainsi chaque étape de discrétisation définit l'association du meilleur attribut et de son meilleur point de coupe. **Ce type de discrétisation a lieu en amont de la construction du modèle de classification**, *i.e.* en pré-traitement. **La discrétisation obtenue est donc indépendante de l'agencement des données dans le modèle construit**. Notons cependant que de ce type de discrétisation peut être stoppée en observant le modèle construit, *i.e.* à chaque étape de discrétisation le modèle est construit à partir des données discrétisées, et s'il satisfait à certaines contraintes (taux d'erreur en resubstitution par exemple) alors la discrétisation est stoppée, sinon une étape supplémentaire de discrétisation est réalisée.

Discrétisation locale

La discrétisation locale, quant à elle, **considère à chaque étape seulement un sous-ensemble d'objets et un sous-ensemble d'attributs**. Généralement, **ces sous-ensembles sont déterminés par le modèle en cours de construction**. Ainsi, pour une étape de discrétisation, la recherche du meilleur attribut et du meilleur point de coupe associé se fait en parcourant uniquement les sous-ensembles d'objets et d'attributs considérés localement par le modèle. Les objets et valeurs n'appartenant pas à ces sous-ensembles ne sont pas considérés.

Par exemple, au cours de la construction d'un arbre de classification, pour un nœud donné (différent de la racine), la discrétisation d'un attribut quantitatif sera réalisée uniquement pour les objets contenus dans ce nœud. Seuls les points de coupe appartenant au domaine de l'attribut redéfini localement, *i.e.* domaine restreint aux valeurs vérifiées par les objets de ce nœud, sont considérés.

Ce type de discrétisation permet de prendre en compte la structure du modèle construit et les coupes sont réalisées uniquement si elles sont nécessaires au modèle.

Discussion

Selon si la discrétisation choisie est locale ou globale, les intervalles obtenus ne sont pas les mêmes. La discrétisation globale peut parfois générer des intervalles qui ne sont pas utiles lors de la construction du modèle de classification, puisqu'elle est réalisée en amont de la construction du modèle.

Exemple. Soit deux attributs quantitatifs $V_{.1}$, $V_{.2} \in \mathcal{V}$, et leur représentation sur le plan 2D ($V_{.1}$,$V_{.2}$). La Figure 5.2.8 présente une discrétisation réalisée globalement (Figure de gauche) et une discrétisation réalisée localement (Figure de droite) ; chaque segment noir horizontal ou vertical représente un point de coupe. La discrétisation locale permet de

définir des points de coupe pour un espace précis, tandis que la discrétisation globale définit des points de coupe appliqués à l'espace complet. Par exemple, le point de coupe $V_2 = 12$, n'est considéré que pour le sous-ensemble d'objets vérifiant $V_1 > 5$.

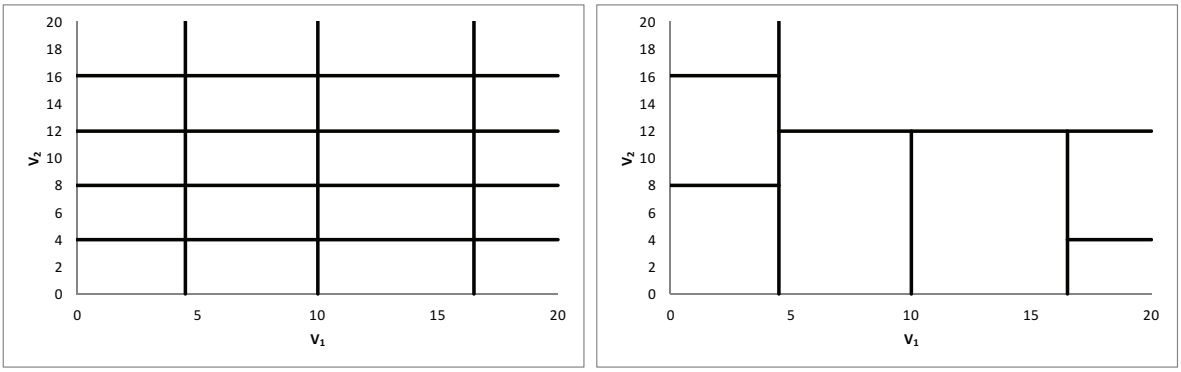


FIGURE 5.2.8: Discrétisation globale (gauche) & discrétisation locale (droite)

Dans le cadre des arbres de classification, l'opposition entre ces deux types de discrétisation a nourri de nombreuses études. Nous nous intéresserons plus précisément à ces études dans la section 5.2.6. Notons cependant qu'il est difficile d'arriver à une quelconque conclusion sur ce point, les paramètres à prendre en compte dépendent à la fois la composition de l'ensemble de données analysé mais aussi la structure du modèle construit.

Dans nos expérimentations (section 5.3.3), nous étudierons les différents impacts de la discrétisation locale par rapport à la discrétisation globale sur notre modèle de classification.

5.2.5 Algorithmes de discrétisation

En considérant un ensemble d'objets donné Ω, la discrétisation d'un attribut quantitatif $V \in \mathcal{V}$ suit un processus simple en plusieurs étapes :

1. Ordonner les valeurs observées pour un ensemble d'objets donné Ω ;
2. Calculer les points de coupe possibles C_V ;
3. Évaluer chaque point de coupe $c_V^i \in C_V$ selon le critère de coupe *gain* ;
4. Si un point de coupe c_V^* satisfait le critère de coupe, couper l'attribut en intervalles $[min_V, c_V^*]$, $]c_V^*, max_V]$;
5. Si le critère d'arrêt, que nous notons $\mathcal{S}$, n'est pas satisfait, répéter à partir du point 3 ; sinon arrêter et renvoyer l'attribut discrétisé I ayant pour modalités les intervalles construits.

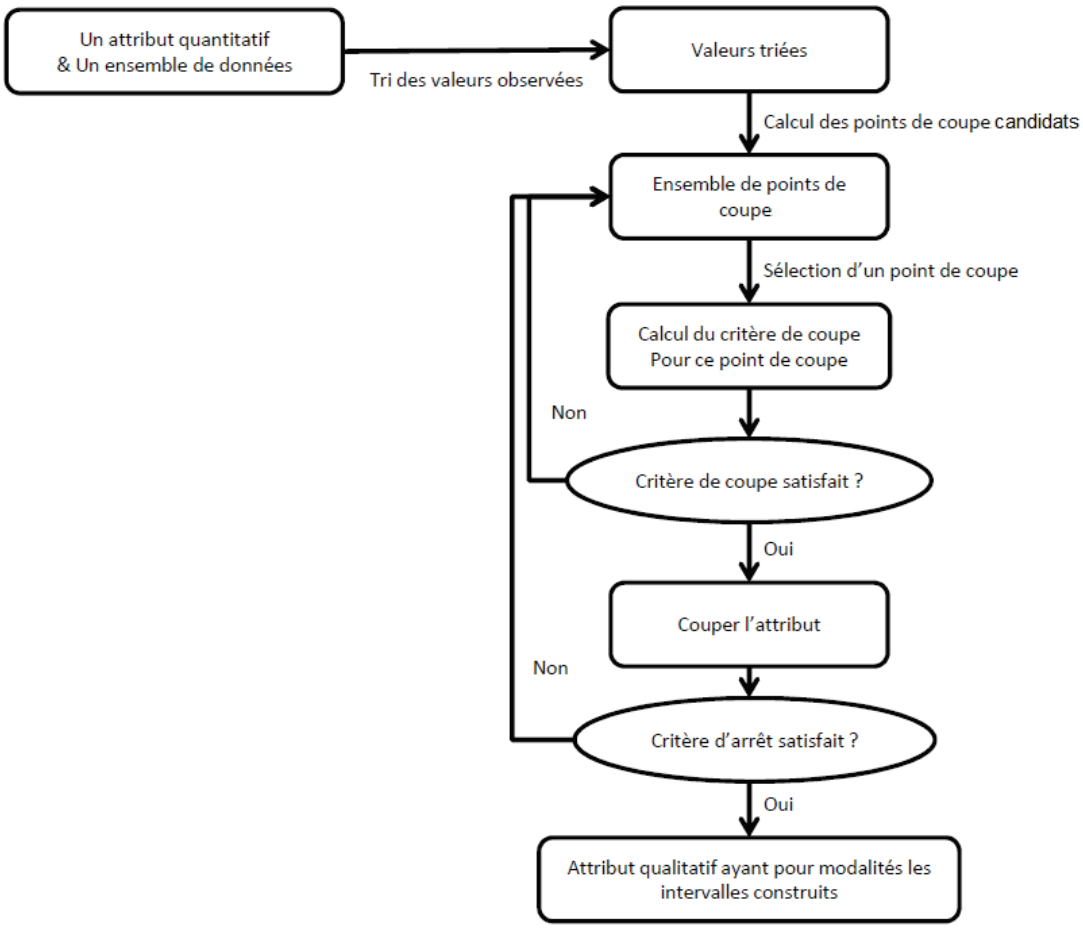


FIGURE 5.2.9: Processus de discrétisation univariée [Liu 02]

La Figure 5.2.9 présente cette suite d'opérations.

La discrétisation repose donc principalement sur les deux critères de coupe et d'arrêt.

Lorsque l'ensemble de données contient plusieurs attributs explicatifs quantitatifs $\mathcal{V} = \{V_{.1}, \ldots, V_{.H}\}$, la discrétisation est menée sur tous ces attributs. **L'étape 4 comporte alors deux sous-étapes** présentées dans l'algorithme 5.1. Tout d'abord, **pour chaque attribut $V_{.h} \in \mathcal{V}$, le meilleur point de coupe associé $c^*_{V_{.h}} \in C_{V_{.h}}$ est calculé** selon l'équation suivante :

$$c^*_{V_{.h}}(\Omega) = argmax_{c^i_{V_{.h}} \in C_{V_{.h}}}(gain(V_{.h}, c^i_{V_{.h}}, \Omega)) \tag{5.2.4}$$

Puis, **le meilleur attribut V^* parmi l'ensemble d'attributs $\mathcal{V}$ est défini** selon l'équation suivante :

$$V^*(\Omega) = argmax_{V_{.h} \in \mathcal{V}}(gain(V_{.h}, c^*_{V_{.h}}(\Omega), \Omega)) \tag{5.2.5}$$

Le critère d'arrêt permet de stopper la discrétisation. L'étape 5 repose sur ce critère, selon si les objets de classes différentes sont suffisamment séparés ou non et s'il reste des découpages pertinents, une étape de discrétisation est relancée ou le processus se termine.

Une fois le processus de discrétisation réalisé, l'ensemble de données quantitatives est transformé en un ensemble de données qualitatives $\Omega' = (O, \mathcal{I}, K)$, la table associée est appelée **table discrétisée**.

Algorithme 5.1 Discrétiser(Ω,*gain*,$\mathcal{S}$)

Entrées :
- $\Omega = (O, \mathcal{V}, K)$ l'ensemble de données quantitatives étiquetées, avec $\mathcal{V} = \{V_{.h}, \forall h \in \{1, \ldots, H\}\}$;
- *gain* le critère de coupe ;
- $\mathcal{S}$ le critère d'arrêt ;

Sorties :

Ω' la table discrétisée (ensemble de données qualitatives étiquetées) ;

DÉBUT

//Pour chaque attribut $V_{.h}$, soit $(v_1, \ldots, v_N)$ ses valeurs observées triées.

//ETAPE 1

Initialiser Ω' avec :
- sur chaque ligne : un objet $o_n \in O$;
- sur chaque colonne : un attribut qualitatif $I_{.h}$;
- dans chaque case de la table, l'unique intervalle $[min_{V_{.h}}, max_{V_{.h}}] = [v_1, v_N]$ contenant toutes les valeurs observées pour l'attribut $V_{.h}$ de la colonne ;

//ETAPE 2

pour chaque $V_{.h} \in \mathcal{V}$ **faire**

$C_{V_{.h}} \leftarrow \{c^1_{V_{.h}}, \ldots, c^{N-1}_{V_{.h}}\}$ avec $c^i_{V_{.h}} = \frac{v_i + v_{i+1}}{2}$;

fin pour

tant que $\mathcal{S}$ n'est pas vérifié **faire**

//ETAPE 3-1

pour chaque $V_{.h} \in \mathcal{V}$ **faire**

Calculer $c^*_{V^*} \in C_{V_{.h}}$ selon *gain* ; //selon équation 5.2.4

fin pour

//ETAPE 3-2

Calculer selon *gain*, le meilleur attribut V^* parmi les $V_{.h} \in \mathcal{V}$ associé à son meilleur point de coupe $c^*_{V^*}$; //selon équation 5.2.5

//ETAPE 4

Dans Ω', couper I^* (associé à V^*) en deux intervalles disjoints selon $c^*_{V^*}$;

Dans Ω', mettre à jour les relations entre chaque o_n et les intervalles créés ;

Supprimer $c^*_{V^*}$ de C_{V^*} ;

fin tant que//ETAPE 5 - bouclage

renvoyer Ω' ;

FIN

La Table 5.7 illustre le déroulement de l'Algorithme 5.1 appliqué à l'ensemble de données de la Table 5.6 avec pour critère de gain et d'arrêt le critère du χ^2. Le découpage, étape par étape, de l'ensemble de données initial est présenté dans cette Table 5.7.

Cette discrétisation a été réalisée de manière univariée, globale et supervisée. La Figure 5.2.10 représente la projection de l'ensemble de données dans l'espace 2D $(V_{.1}, V_{.2})$ ainsi que les différents axes de découpage des attributs $V_{.1}$ et $V_{.2}$ obtenus en traits noirs. Cette Figure permet d'illustrer que la discrétisation a permis de séparer les objets de classes différentes dans des espaces différents.

O	$\mathcal{V}$			K
	$V_{.1}$ [0,20]	$V_{.2}$ [0,20]	$V_{.3}$ [15,18]	
o_1	1	4	15	1
o_2	0	0	18	
o_3	1	12	16	2
o_4	0	16	17	
o_5	3	12	18	
o_6	8	16	15	3
o_7	6	20	17	
o_8	15	12	16	
o_9	18	4	17	4
o_{10}	20	12	18	

Table 5.6: Table de données quantitatives étiquetées - Ω

Après cette introduction générale de la discrétisation, intéressons nous à son utilisation par les deux modèles de classification supervisée que nous étudions à savoir les arbres de classification et les treillis de Galois.

5.2.6 Discrétisation et modèles de classification arborescents

5.2.6.1 Discrétisation et arbres de classification

En général, les algorithmes de construction d'arbres de classification (CART, C4.5, ChAID) utilisent une discrétisation locale supervisée, leur critère de coupe correspond alors à leur critère de division (*cf.* section 2.5).

Pour chaque nœud courant $\mathcal{N}$ ne satisfaisant pas le critère d'arrêt, le meilleur attribut $V^* \in \mathcal{V}_\mathcal{N}$ et son meilleur point de coupe $c^*_{V^*}$ sont recherchés pour séparer les objets $O_\mathcal{N}$ contenus dans ce nœud. Deux nœuds fils sont ensuite construits, l'un avec les objets de $O_\mathcal{N}$ vérifiant l'intervalle $[min_{V^*}, c^*_{V^*}]$ et l'autre avec les objets de $O_\mathcal{N}$ vérifiant l'intervalle $]c^*_{V^*}, max_{V^*}]$. Puis comme lors de la construction d'un arbre à partir de données qualitatives, le processus est réitéré sur chaque nœud construit jusqu'à satisfaction d'un critère d'arrêt (*cf.* section 2.3).

O	$\mathcal{V}$				K
	$V_{.1}$		$V_{.2}$	$V_{.3}$	
	[0,4.5]	]4.5,20]	[0,20]	[15,18]	
o_1	1	-	4	15	1
o_2	0	-	0	18	
o_3	1	-	12	16	2
o_4	0	-	16	17	
o_5	3	-	12	18	
o_6	-	8	16	15	3
o_7	-	6	20	17	
o_8	-	15	12	16	
o_9	-	18	4	17	4
o_{10}	-	20	12	18	

(i) Étape 1 - Ω

O	$\mathcal{V}$					K
	$V_{.1}$		$V_{.2}$		$V_{.3}$	
	[0,4.5]	]4.5,20]	[0,8]	]8,20]	[15,18]	
o_1	1	-	4	-	15	1
o_2	0	-	0	-	18	
o_3	1	-	-	12	16	2
o_4	0	-	-	16	17	
o_5	3	-	-	12	18	
o_6	-	8	-	16	15	3
o_7	-	6	-	20	17	
o_8	-	15	-	12	16	
o_9	-	18	4	-	17	4
o_{10}	-	20	-	12	18	

(ii) Étape 2 - Ω

O	$\mathcal{V}$						K
	$V_{.1}$			$V_{.2}$		$V_{.3}$	
	[0,4.5]	]4.5,16.5]	]16.5,20]	[0,8]	]8,20]	[15,18]	
o_1	1	-	-	4	-	15	1
o_2	0	-	-	0	-	18	
o_3	1	-	-	-	12	16	2
o_4	0	-	-	-	16	17	
o_5	3	-	-	-	12	18	
o_6	-	8	-	-	16	15	3
o_7	-	6	-	-	20	17	
o_8	-	15	-	-	12	16	
o_9	-	-	18	4	-	17	4
o_{10}	-	-	20	-	12	18	

(iii) Étape 3 - Ω

O	$\mathcal{I}$			K
	$I_{.1}$	$I_{.2}$	$I_{.3}$	
o_1	[0,4.5]	[0,8]	[15,18]	1
o_2	[0,4.5]	[0,8]	[15,18]	
o_3	[0,4.5]	]8,20]	[15,18]	2
o_4	[0,4.5]	]8,20]	[15,18]	
o_5	[0,4.5]	]8,20]	[15,18]	
o_6	]4.5,16.5]	]8,20]	[15,18]	3
o_7	]4.5,16.5]	]8,20]	[15,18]	
o_8	]4.5,16.5]	]8,20]	[15,18]	
o_9	]16.5,20]	[0,8]	[15,18]	4
o_{10}	]16.5,20]	]8,20]	[15,18]	

(iv) Table discrétisée - Ω'

Table 5.7: Étapes de discrétisation de la table de données quantitatives 5.6

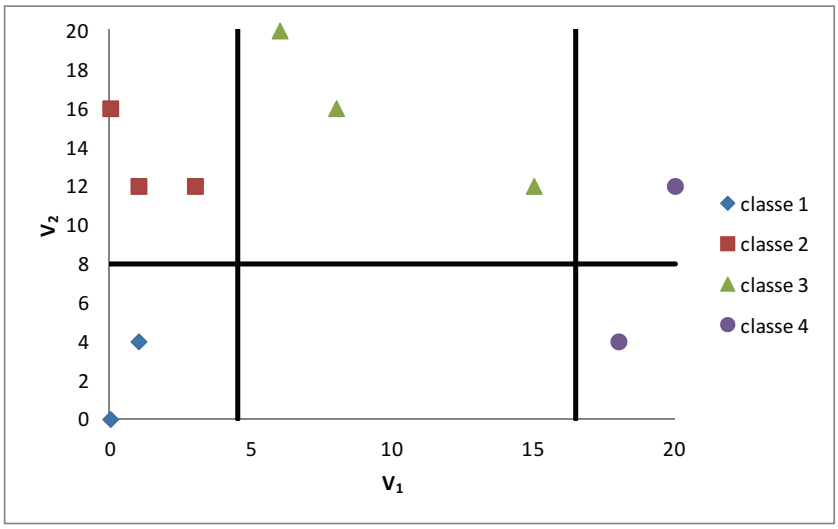


FIGURE 5.2.10: Représentation de la discrétisation (Table 5.7(iv)) dans le plan 2D ($V_{.1}$,$V_{.2}$)

Les découpages sont donc indépendants d'une branche à l'autre. La sélection des points de coupe est réalisée localement, *i.e.* seul le sous-ensemble d'objets du nœud compte pour le choix du meilleur attribut et du meilleur point de coupe.

Suite à la construction de l'arbre, les tests associés aux branches ne sont plus des égalités sur modalités d'un attribut qualitatif mais des tests d'appartenance d'une valeur à un intervalle.

Exemple. La Figure 5.2.11 présente l'arbre construit selon l'algorithme C4.5, à partir de l'ensemble de données de la Table 5.6. On peut remarquer sur cette Figure que le point de coupe 16.5 est sélectionné pour l'attribut $V_{.1}$ uniquement pour séparer les objets o_6 à o_{10}, il n'est pas utilisé pour séparer les objets o_1 à o_5. Ces derniers sont séparés par le point de coupe 8 associé à l'attribut $V_{.2}$, lui aussi utilisé uniquement pour séparer ces cinq objets.

Sur cette Figure, le chemin en tirets gris présente la classification par navigation de l'objet o_{11}. Chaque étape de navigation correspond au test vérifiant dans quel intervalle test est inclue la valeur vérifiée par o_{11}. Selon la classification, o_{11} est classé dans la classe 1.

L'impact du type de discrétisation utilisé, sur la structure de l'arbre et sur ses performances en classification, a été étudié dans de nombreux articles [Catlett 91, Dougherty 95, Quinlan 96b, Ho 97, Rakotomalala 97, Liu 02, Boland 07].

De manière générale, il en ressort que **pour les arbres de classification, il est d'usage de privilégier une discrétisation univariée, supervisée et locale**.

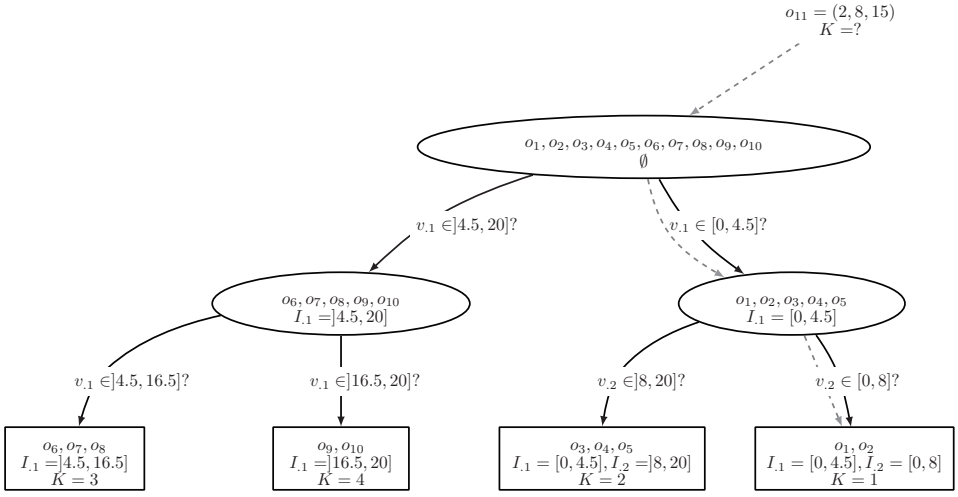


FIGURE 5.2.11: Arbre de classification C4.5 construit à partir de la Table 5.6

Cependant, il faut noter que les études opposant discrétisation locale et discrétisation globale mettent en avant divers éléments de comparaison.

Dans [Dougherty 95], Dougherty *et al.* indiquent que la discrétisation locale (utilisée dans C4.5, critère de gain ratio) **engendre des arbres plus complexes que la discrétisation globale** (basée sur le critère d'Entropie) **sans pour autant améliorer les performances en classification.**

Dans [Catlett 91], Catlett indique que l'étape d'apprentissage de l'arbre nécessite moins de temps de calcul lorsqu'une discrétisation globale est utilisée.

Dans [Quinlan 96b], Quinlan répond, entre autre à l'article de Dougherthy *et al.*, **que la discrétisation locale améliore les performances en classification** des arbres de classification en particulier lorsque l'ensemble de données étudié est grand. Il base ses propos sur une modification du critère de coupe utilisé et sur différentes expérimentations.

Comme précisé par Rakotomalala [Rakotomalala 97], il est difficile de clore ce débat tant les paramètres d'évaluation sont divers.

Ainsi, pour la construction des arbres de classification, il est possible d'utiliser différents types de discrétisation. Nous allons voir que pour les treillis de Galois, jusqu'alors, l'usage était d'utiliser une discrétisation globale.

5.2.6.2 Discrétisation et treillis de Galois

Les treillis de Galois sont définis à partir d'un contexte, *i.e.* une table de données binaires. Ainsi, lorsqu'un ensemble de données quantitatives doit être analysé, le type de

ces données doit être transformé avant toute construction de treillis. Il est donc naturel d'appliquer une discrétisation globale à l'ensemble de données quantitatives.

Discrétisation en intervalles non disjoints

Comme mentionné précédemment, dans le cadre de l'analyse formelle de concepts, une discrétisation en intervalles non disjoints a été proposée [Ganter 99]. **Dans la pratique, le contexte obtenu est plus complexe** (*i.e.* plus d'attributs binaires et d'intersections entre attributs) **que celui obtenu avec une discrétisation en intervalles disjoints, de ce fait cette discrétisation est plus rarement utilisée.**

En effet, de par sa définition, dans le cas non supervisé l'échelonnage génère dans le contexte binaire construit, au moins une colonne par valeur observée pour chaque attribut quantitatif. Rappelons que pour le traitement des données quantitatives continues, l'échelonnage inter-ordinal est généralement utilisé [Ganter 99, Kaytoue 11], ce dernier induit alors deux colonnes par valeur observée et pour chaque attribut. Sur l'exemple précédent (Table 5.2), on peut remarquer que malgré une table initiale très simple, de nombreuses informations sont redondantes dans le contexte obtenu, par exemple $V_{\cdot 1} \leq 1 \Rightarrow V_{\cdot 1} \leq 3$.

Avec ce processus, le contexte construit contient $\prod_{h \in \{1,\ldots,H\}} \frac{|V_{\cdot h}| \times (|V_{\cdot h}|+1)}{2}$ **intervalles**, où $|V_{\cdot h}|$ est le nombre de valeurs observées pour l'attribut $V_{\cdot h} \in \mathcal{V}$.

Notons que ce phénomène est lié à **l'absence de supervision** qui permettrait d'évaluer la pertinence de chaque valeur observée dans le découpage du domaine de l'attribut quantitatif. De plus, la redondance des informations peut, en partie, être effacée par une réduction sur les attributs binaires du contexte construit (*cf.* section 3.2.3).

Kaytoue *et al.* [Kaytoue 09a] ont démontré que la construction d'un treillis à partir d'un tel contexte, même issu de l'échelonnage inter-ordinal d'un ensemble de données quantitatives peu volumineux ($|O| = 20000$, $|\mathcal{V}| = 10$, et $|V_{\cdot h}| \cong 100$), **est très difficile**. De plus, il a déjà été démontré [Ganter 99, Kaytoue 09a] que parmi tous les intervalles possibles, peu sont intéressants. C'est le cas, par exemple, de l'intervalle $[0, 3]$ qui couvre tous les objets. Ce dernier ne séparant pas les objets de classes différentes, il ne sera pas utilisé dans un processus de classification.

Pour limiter ce phénomène, dans ses travaux Kaytoue [Kaytoue 09b, Kaytoue 09a, Kaytoue 11] définit une structure de patron d'intervalles, s'appuyant sur la notion de structure de patron définie par Ganter et Kuznetsov [Ganter 01]. Ce développement, dans un cadre non supervisé, permet de diminuer le nombre d'intervalles générés.

Ce type de discrétisation a principalement été utilisé dans un cadre non supervisé. Or, la discrétisation supervisée permet d'améliorer les performances

en classification des arbres de classification comparée à une discrétisation non supervisée. Considérant ce point, ainsi que les liens forts existants entre arbres de classification et treillis de Galois, mais aussi la propriété des treillis lorsqu'ils sont définis à partir d'une table discrétisée en intervalles disjoints (*cf.* section suivante), **nous nous intéressons plus particulièrement à la discrétisation supervisée en intervalles disjoints**.

Discrétisation en intervalles disjoints

Ainsi, lorsqu'un ensemble de données quantitatives étiquetées doit être analysé, une discrétisation globale supervisée est généralement utilisée avant de construire le treillis de Galois associé à la table discrétisée. Cette discrétisation est menée en utilisant des critères de coupe tels que le critère de Gini, d'Entropie, du χ^2 ou de Hotelling. Les opérations de coupe, telles que présentées en section 5.2, sont répétées tant qu'il existe des coupes pertinentes ou jusqu'à ce que les objets de classes différentes soient représentés par des attributs qualitatifs différents, *i.e.* des intervalles différents.

La définition d'un treillis de Galois à partir d'une table discrétisée, induit tout d'abord des modifications simples dans son utilisation pour la classification de nouveaux objets mais aussi une propriété structurelle forte.

Navigation dans un treillis défini à partir d'une table discrétisée

Lorsqu'un treillis de Galois est défini à partir d'une table discrétisée, comme pour l'arbre de classification, les tests associés aux branches ne sont plus des égalités entre modalités d'un attribut qualitatif mais des tests d'appartenance d'une valeur à un intervalle.

Comme nous l'avons mentionné dans la section 3.4, lors de la classification d'un nouvel objet, par navigation dans le diagramme de Hasse du treillis, plusieurs tests peuvent être validés au cours d'une même étape de navigation. Lorsque les tests validés sont associés à des attributs qualitatifs, le choix du meilleur concept successeur repose alors sur des critères tels que le critère de support présenté dans cette même section 3.4.

Plus généralement, comme défini par Guillas [Guillas 07], lors de la classification d'un objet quantitatif, le choix du meilleur concept successeur est fait selon la distance entre les valeurs vérifiées par l'objet à classer et les points de référence (par exemple valeur minimale, maximale ou encore médiane de l'intervalle) **des intervalles tests.**

Dans la méthode Navigala ([Guillas 07], p. 126-127), différents points de références sont proposés, le centre, la médiane, le point le plus proche, ...

De plus, la notion d'intervalles à bornes floues est définie ([Guillas 07], p. 104-105).

Les intervalles à bornes floues sont définis à partir des intervalles créés lors de la discrétisation, et sont paramétrés par un degré de flou $\theta \in [0, 1[$. La Figure 5.2.12, extraite de [Guillas 07], présente cette notion de bornes floues associées à un intervalle (l'intervalle d'origine couvre $I1$ et $I2$, les bornes floues appliquées à l'intervalle correspondent aux points d'intersection de l'axe horizontal et des segments rouges).

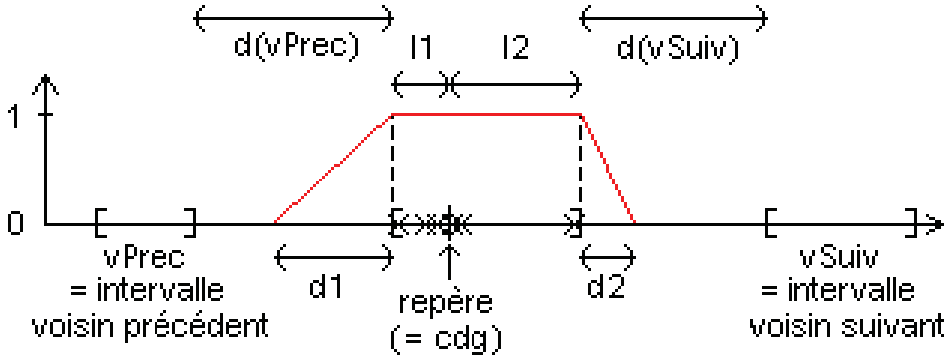


FIGURE 5.2.12: Définition d'un intervalle à bornes floues

Les bornes floues d'un intervalle sont définies selon la distribution des valeurs observées dans l'intervalle, mais aussi selon la proximité avec les intervalles précédent et suivant. Dans la Figure 5.2.12, un intervalle I est symbolisé par ses 2 bornes et par un ensemble de croix représentant la distribution des valeurs observées. De plus, *vPrec* et *vSuiv* sont les intervalles voisins précédent et suivant l'intervalle *I*. *d(vPrec)* et *d(vSuiv)* sont respectivement les distances entre les bornes des intervalles *vPrec* et *I*, et entre les bornes des intervalles *I* et *vSuiv*. La valeur médiane de I est utilisée comme point de repère (notée *cdg* dans la Figure). Les distances du point repère à la borne inférieure et à la borne supérieure de I sont notées respectivement l_1 et l_2. Les bornes floues inférieure et supérieure d'un intervalle I sont alors définies comme suit :

- la borne inférieure $d_1 = min(f(l_2; \theta); d(vPrec))$
- la borne supérieure $d_2 = min(f(l_1; \theta); d(vSuiv))$

Avec $f(l, \theta) = l \times \theta$, et θ le degré de flou.

Lorsque ces intervalles à bornes floues sont définis, le processus de navigation est mis en œuvre dans Navigala. Lors d'une étape de navigation, le choix du meilleur concept successeur est mené de manière progressive :

1. Choix du successeur pour lequel tous les intervalles sont validés, *i.e.* les valeurs vérifiées par l'objet à classer tombent exactement dans les intervalles test ;
2. En cas d'égalité entre plusieurs successeurs, assouplissement du traitement en validant le successeur pour lequel tous les intervalles flous sont validés ;
3. En cas de nouvelle égalité entre plusieurs successeurs, choix du successeur pour

lequel la distance moyenne est la plus petite, *i.e.* la distance entre les valeurs vérifiées par l'objet à classer et les médianes des intervalles tests.

L'utilisation des bornes floues est mise en place dans Navigala du fait que la discrétisation utilisée considère comme points de coupe, des valeurs vérifiées par les objets de l'ensemble de données analysé. De ce fait, il existe un espace entre deux intervalles consécutifs permettant d'y définir des bornes floues.

Pour notre part, nous utilisons les points de coupe tels que définis précédemment, *i.e.* les points de coupe sont des valeurs intermédiaires entre deux valeurs observées (*cf.* définition C_V, section 5.2, discrétisation en intervalles disjoints), **aussi le point 2 n'est pas utilisé par la navigation,** *d(vPrec)* et *d(vSuiv)* étant nulle**s.**

Exemple. La Figure 5.2.13 illustre le diagramme de Hasse du treillis de Galois défini à partir de la Table discrétisée 5.7(iv). Les tests d'appartenance d'une valeur à un intervalle sont représentés sur les branches. La classification de l'objet o_{11} est illustrée par le chemin en pointillés gris. Lors de la première étape de navigation, deux successeurs sont validés par o_{11}, le concept $(\{o_1, o_2, o_3, o_4, o_5\}, I_{.1} = [0, 4.5])$ et le concept $(\{o_1, o_2, o_9\}, I_{.2} = [0, 8])$. La distance de la valeur vérifiée par o_{11} pour $I_{.1}$ (2) à la médiane de $[0, 4.5]$ (1) étant plus faible que la distance de la valeur vérifiée par o_{11} pour $I_{.2}$ (8) à la médiane de $[0, 8]$ (4), le premier est choisi. Par navigation, l'objet o_{11} est classé dans la classe 1.

Propriété structurelle des treillis définis à partir d'une table discrétisée

Lorsqu'une table de données qualitatives binaires est issue d'un processus de discrétisation en intervalles disjoints, cela lui confère une propriété importante. **En effet, à tout attribut quantitatif, est associé un attribut qualitatif dont les modalités sont les intervalles disjoints construits par discrétisation. Ces intervalles étant disjoints, ils forment une partition de l'ensemble des objets.** Par définition des intervalles disjoints, pour chaque attribut qualitatif $I_{.h} \in \mathcal{I}$, chaque objet vérifie un seul et unique intervalle, *i.e.* une seule et unique modalité.

Ainsi, lorsque les modalités/intervalles sont considérés comme attributs binaires dans le contexte (ou la table binaire), ces derniers sont complémentaires (*cf.* section 4.4).

Exemple. Soit la Table binaire réduite 5.8 issue de la binarisation et de la réduction de la Table discrétisée 5.7(iv).

Dans cette table, l'attribut binaire I_1^1 a pour attributs complémentaires I_1^2 et I_1^3 (*i.e.* $\beta(I_1^1) \cup \beta(I_1^2) \cup \beta(I_1^3) = O$ et $\beta(I_1^1) \cap \beta(I_1^2) = \beta(I_1^1) \cap \beta(I_1^3) = \emptyset$), la complémentarité inverse est aussi vérifiée. De plus, I_2^1 a pour attribut complémentaire I_2^2 (*i.e.* $\beta(I_2^1) \cup \beta(I_2^2) = O$ et $\beta(I_2^1) \cap \beta(I_2^2) = \emptyset$).

La complémentarité entre attributs induit la propriété de $\vee$-complémentarité pour le treillis associé. De plus, nous avons vu dans la section 4.4.1, que tout treillis défini à partir d'une table d'attributs complémentaires est un treillis dichotomique. D'où la propriété suivante [Bertet 09] :

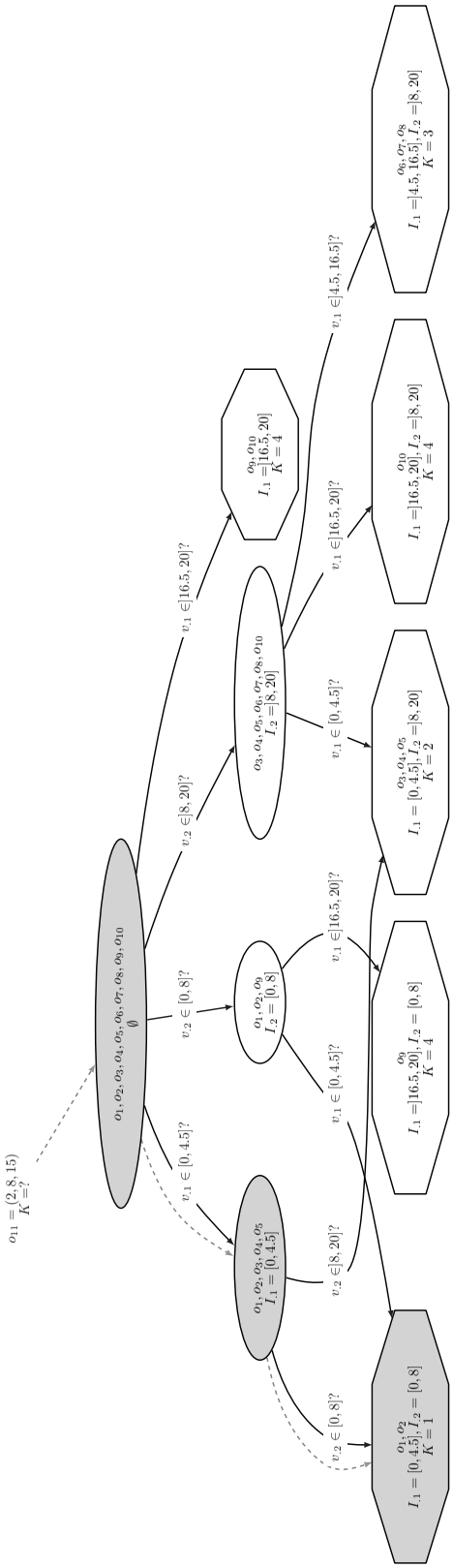


FIGURE 5.2.13: Classification de l'objet

O	$\mathcal{I}$					K
	$I_{.1}$			$I_{.2}$		
	$I_1^1 = [0-4,5]$	$I_1^2 =]4,5-16,5]$	$I_1^3 =]16,5-20]$	$I_2^1 = [0-8]$	$I_2^2 =]8-20]$	
o_1	×			×		1
o_2	×			×		
o_3	×				×	2
o_4	×				×	
o_5	×				×	
o_6		×			×	3
o_7		×			×	
o_8		×			×	
o_9			×	×		4
o_{10}			×		×	

TABLE 5.8: Contexte formel issu de la binarisation-réduction de la Table discrétisée 5.7(iv)

Proposition 8. *[Bertet 09]Les treillis définis à partir d'une table discrétisée en intervalles disjoints sont des treillis dichotomiques.*

Discussion

Cette proposition, permet de faire le lien entre le type de données obtenues par discrétisation en intervalles disjoints (attributs complémentaires) et la structure même du treillis associé. Comme nous l'avons vu dans la section 4.4, les treillis dichotomiques ont des propriétés structurelles fortes ($\vee$-complémentarité et co-atomicité) et ont un lien de fusion intéressant avec les arbres de classification.

Certaines études [Quinlan 96b] ayant montré que la discrétisation locale pouvait améliorer les résultats en classification des modèles à base de graphe tels que les arbres de classification, nous nous sommes donc intéressés à l'extension de la discrétisation locale aux treillis de Galois en nous appuyant sur les propositions 8, 6 et 5.

5.3 Contribution 2 - Discrétisation locale et treillis de Galois

Nous avons vu, dans le chapitre 4, qu'il existe des liens forts entre treillis de Galois et arbres de classification. Nous venons de voir, dans la section précédente, que malgré leurs liens, ces deux modèles utilisent deux types de discrétisations différentes pour traiter les ensembles de données quantitatives. Pour la construction des treillis, une étape

de discrétisation globale est mise en œuvre tandis que pour les arbres de classification il s'agit, en général, d'une discrétisation locale.

Comme mentionné précédemment, les études sur l'impact d'une discrétisation locale (par rapport à une discrétisation globale), sur la structure des arbres et sur leurs performances en classification tendent à montrer que la discrétisation locale peut améliorer les performances en classification des arbres de classification. Cependant la discrétisation locale induit des arbres plus complexes [Dougherty 95, Quinlan 96b].
Des expérimentations sur des bases d'images [Guillas 07, Visani 11] montrent que le treillis de Galois obtient de meilleurs résultats en classification que les arbres de classification, lorsque les deux structures sont définies à partir des mêmes données binaires ou des mêmes données discrétisées (cas de discrétisation globale).

En s'appuyant sur la discrétisation locale telle qu'elle est définie dans les arbres de classification, et sur la propriété des treillis de Galois lorsque ceux-ci sont définis à partir de données discrétisées, nous avons développé une discrétisation locale pour les treillis de Galois. **L'objectif majeur est d'améliorer ses performances en classification.**

5.3.1 Définition générale

De manière similaire à la discrétisation locale dans les arbres de classification (*cf.* sections 5.2.4 et 5.2.6)**, nous proposons une discrétisation locale pour les treillis de Galois. Cette discrétisation est basée sur la propriété des concepts terminaux et sur la propriété des treillis définis à partir d'une table discrétisée.**

Pour définir une discrétisation locale, il faut d'abord définir sur quels sous-ensembles de données la discrétisation doit être menée. Pour les treillis, les concepts terminaux sont l'équivalent des feuilles dans les arbres (*i.e.* éléments de fin des chemins de navigation). Or, selon la propriété 1 (chapitre 3), les inf-irréductibles sont des concepts terminaux. De plus, les treillis définis à partir de données discrétisés sont des treillis dichotomiques (cf. propriété 8), ils sont donc coatomistique. Aussi **la discrétisation locale doit être menée sur l'ensemble M des coatomes.**

Contrairement à un arbre de classification, la discrétisation d'un attribut dans un concept du treillis impacte une grande partie des autres concepts du treillis. En effet, quand un attribut est discrétisé dans un nœud de l'arbre, les prédécesseurs et les autres branches ne sont pas touchés. **Dans un treillis par contre, discrétiser un attribut dans un concept revient à créer deux attributs dans le contexte. Par définition des treillis de Galois** (définition 16)**, tous les concepts contenant l'attribut discrétisé sont modifiés et la structure du treillis est mo-**

difiée. De plus, par définition de la relation d'ordre $\leq$ entre les concepts (*cf.* équation (3.2.3)), **les relations entre concepts sont également modifiées**.

Cependant, les coatomes peuvent être calculés directement à partir du contexte [Wille 82]. Même s'il existe des algorithmes incrémentaux efficaces [Godin 94], la génération complète du treillis à chaque étape de discrétisation est inutile. **A chaque étape de discrétisation, le contexte est mis à jour avec les intervalles créés, et les coatomes sont recalculés pour mener une nouvelle étape de discrétisation. Le processus est itéré jusqu'à ce que tous les coatomes satisfassent le critère d'arrêt.**

Ainsi, chaque étape de discrétisation n'est pas appliquée à un unique concept, contrairement aux arbres de classification où chaque nœud est traité indépendamment des autres, **mais à l'ensemble M des coatomes ne satisfaisant pas le critère d'arrêt. Nous adaptons alors le choix du meilleur attribut comme suit :**

1. Pour chaque coatome $(A, B) \in M$, le meilleur attribut $V^*((A, B))$ et son meilleur point de coupe sont calculés localement, *i.e.* de manière similaire aux calculs effectués pour un nœud dans un arbre de classification ;
2. **Le meilleur attribut pour l'ensemble des coatomes $V^*(M)$ est ensuite calculé**, *i.e.* il est choisi parmi l'ensemble des meilleurs attributs définis pour chaque coatome (*cf.* section suivante équations (5.3.2) et (5.3.3)) ;

Seul ce dernier attribut $V^*(M)$ est discrétisé. Le contexte est mis à jour avec les deux intervalles créés et l'ensemble M des coatomes est recalculé à partir du contexte mis à jour. Le processus est réitéré sur le nouvel ensemble M.

L'Algorithme 5.2 présente notre algorithme de discrétisation locale pour les treillis de Galois. Une binarisation de la table de données quantitatives est d'abord réalisée pour initialiser le contexte. Comme l'illustre la Table 5.9(ii), la binarisation associe à chaque attribut quantitatif, l'intervalle couvrant toutes les valeurs observées dans l'ensemble de données. Chaque intervalle est un attribut binaire dans le contexte, donc chaque objet vérifie chaque attribut binaire. Le treillis correspondant ne contient alors qu'un seul concept, composé de tous les objets et tous les attributs. Ce treillis contient donc un seul inf-irréductible (O, I) qui permet d'initialiser M.

La discrétisation consiste alors à calculer le meilleur attribut $V^*(M)$, puis à le remplacer dans le contexte par les deux intervalles créés. L'ensemble M des coatomes ne satisfaisant pas le critère d'arrêt est alors recalculé à partir du contexte mis à jour.

Tant qu'il existe des coatomes ne satisfaisant pas le critère d'arrêt, la discrétisation est réitérée.

A partir d'un contexte binaire, les coatomes se calculent soit, comme les inf-irréductibles,

par calcul des fermetures sur les objets (*cf.* section 3.2.4), soit plus simplement en générant les prédécesseurs immédiats du concept maximal $\top$ en adaptant la fonction successeur de Bordat (*cf.* section 3.3.2).

Algorithme 5.2 Discrétisation locale pour les treillis

Entrées :

- $\Omega = (O, \mathcal{V}, K)$ l'ensemble de données quantitatives étiquetées, avec $\mathcal{V} = \{V_{.h}, \forall h \in \{1, \ldots, H\}\}$;
- *gain* le critère de coupe;
- $\mathcal{S}$ le critère d'arrêt;

Sorties :

$\Omega' = (O, \mathcal{I}, K)$ la table discrétisée binaire (ensemble de données qualitatives étiquetées);

DÉBUT

Initialiser Ω' avec :

- sur chaque ligne : un objet $o_n \in O$;
- sur chaque colonne : un attribut qualitatif $I_{.h}$ avec pour unique modalité l'intervalle contenant toutes les valeurs observées pour l'attribut $V_{.h}$ de la colonne;
- dans chaque case une $\times$ indiquant que chaque objet vérifie chaque attribut $I_{.h}$;

$M \leftarrow \{(O, \mathcal{I})\}$;
tant que $M \neq \emptyset$ **faire**
Calculer selon *gain*, le meilleur attribut $V^*(M)$ avec son meilleur point de coupe $c^*_{V^*}$; //(équations 5.3.2,5.3.3)
Couper $V^*(M)$ en deux intervalles disjoints I^1, I^2 selon $c^*_{V^*}$;
Remplacer dans Ω' l'intervalle correspondant à $V^*(M)$ par I^1 ou I^2;
Mettre à jour les relations objets $\times$ intervalles dans Ω';
Recalculer l'ensemble des coatomes du treillis;
Conserver dans M les coatomes ne satisfaisant pas le critère d'arrêt $\mathcal{S}$;
fin tant que

renvoyer Ω';

End

Exemple. Appliquer l'Algorithme 5.2 à l'ensemble de données quantitatives étiquetées déjà utilisé en exemple précédemment, *(cf.* Table 5.9(i)) induit, lors de l'initialisation, la construction de la Table 5.9(ii) et de l'ensemble d'inf-irréductibles M contenant uniquement $\bot$.

Trois étapes de discrétisation sont nécessaires pour que l'ensemble des coatomes satisfassent le critère d'arrêt, *i.e.* qu'ils ne contiennent que des objets appartenant tous à la même classe ou qu'il n'y ait plus de coupe pertinente. Ces trois étapes de mise à jour de Ω' et de M sont présentées dans les Tables 5.10, 5.11 et 5.12.

Lors de la seconde étape (Table 5.11), le point de coupe a été choisi localement dans le concept $(\{o_6, o_7, o_8, o_9, o_{10}\}, \{I_{.1} =]4.5, 20], I_{.2} = [0, 20], I_{.3} = [15, 18]\})$, il permet de séparer les objets de ce concept dans deux concepts différents contenant chacun des classes différentes. Ce choix ne permet pas de séparer les objets du second concept $(\{o_1, o_2, o_3, o_4, o_5\}, \{I_{.1} = [0, 4.5], I_{.2} = [0, 20], I_{.3} = [15, 18]\})$. Une étape supplémentaire de discrétisation est donc nécessaire.

Lors de la dernière étape, seul ce concept est pris en compte pour définir le meilleur attribut $V^*(M)$ et son meilleur point de coupe. Nous pouvons voir dans la Table 5.12 que malgré ce choix local, cela impacte le concept $(\{o_9, o_{10}\}, \{I_{.1} =]16.5, 20], I_{.2} = [0, 20], I_{.3} = [15, 18]\})$ qui se retrouve divisé en deux concepts malgré le fait qu'il satisfasse le critère d'arrêt.

Cet exemple nous permet de mettre en avant la différence entre le déroulement d'une discrétisation locale dans un arbre de classification (cette dernière impacte uniquement les nœuds successeurs du nœud courant), **par rapport à la discrétisation locale dans un treillis** (cette dernière impacte tous les concepts qui contiennent l'attribut discrétisé).

En sortie de l'Algorithme 5.2, la table discrétisée est renvoyée. Nous pourrions aussi construire le treillis et le considérer comme résultat de l'algorithme. Cependant dans [Bertet 07, Visani 11], une génération du treillis à la demande a été définie en utilisant la fonction successeur de Bordat (*cf.* section 3.4.2). De fait, pour la classification de tout nouvel objet, seul le contexte, *i.e.* la table discrétisée, est nécessaire.

5.3.2 Adaptation des critères de coupe pour la discrétisation locale dans les treillis

De manière plus formelle, chaque étape de discrétisation est menée sur l'ensemble des coatomes ne satisfaisant pas le critère d'arrêt, notons cet ensemble $M = \{(A_q, B_q),\ q = \{1, \ldots, Q\}\}$.

Critère de coupe

Tout d'abord (point 1 mentionné précédemment), pour chaque coatome $\Omega_{M_q} = (A_q, B_q) \in M$, le meilleur attribut $V^*(\Omega_{M_q})$ est calculé localement selon l'équation

O	$\mathcal{V}$			K
	$V_{.1}$ [0,20]	$V_{.2}$ [0,20]	$V_{.3}$ [15,18]	
o_1	1	4	15	1
o_2	0	0	18	
o_3	1	12	16	2
o_4	0	16	17	
o_5	3	12	18	
o_6	8	16	15	3
o_7	6	20	17	
o_8	15	12	16	
o_9	18	4	17	4
o_{10}	20	12	18	

(i) Ensemble de données quantitatives - Ω

O	$\mathcal{I}$			K
	$I_{.1}$ [0,20]	$I_{.2}$ [0,20]	$I_{.3}$ [15,18]	
o_1	×	×	×	1
o_2	×	×	×	
o_3	×	×	×	2
o_4	×	×	×	
o_5	×	×	×	
o_6	×	×	×	3
o_7	×	×	×	
o_8	×	×	×	
o_9	×	×	×	4
o_{10}	×	×	×	

Coatomes associés
Ne satisfaisant pas $\mathcal{S}$
$(O, \mathcal{I})$
Satisfaisant $\mathcal{S}$

(ii)Initialisation - Binarisation de la table de données quantitatives - Ω' ; Calcul de M

Table 5.9: Données d'entrées de l'Algorithme 5.2 et initialisation

O	$\mathcal{I}$			K
	$I_{.1}$ [0,4.5]	$I_{.1}$]4.5,20]	$I_{.2}$ [0,20]	$I_{.3}$ [15,18]	
o_1	×		×	×	1
o_2	×		×	×	
o_3	×		×	×	2
o_4	×		×	×	
o_5	×		×	×	
o_6		×	×	×	3
o_7		×	×	×	
o_8		×	×	×	
o_9		×	×	×	4
o_{10}		×	×	×	

Coatomes associés
Ne satisfaisant pas $\mathcal{S}$ - M
$(\{o_1, o_2, o_3, o_4, o_5\}, \{I_{.1} = [0, 4.5], I_{.2} = [0, 20], I_{.3} = [15, 18]\})$
$(\{o_6, o_7, o_8, o_9, o_{10}\}, \{I_{.1} =]4.5, 20], I_{.2} = [0, 20], I_{.3} = [15, 18]\})$
Satisfaisant $\mathcal{S}$

Table 5.10: Étape 1, mise à jour de Ω' et calcul des coatomes

O	$\mathcal{I}$					K
	$I_{.1}$			$I_{.2}$	$I_{.3}$	
	[0,4.5]	]4.5,16.5]	]16.5,20]	[0,20]	[15,18]	
o_1	×			×	×	1
o_2	×			×	×	
o_3	×			×	×	2
o_4	×			×	×	
o_5	×			×	×	
o_6		×		×	×	3
o_7		×		×	×	
o_8		×		×	×	
o_9			×	×	×	4
o_{10}			×	×	×	

Coatomes associés
Ne satisfaisant pas $\mathcal{S}$ - M
$(\{o_1, o_2, o_3, o_4, o_5\}, \{I_{.1} = [0, 4.5], I_{.2} = [0, 20], I_{.3} = [15, 18]\})$
Satisfaisant $\mathcal{S}$
$(\{o_6, o_7, o_8\}, \{I_{.1} =]4.5, 16.5], I_{.2} = [0, 20], I_{.3} = [15, 18]\})$ $(\{o_9, o_{10}\}, \{I_{.1} =]16.5, 20], I_{.2} = [0, 20], I_{.3} = [15, 18]\})$

Table 5.11: Étape 2, mise à jour de Ω' et calcul des coatomes

O	$\mathcal{I}$					K
	$I_{.1}$			$I_{.2}$	$I_{.3}$	
	[0,4.5]	]4.5,16.5]	]16.5,20]	[0,8]]8,20]	[15,18]	
o_1	×			×	×	1
o_2	×			×	×	
o_3	×			×	×	2
o_4	×			×	×	
o_5	×			×	×	
o_6		×		×	×	3
o_7		×		×	×	
o_8		×		×	×	
o_9			×	×	×	4
o_{10}			×	×	×	

Coatomes associés
Ne satisfaisant pas $\mathcal{S}$ - M
Satisfaisant $\mathcal{S}$
$(\{o_1, o_2\}, \{I_{.1} = [0, 4.5], I_{.2} = [0, 8], I_{.3} = [15, 18]\})$
$(\{o_3, o_4, o_5\}, \{I_{.1} = [0, 4.5], I_{.2} =]8, 20], I_{.3} = [15, 18]\})$
$(\{o_6, o_7, o_8\}, \{I_{.1} =]4.5, 16.5], I_{.2} =]8, 20], I_{.3} = [15, 18]\})$
$(\{o_9\}, \{I_{.1} =]16.5, 20], I_{.2} = [0, 8], I_{.3} = [15, 18]\})$
$(\{o_{10}\}, \{I_{.1} =]16.5, 20], I_{.2} =]8, 20], I_{.3} = [15, 18]\})$

Table 5.12: Étape 3, mise à jour de Ω' et calcul des coatomes

suivante :

$$V^*(\Omega_{M_q}) = argmax_{V \in B_q}(gain(V, c^*_V, \Omega_{M_q})) \quad (5.3.1)$$

où chaque c^*_V est le meilleur point de coupe associé à chaque attribut $V \in B_q$ défini selon l'équation (5.2.3) pour le sous-ensemble d'objets A_q.

Chaque meilleur attribut $V^*(\Omega_{M_q})$ est bien défini localement, l'équation (5.3.1) est une réécriture de l'équation (5.2.5), appliquée au sous-ensemble de données Ω_{M_q} plutôt que Ω complet.

Soit l'ensemble $I^*_M = \left\{V^*(\Omega_{M_1}), \ldots, V^*(\Omega_{M_Q})\right\}$ des meilleurs attributs associés à chaque concept $\Omega_{M_q} = (A_q, B_q) \in M$. Le meilleur attribut $V^*(M)$ parmi les attributs de I^*_M (point 2 mentionné précédemment), se définit facilement par extension de l'équation (5.2.5).

Discrétisation locale :

La discrétisation locale sélectionne l'attribut $V \in I^*_M$ qui maximise le critère de coupe pour M, selon l'équation suivante :

$$V^*(M) = argmax_{V^*(\Omega_{M_q}) \in I^*_M}(gain(V^*(\Omega_{M_q}), c^*_{V^*(\Omega_{M_q})}, \Omega_{M_q})) \quad (5.3.2)$$

A noter que, à la définition près de I^*_M, l'équation (5.3.2) est proche de l'équation (5.2.5).

Le choix du meilleur point de coupe tel que proposé dans l'équation 5.3.2, ne prend pas en compte le fait que la discrétisation d'un attribut $V \in I^*_M$ selon c^*_V dans un concept $\Omega_{M_q} = (A_q, B_q)$ peut impacter d'autres concepts et en particulier d'autres coatomes de M, voire permettre de séparer les objets de classes différentes dans d'autres concepts dont les autres coatomes.

Pour prendre en compte ce phénomène spécifique au treillis, dans la sélection du meilleur point de coupe, nous proposons une seconde façon de définir $V^*(M)$. Nous nommons cette méthodologie discrétisation locale linéaire.

Discrétisation locale linéaire :

La discrétisation locale linéaire sélectionne l'attribut $V \in I^*_M$, qui maximise la somme pondérée du critère de coupe pour chaque $\Omega_{M_q} = (A_q, B_q) \in M$, selon l'équation suivante :

$$V^*(M) = argmax_{V \in I^*_M}\Big(\sum_{\Omega_{M_{q'}} \in M | V \in B_{q'}} \frac{|A_{q'}|}{m} * gain(V, c^*_V, \Omega_{M_{q'}})\Big) \quad (5.3.3)$$

Où $|A_{q'}|$ est le nombre d'objets contenus dans le coatome $\Omega_{M_{q'}}$ et $m = \sum_{\Omega_{M_q} \in M} |A_q|$.

Critère d'arrêt

Comme précisé précédemment, l'objectif de la discrétisation est d'obtenir des concepts terminaux contenant des objets appartenant à la même classe. Pour ce faire, le critère d'arrêt mis en place est relativement simple : si un coatome ne satisfait pas à cette condition (*i.e.* il contient des objets de classes différentes) alors il est ajouté à l'ensemble M pour une nouvelle étape de discrétisation locale.

Cependant, toutes les données ne sont pas forcement séparables. Dans ce cas, le critère de coupe est nul. Lorsque le critère de coupe est nul pour tous les concepts de M alors il n'existe plus de point de coupe pertinent qui permettrait de séparer les objets de classes différentes ; cela constitue un critère d'arrêt complémentaire au premier.

Nous utilisons donc la combinaison de ces deux critères dans notre algorithme pour stopper la discrétisation (*i.e.* tant qu'il existe des coatomes contenant des objets de classes différentes et des points de coupe pertinents, alors la discrétisation se poursuit, sinon elle s'arrête).

Une fois la discrétisation locale réalisée, le treillis est construit à partir de la table discrétisée, puis l'étiquetage des concepts terminaux est réalisé comme dans Navigala. Les concepts contenant uniquement des objets de la même classe sont étiquetés avec cette dernière et les coatomes non purs (cas des données non séparables) sont étiquetés avec leur classe majoritaire.

Afin de valider cette discrétisation locale, nous avons mené différentes expérimentations présentées dans la section suivante.

5.3.3 Expérimentations

5.3.3.1 Protocole expérimental

Critères d'études

Cette section présente différents résultats expérimentaux permettant tout d'abord de comparer les différentes discrétisations existantes pour les treillis, mais aussi de comparer les performances en classification de ces treillis par rapport à des modèles plus connus : arbre de classification et SVM.

Dans un premier temps, en sections 5.3.3.2 et 5.3.3.3, nous comparons les résultats obtenus en utilisant la discrétisation globale, la discrétisation locale et la discrétisation locale linéaire pour les treillis de Galois. En effet, pour les mêmes données quantitatives étiquetées, les différentes discrétisations génèrent des tables discrétisées différentes et donc des treillis de Galois différents. Dans la section 5.3.3.2, nous nous intéressons au

choix du meilleur critère de coupe. Puis, dans la section 5.3.3.3, nous étudions l'impact des différentes discrétisations sur la structure des treillis.

Dans la section 5.3.3.4, nous comparons les performances de ces treillis en classification supervisée par rapport à celles de modèles plus connus que sont l'arbre de classification et la SVM. Pour ce dernier point, la classification des objets, pour l'arbre et les treillis, se fait par navigation. Plus précisément pour le treillis de Galois, la méthode Navigala est utilisée (*cf.* section 5.2.6.2)[Guillas 07, Visani 11].

Les ensembles de données utilisés

Comme nous l'avons déjà évoqué précédemment, nous nous intéressons à la classification d'images lorsque celles-ci sont décrites par des descripteurs quantitatifs. Ces travaux faisant suite à la thèse de Guillas [Guillas 07], la première base exploitée est la base d'images de symboles GREC 2003 [GREC 03] (*cf.* exemples dans l'Annexe D). Différentes signatures ont été définies pour décrire ces images. Pour nos expérimentations, nous avons retenu la signature structurelle [Coustaty 11b] et la signature de Radon [Radon 17, Tabbone 03a, Tabbone 03b]. Nous étudions donc les deux ensembles de données quantitatives associées : GREC Structurelle et GREC Radon.

A ces deux ensembles, nous avons cherché à ajouter d'autres ensembles d'images avec les contraintes d'avoir des images supervisées et décrites par des attributs exclusivement quantitatifs. Nous avons donc choisi, parmi les ensembles de données supervisées de l'UCI Machine Learning Repository (*MLRepository*) [Frank 10], l'ensemble *Image segmentation*.

Nous avons aussi choisi de valider notre méthode sur des ensembles de données quantitatives supervisées fréquemment utilisés dans la littérature pour la comparaison des performances de modèles de classification.

En particulier, les ensembles de données très connus que sont Breast Cancer et Iris, mais aussi l'ensemble Glass.

Ces différents ensembles de données quantitatives étiquetées sont composés comme suit (pour plus de précisions sur chacun d'eux, voir les notices en Annexe D).

Image1 : L'ensemble de données *Image Segmentation* est composé de 2 310 objets distribués dans 7 classes. Les exemples ont été tirés au hasard à partir d'un ensemble de 7 classes d'images de photographies prises en extérieur. Chaque image est décrite par 19 descripteurs quantitatifs. Sur le MLRepository, les bases d'apprentissage et de test sont définies. La base d'apprentissage est composée de 210 objets (30 objets par classe) et la base de test est composée des objets restants, soit 2 100 objets (300 objets par classe).

GLASS : L'ensemble de données *GLASS* est une base d'identification de morceaux de verres. Il est composé de 214 objets répartis en 6 classes. Chaque objet est décrit par 9 descripteurs quantitatifs.

IRIS : L'ensemble de données *Iris*, l'un des plus connus, se compose de 150 objets répartis en 3 classes. Chaque objet est décrit par 4 descripteurs quantitatifs.

BREAST CANCER : L'ensemble de données *Wisconsin Breast Cancer*, qui est aussi très réputé, est composé de 699 objets distribués en 2 classes avec 65.5% des objets dans la classe *Benign* et 34.5% dans la classe *Malignant*. Chaque objet est décrit par 10 descripteurs quantitatifs.

GREC Structurelle : Le premier ensemble de données, extrait de la base GREC 2003, contient 1 900 exemples distribués en 19 classes. Chaque image est décrite par une signature structurelle [Coustaty 11b] composée de 15 descripteurs quantitatifs. Pour cet ensemble, une base d'apprentissage et une base de test sont données, la base d'apprentissage contient 10% des objets (soit 190 objets) et la base de test 90% des objets soit (1 710 objets). Notons que pour cet ensemble, les classes ne sont pas séparables, aussi lors des discrétisations, le critère d'arrêt utilisé correspond à l'absence de coupe pertinente.

GREC Radon : Le second ensemble, extrait de la base GREC 2003, contient 910 exemples distribués en 10 classes. Chaque image est décrite par une signature statistique de 50 descripteurs quantitatifs. Pour cet ensemble, comme pour une validation croisée à 10 paquets, dix bases d'apprentissages sont définies et chacune d'elles est associée à une base de test. Les bases d'apprentissage ont été définies avec 10% des objets soit 91 objets et les bases de test correspondent aux 90% d'objets non utilisés dans la base d'apprentissage, soit 819 objets. Comme pour une validation croisée classique, les taux de reconnaissances présentés ci-dessous correspondent aux moyennes des taux de reconnaissance obtenus sur chaque paquet.

Pour que les comparaisons soient cohérentes, lorsqu'une validation croisée est mise en œuvre, nous utilisons les mêmes paquets pour tous les algorithmes (*i.e.* ils ne sont pas redéfinis à la volée à chaque exécution). La Table 5.13 récapitule les informations sur chaque ensemble de données utilisé. La colonne "% BA" indique le pourcentage d'objets de l'ensemble utilisé en apprentissage. La colonne "% BT" indique le pourcentage d'objets de l'ensemble utilisé en test. La colonne "VC" indique, lorsqu'une validation croisée est mise en œuvre, le nombre de paquets utilisés.

5.3.3.2 Critères de coupe

Afin de déterminer le meilleur critère de coupe, nous avons effectué des tests avec quatre critères de coupe mentionnés précédemment. Plus précisément, nous avons utilisé les trois critères usuels des arbres de classification, *i.e.* le critère du χ^2, l'indice de Gini, le critère d'Entropie (gain ratio) et le critère utilisé dans la méthode Navigala, le critère de Hotelling.

Ensemble de données	#Attributs	#Exemples	#Classes	% BA	% BT	VC
Image1	19	2,310	7	10%	90%	ND
GLASS	9	214	6	90%	10%	10
IRIS	4	150	3	90%	10%	10
BREAST CANCER	10	699	2	90%	10%	10
GREC Structurelle	15	1,900	19	10%	90%	ND
GREC Radon	50	910	10	10%	90%	10

TABLE 5.13: Ensembles de données utilisés pour nos expérimentations

Les Figures 5.3.1, 5.3.2, et 5.3.3 présentent les taux de reconnaissance pour chaque ensemble de données et pour chaque critère de coupe en utilisant respectivement la discrétisation globale, la discrétisation locale, et la discrétisation locale linéaire. Les taux de reconnaissance moyens observés sont indiqués entre parenthèses à côté de chaque critère dans chaque Figure.

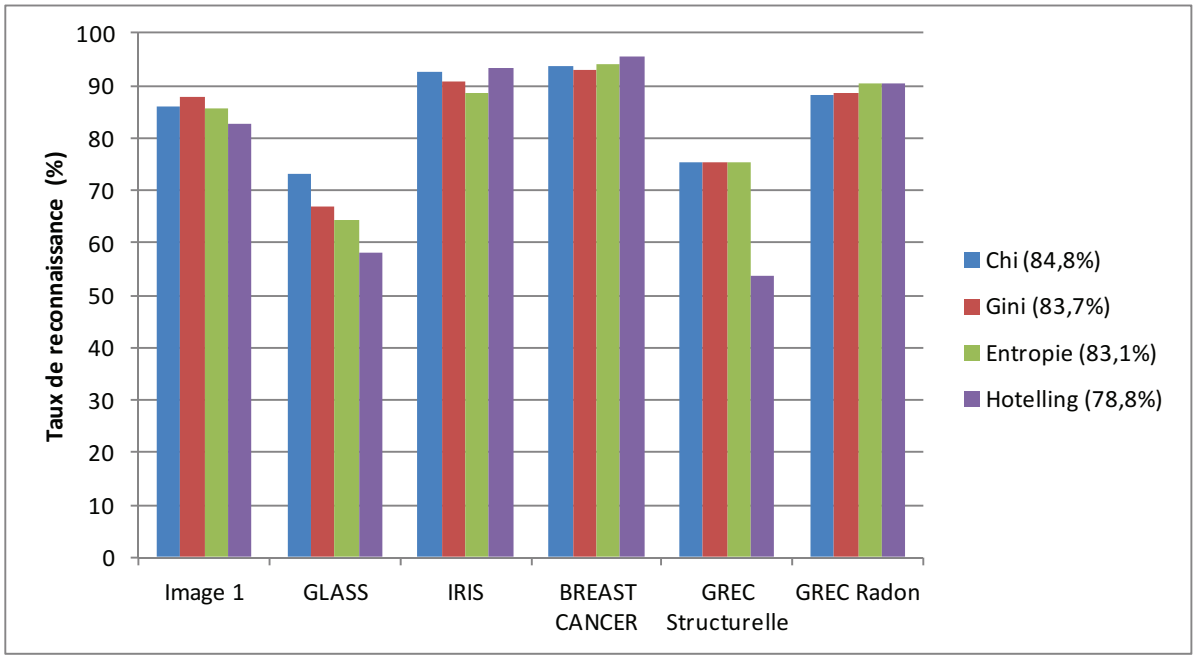


FIGURE 5.3.1: Discrétisation globale - comparaison des critères de coupe

Sur ces Figures, nous pouvons voir *via* les taux de reconnaissance moyens observés sur tous les ensembles de données que le critère du χ^2 surpasse le critère de Gini de 1% avec les discrétisations globale et locale linéaire, et de 2% avec la discrétisation locale. De plus, le critère du χ^2 surpasse aussi le critère d'Entropie de 0.4% avec la discrétisation locale linéaire, de $1,7\%$ avec la discrétisation globale et de 3% avec la discrétisation lo-

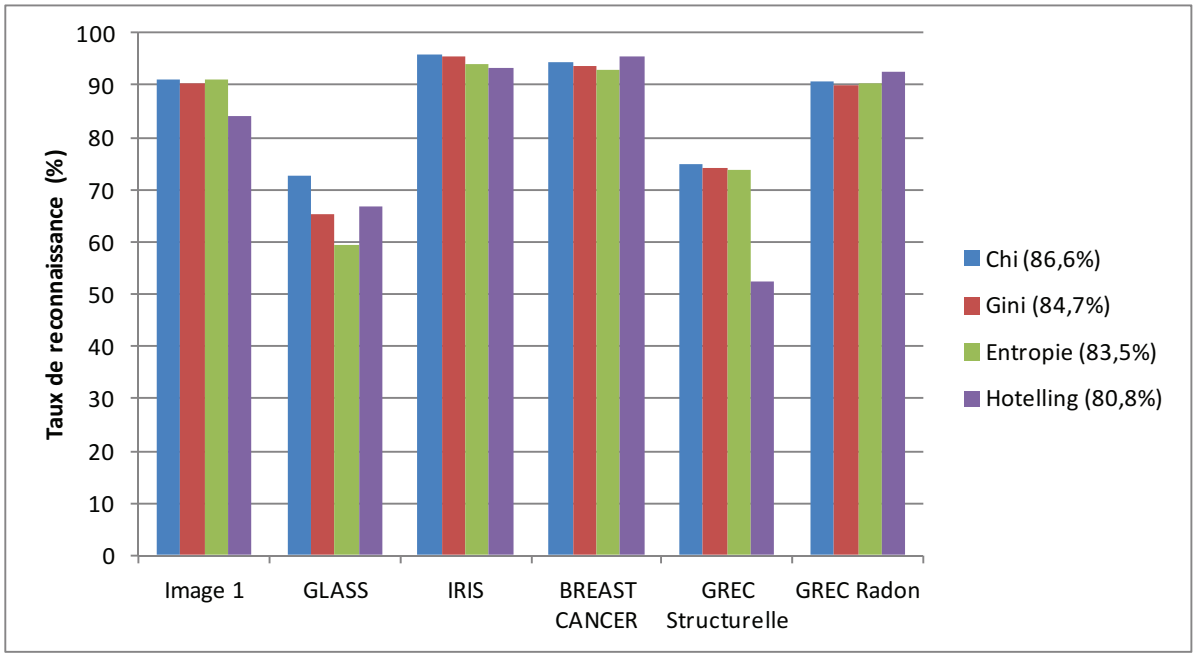


FIGURE 5.3.2: Discrétisation locale - comparaison des critères de coupe

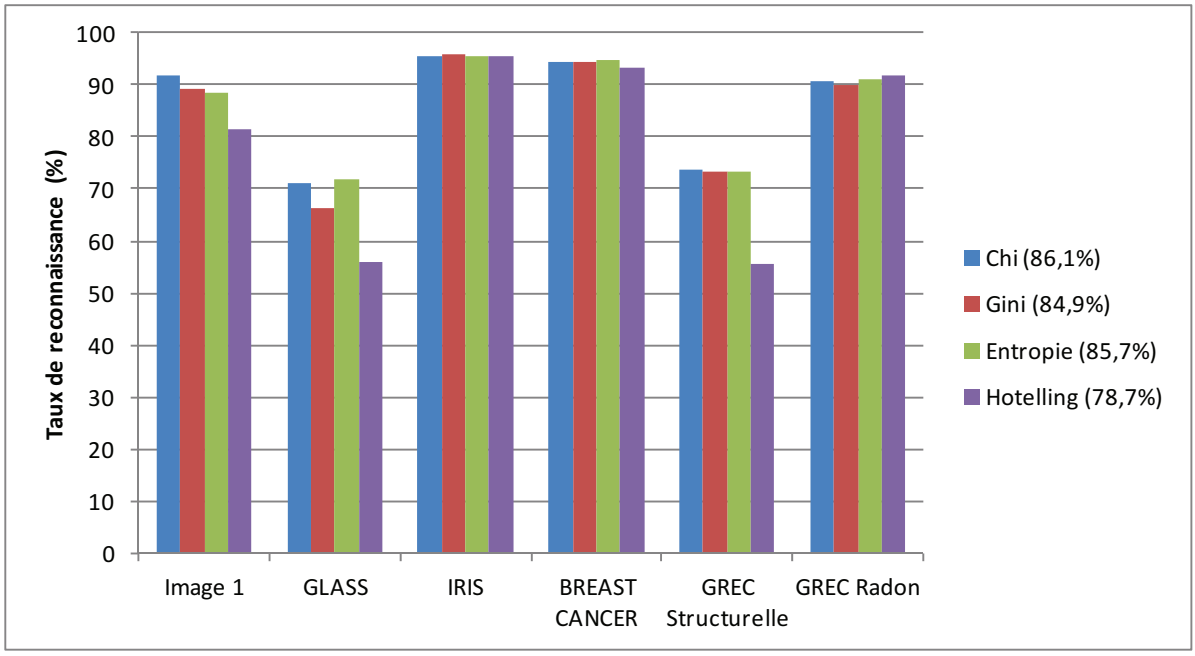


FIGURE 5.3.3: Discrétisation locale linéaire - comparaison des critères de coupe

cale. Enfin, le critère du χ^2 surpasse le critère de Hotelling de 6% avec les discrétisations globale et locale et de $7,3\%$ avec la discrétisation locale linéaire.

Nous pouvons donc conclure de manière générale que le critère du χ^2 donne les taux de reconnaissance moyens les plus satisfaisants.

De plus, tels que représentés dans les Figures 5.3.4, 5.3.5, 5.3.6 et 5.3.7, ses performances sont les plus stables au travers des différentes expériences, lorsqu'une validation croisée est mise en œuvre. En effet, les boîtes à moustaches présentées dans ces Figures 5.3.4, 5.3.5, 5.3.6, représentent les variations des taux de reconnaissance au cours des dix étapes des validation croisée pour les différents ensembles de données GLASS, IRIS, Breast Cancer et GREC Radon. Les valeurs minimales, maximales et médianes sont représentées ainsi que les valeurs du premier et du troisième quartiles. Nous pouvons voir que les médianes observées pour le critère du χ^2 sont, en général, mieux centrées que pour les autres autres critères.

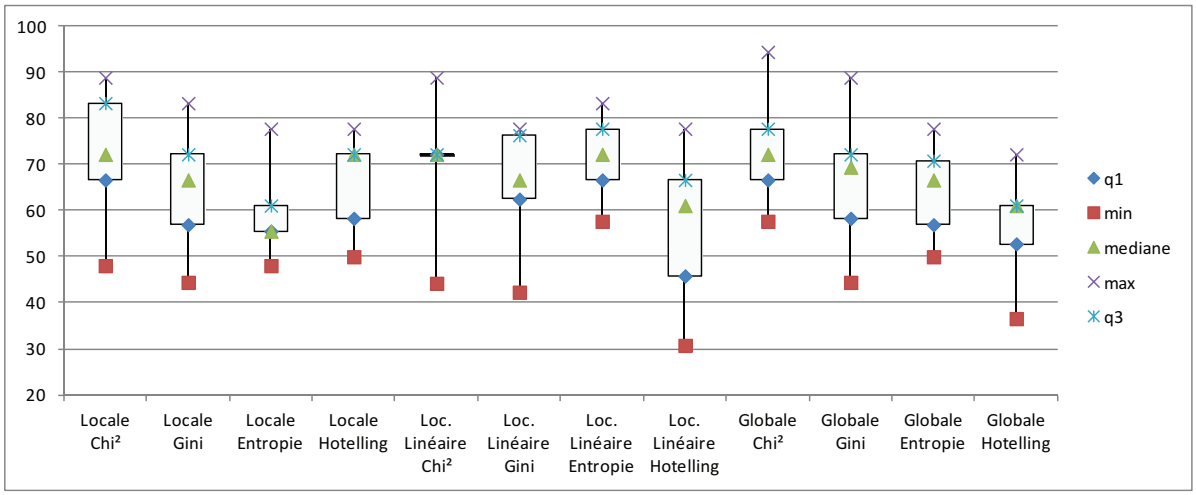


FIGURE 5.3.4: Boites à Moustaches - GLASS

Vu ces résultats, pour les expériences suivantes, nous utilisons exclusivement le critère du χ^2 comme critère de coupe.

5.3.3.3 Complexité algorithmique et complexité structurelle

Nous avons ensuite mené une étude comparative entre les différentes discrétisations d'un point de vu algorithmique et structurel. Même si le temps de calcul n'est pas une contrainte forte pour l'étape d'apprentissage, c'est un élément de comparaison intéressant.

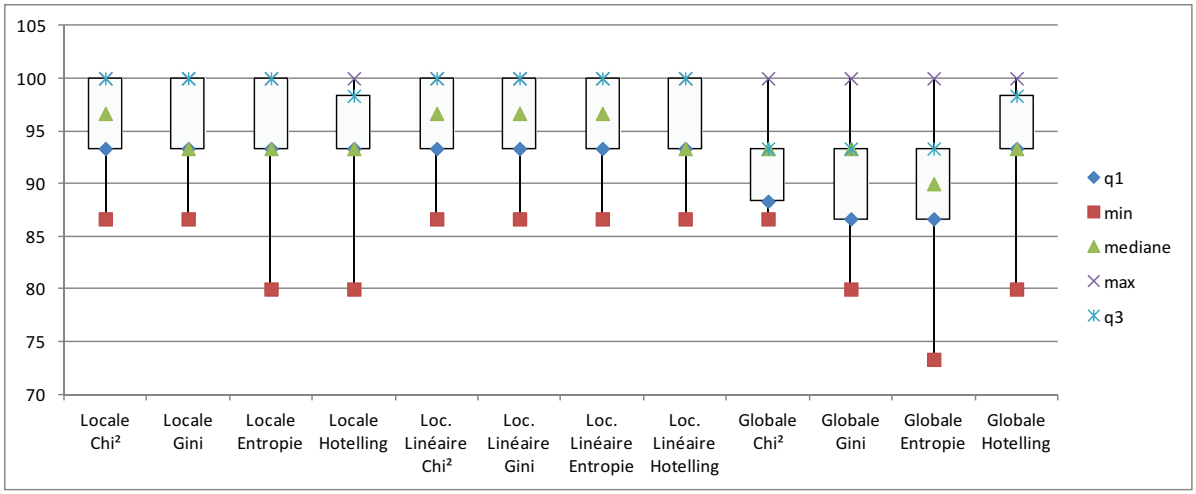


FIGURE 5.3.5: Boites à Moustaches - IRIS

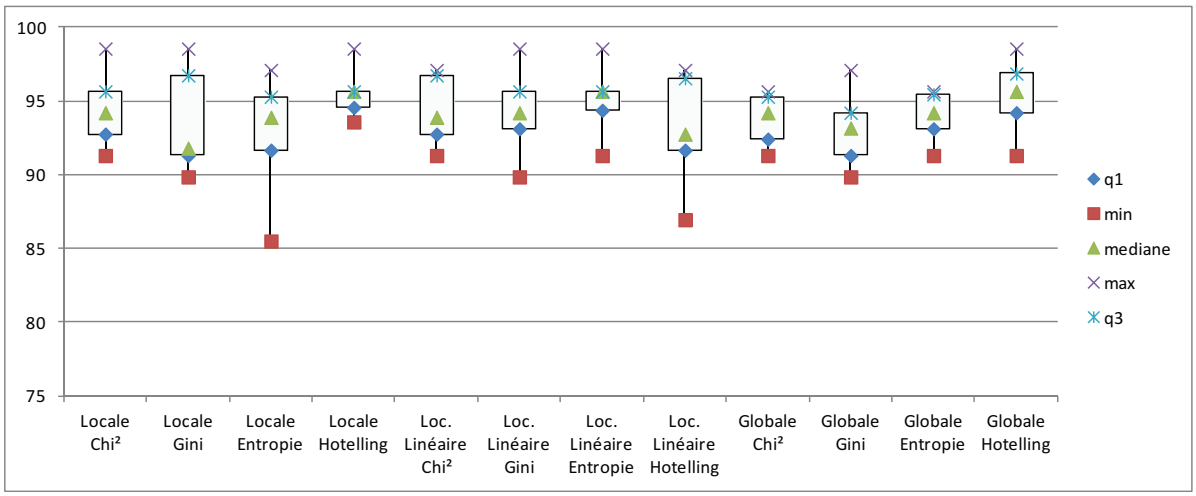


FIGURE 5.3.6: Boites à Moustaches - Breast Cancer

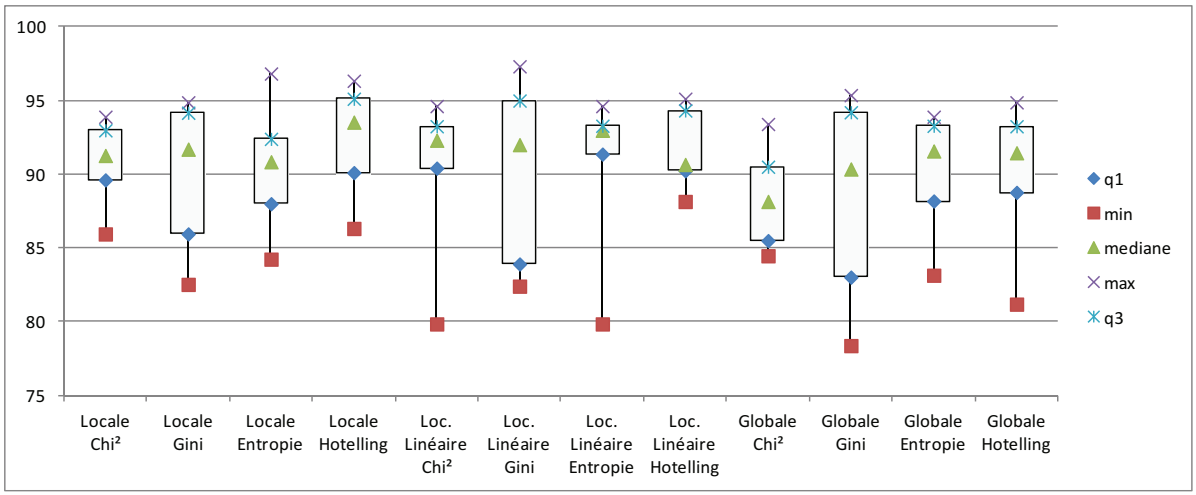


FIGURE 5.3.7: Boites à Moustaches - GREC Radon

Complexité algorithmique - Temps de calcul

Pour chaque ensemble de données, nous avons comparé le temps de calcul nécessaire pour chaque algorithme de discrétisation, en utilisant un cœur d'un processeur quadri-cœur Intel Xeon cadencé à 3 GHz avec 4 Go de mémoire physique dédiée.

La Figure 5.3.8 présente les temps de calcul nécessaires à chaque algorithme de discrétisation (l'axe du temps est présenté selon une échelle logarithmique). A chaque type de discrétisation est associée, entre parenthèses, la moyenne des temps de calcul sur tous les ensembles de données.

Nous pouvons déduire de cette expérimentation que la discrétisation globale nécessite plus de temps (jusqu'à 2 fois plus de temps) **que les algorithmes de discrétisation locale, quelque soit l'ensemble de données utilisé.**

Pour mieux comprendre cette différence de temps de calcul, nous avons mené des expérimentations sur le nombre d'étapes de discrétisation et le nombre de concepts du treillis.

Complexité algorithmique - Nombre d'étapes de discrétisation

Le temps de calcul est fortement lié au nombre d'étapes de discrétisation effectuées. En effet, le plus tôt l'algorithme trouve les meilleurs points de coupe pour séparer les objets de classes différentes dans les inf-irréductibles, le plus tôt le critère d'arrêt sera satisfait.

La Figure 5.3.9 présente le nombre d'étapes de discrétisation nécessaires à chaque algorithme et pour chaque ensemble de données.

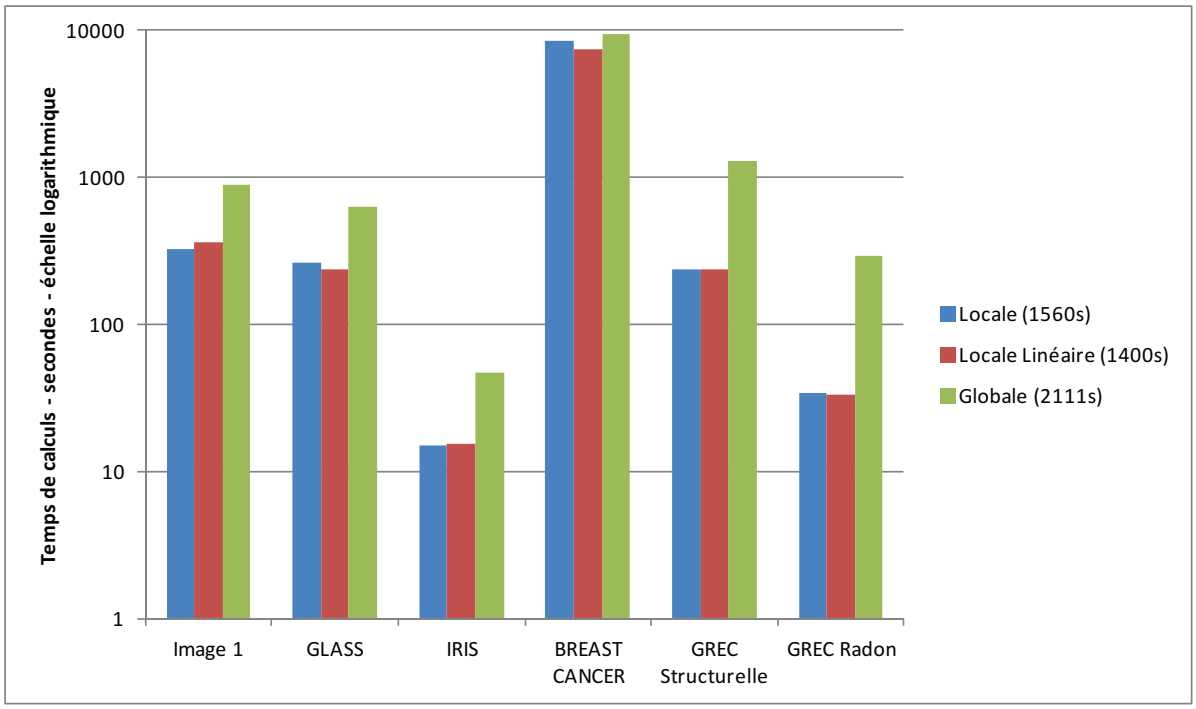


FIGURE 5.3.8: Temps de Calcul

Nous pouvons voir que la discrétisation locale linéaire nécessite, en moyenne, moins d'étapes que la discrétisation locale et la discrétisation globale. **Les discrétisations locale et locale linéaire sont donc moins coûteuses que la discrétisation globale.**

Ce résultat s'explique par le fait que le domaine de recherche des discrétisations locales est de plus en plus petit au fur et à mesure des étapes, tandis que la discrétisation globale parcourt l'ensemble complet des objets à chaque étape. En discrétisation globale, un point de coupe peut parfois être choisi pour séparer deux classes qui sont déjà séparées dans les inf-irréductibles. En discrétisation locale, le domaine de recherche correspond seulement à un sous-ensemble spécifique d'objets et d'attributs ; ainsi lorsqu'un attribut et un point de coupe sont choisis pour séparer deux classes, c'est que ces deux classes n'ont pas encore été séparées dans l'un des inf-irréductibles.

Complexité structurelle - Nombre de concepts

Dans leur article [Dougherty 95], Dougherty *et al.* expliquent que malgré le fait que la discrétisation locale améliore les performances des arbres de classification en classification, comme le démontre Quinlan dans [Quinlan 96b], celle-ci engendre la génération d'arbres plus complexes (*i.e.* contenant plus de nœuds). Qu'en est-il pour les treillis de Galois ?

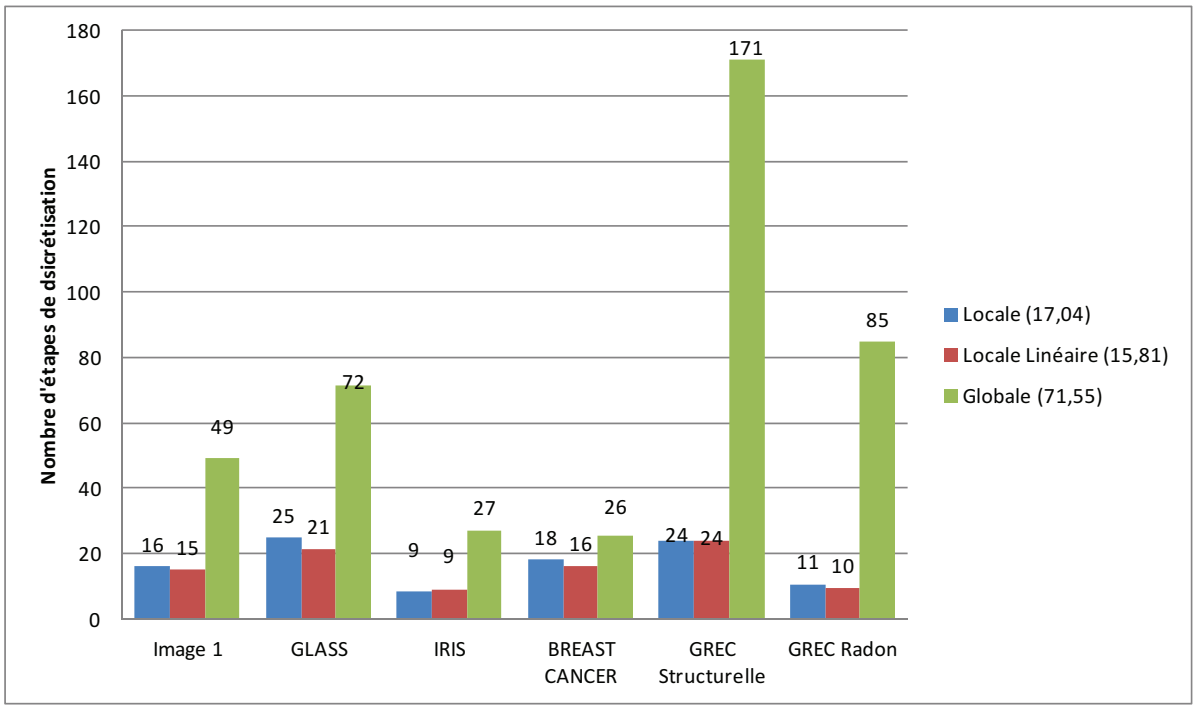


FIGURE 5.3.9: Nombre d'étapes de discrétisation

La Table 5.14 présente le nombre de concepts des treillis de Galois selon la discrétisation utilisée et pour chaque ensemble de données. Le nombre de concepts le plus faible est indiqué en gras pour chaque ensemble de données.

Pour chaque discrétisation locale et chaque ensemble de données, nous indiquons le pourcentage de concepts en moins comparé à la discrétisation globale.

	Discrétisation				
Ensembles de données	Globale	Locale		Locale linéaire	
Image1	12172	**581**	-95.22%	649	-94.66%
GLASS	2074	**2039**	-1.68%	2267	+9.30%
IRIS	195	**40**	-79.48%	41	-78.97%
BREAST CANCER	7784	**2887**	-62.91%	2961	-61.96%
GREC Structurelle	4308	**3515**	-18.40%	3851	-10.60%
GREC Radon	2192	**69**	-98.85%	90	-95.89%

TABLE 5.14: Nombre de concepts

Cette expérience montre que, pour les treillis de Galois, au contraire, la discrétisation locale permet une réduction de la complexité structurelle. En

effet, avec moins d'étapes de discrétisation, il y a moins d'intervalles générés, et par conséquent il y a moins de concepts dans le treillis.

Discussion

Ces expériences montrent que la discrétisation locale nous permet d'obtenir une table discrétisée plus adaptée à la structure des treillis. La discrétisation locale utilisant la structure même du treillis, elle choisit des points de coupe plus pertinents par rapport aux sous-ensembles d'objets contenus dans les concepts terminaux. L'aspect local de cette discrétisation, permet de choisir des points de coupe utiles pour séparer les classes qui n'ont pas déjà été séparées. Ainsi la table discrétisée ne contient pas d'intervalles inutiles.

Ainsi, la discrétisation locale (qu'elle soit linéaire ou non) améliore la complexité structurelle et algorithmique par rapport à la globale.

La prochaine section se concentre sur la comparaison des performances en classification des différents treillis obtenus et de deux modèles de classification supervisée plus connus, à savoir un SVM et un arbre de classification construit avec l'algorithme ChAID.

5.3.3.4 Performances en classification

La Table 5.15 présente précisément les performances en classification (taux de reconnaissance) des cinq modèles que sont les treillis de Galois définis avec les trois discrétisations, l'arbre de classification ChAID [Kass 80] (*cf.* section 2.5) qui utilise le χ^2 en tant que critère de coupe, et un SVM avec comme noyau, celui offrant le meilleur [2] taux de reconnaissance moyen sur les ensembles de données utilisés, à savoir la fonction à base radiale (*cf.* section 1.3.2.3 avec pour paramètre $\gamma = 1$). Lorsqu'une validation croisée est utilisée, la variance des taux de reconnaissance est indiquée.

Ces résultats sont aussi présentés sous forme d'histogramme dans la Figure 5.3.10.

Les performances respectives de chaque modèle dépendent des ensembles de données. Les meilleurs taux de reconnaissance sont indiqués en gras, et les meilleurs taux parmi tous les treillis de Galois sont indiqués en italique.

Pour Image1, la différence entre les modèles les plus performants, que sont les treillis définis avec les discrétisations locales, et les autres varie d'environ 1% à 6%. Le modèle le moins performant est le treillis de Galois défini avec la discrétisation globale, la plus

2. Plusieurs noyaux ont été testés, les polynômes de degrés 1, 2 et 3, la fonction à base radiale avec $\gamma = 0.001$,$\gamma = 0.5$ et $\gamma = 1$. Nous avons choisi celui offrant le meilleur taux de reconnaissance moyen sur les ensembles de données utilisés.

faible différence avec les autres modèles est de 2.5%. Le treillis de Galois défini avec la discrétisation locale linéaire obtient le meilleur taux de reconnaissance. **Pour cet ensemble de données, une discrétisation locale associée au treillis de Galois est toute indiquée.**

Pour GLASS, les treillis de Galois se démarquent des autres modèles. Ils enregistrent des taux de reconnaissance de 2% à 10% meilleurs que le SVM et ChAID. La discrétisation locale diminue légèrement (-0.4%) les performances des treillis comparée à la discrétisation globale. **Pour cet ensemble de données, l'utilisation des treillis est toute indiquée.**

Pour IRIS, la différence entre chaque modèle est comprise entre 1% et 4%. Le SVM donne le meilleur taux de reconnaissance, avec juste derrière (-0.6%) le treillis de Galois défini avec la discrétisation locale. Le treillis de Galois défini avec la discrétisation globale obtient le taux de reconnaissance le plus faible. **Pour cet ensemble de données, en cas d'utilisation des treillis de Galois, il est préférable d'utiliser une discrétisation locale.**

Pour BREAST CANCER, le SVM est meilleur que tous les autres modèles, avec une différence entre 2 et 3%. Les modèles obtenant les moins bons résultats sont l'arbre de classification et le treillis de Galois défini avec une discrétisation globale. Cependant, les deux treillis de Galois définis avec les discrétisations locales sont meilleurs que celui défini par discrétisation globale. **De nouveau, pour cet ensemble de données, en cas d'utilisation des treillis de Galois, il est préférable d'utiliser une discrétisation locale.**

Pour GREC Structurel, avec une différence d'environ 10%, les modèles symboliques (arbre de classification et treillis de Galois) donnent de meilleurs résultats que le modèle statistique (SVM). La discrétisation locale diminue légèrement (-0.4%) les performances des treillis comparée à la discrétisation globale. Comme pour GLASS, **pour cet ensemble de données, l'utilisation des treillis est toute indiquée.**

Pour GREC Radon, le SVM est meilleur que les autres modèles, avec une différence d'à peine 0.2% par rapport à ChAID et de 2.5% à 5% par rapport aux treillis. Cependant, les deux treillis de Galois définis avec les discrétisations locales sont meilleurs que celui défini par discrétisation globale. **De nouveau, pour cet ensemble de données, en cas d'utilisation des treillis de Galois, il est préférable d'utiliser une discrétisation locale.**

Notons que les treillis de Galois définis avec les deux discrétisations locales obtiennent une meilleure moyenne sur tous les ensembles de données que les autres modèles.

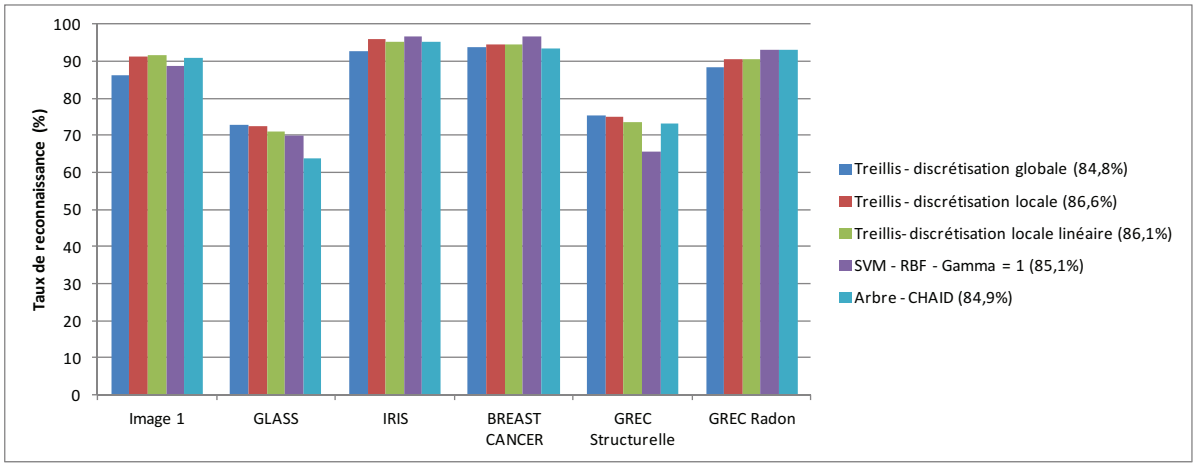


FIGURE 5.3.10: Taux de reconnaissance selon les différents modèles

Comme pour les arbres de classification, la discrétisation locale améliore en général les taux de reconnaissance des treillis de Galois. Lorsqu'elle diminue les performances, ce n'est que légèrement (-0.4%) et c'est au profit d'un modèle beaucoup moins complexe (moins de concepts : entre -1.6% et -18% ; temps de calcul diminué)**.**

Pour les ensembles de données quantitatives utilisés, **les deux discrétisations locales permettent aux treillis de Galois de s'approcher en termes de taux de reconnaissance avec les SVM.**

Ces expérimentations montrent **l'efficacité de la discrétisation locale proposée.**

5.4 Conclusion

Dans ce chapitre, nous avons rappelé le processus de discrétisation qui permet de transformer les données quantitatives en données qualitatives. La différence entre discrétisation globale et discrétisation locale a été présentée et étudiée.

Les arbres de classification utilisant en général une discrétisation locale, et celle-ci tendant à améliorer leurs performances en classification, nous nous sommes intéressés au développement d'une discrétisation locale pour les treillis de Galois.

En effet, pour définir ces derniers à partir d'un ensemble de données quantitatives, une discrétisation globale (dans les cas supervisés) ou une discrétisation en intervalles non disjoints (dans les cas non supervisés) étaient auparavant utilisées. La discrétisation en intervalles non disjoints pouvant conduire à une forte complexité dans la construction du treillis, nous avons privilégié la discrétisation en intervalles disjoints qui induit la

	Modèles de classification				
	Treillis de Galois/discrétisation			SVM	ChAID
Ensemble de données	Globale	Locale	Locale linéaire		
Image1	86,14	91,09	***91,71***	88,61	90,95
GLASS	***72,99*** ±10,34	72,58 ±13,36	71,09 ±10,53	69,78 ±11,92	63,72 ±5,96
IRIS	92,66 ±4,67	*96,00* ±4,42	95,33 ±5,21	**96,67** ±3,34	95,33 ±5,21
BREAST CANCER	93,72 ±1,63	*94,45* ±2,17	94,43 ±2,05	**96,70** ±2,16	93,47 ±2,74
GREC Structurelle	***75,43***	75,04	73,68	65,69	73,16
GREC Radon	88,27 ±2,93	*90,69* ±2,71	*90,69* ±4,33	**93,13** ±3,42	92,94 ±3,74
Moyenne	84,87	***86,64***	86,15	85,10	84,93
1er rang / nb ensembles	2 / 6	0 / 6	1 / 6	3 / 6	0 / 6

TABLE 5.15: Taux de reconnaissance selon les différents modèles

propriété de dichotomie du treillis associé.

Pour cela, nous avons défini les concepts du treillis sur lesquels la discrétisation locale doit être menée, *i.e.* les inf-irréductibles. **Nous avons défini un algorithme de discrétisation locale pour les treillis de Galois, basé sur ces concepts spécifiques**. *Via* la propriété de coatomicité, cet algorithme est simple. **Nous avons adapté les critères de coupe à la structure particulière du treillis de Galois, permettant la prise en compte de l'impact du découpage d'un attribut dans un concept sur l'ensemble des concepts contenant cet attribut.**

Enfin, nous avons mené des expérimentations sur différents ensembles de données quantitatives. Ces expérimentations nous ont permis de démontrer qu'à l'inverse de l'arbre de classification, la discrétisation locale permet de diminuer la complexité structurelle des treillis de Galois. De plus, comme pour les arbres de classification, les performances des treillis sont améliorées par la discrétisation locale ou, dans le pire cas, diminuées de seulement 0.4% comparée à la discrétisation globale. **Le choix d'un treillis moins complexe même diminué de 0.4% en reconnaissance, nous paraît plus intéressant qu'un treillis beaucoup plus complexe avec des performances faiblement supérieures.**

Points clés

Positionnement

- ❏ Nous avons rappelé **le processus de discrétisation**.
- ❏ Nous avons présenté **la différence en discrétisation locale et discrétisation globale,** ainsi que les débats existants quant à leur utilisation pour la définition des arbres de classification.
- ❏ Nous avons rappelé la discrétisation en intervalles non disjoints définie par Ganter et Wille [Ganter 99] pour la définition de treillis de Galois. **Nous avons précisé notre choix pour la discrétisation en intervalles disjoints en raison de la complexité de la discrétisation en intervalles non disjoints liée à la taille des données utilisées, et irréalisable en pratique.**

Contributions

- ❏ **Nous avons défini un algorithme de discrétisation locale efficace pour les treillis de Galois.**
- ❏ Nous avons adapté la définition des critères de coupe à la structure de treillis.
- ❏ Nous avons mené des expérimentations permettant de mettre en avant l'impact structurel et pratique de la discrétisation locale sur les treillis de Galois.

Chapitre 6

Contribution 3 - Simplifications de la structure

6.1 Introduction

Dans leur livre très célèbre [Breiman 84], Breiman *et al.* affirment que les performances de l'arbre de classification sont liées à sa taille, *i.e.* il faut trouver l'arbre de bonne taille permettant un compromis entre ajustement à la base d'apprentissage et généralisation à de nouveaux exemples. La question peut se poser également pour le treillis de Galois, est-il plus performant qu'une sous-structure bien choisie issue de ce dernier ?

Ce chapitre s'intéresse donc à **la simplification du treillis construit après une discrétisation locale**. Comme nous l'avons déjà évoqué, l'un des avantages du treillis de Galois sur les arbres de classification est la multiplicité des chemins permettant d'atteindre un même concept terminal. Cela procure au treillis une robustesse accrue face aux données bruitées. Cependant, la complexité de la structure (nombre de concepts et de relations) est un défaut en termes de rapidité mais aussi de lisibilité par rapport à l'arbre de classification. En effet, bien que la discrétisation locale permette de diminuer la taille des treillis construits comparée à une discrétisation globale, ces treillis contiennent un nombre de concepts bien plus important que le nombre moyen de nœuds contenus dans les arbres de classification (*cf.* Figure 6.1.1 - axe vertical : échelle logarithmique).

Comme nous l'avons vu dans le chapitre 2, l'élagage est généralement utilisé pour les arbres de classification afin d'éviter la génération d'arbres trop complexes, *i.e.* contenant trop de nœuds, pour des raisons d'espace mémoire et de temps de calcul en classification, mais aussi et surtout pour éviter le phénomène de sur-apprentissage.

Rappelons que pour les arbres de classification, il existe deux modes d'élagage que sont le pré-élagage et le post-élagage. Le pré-élagage consiste à fixer un critère d'arrêt stoppant la construction du modèle (souvent un seuil est appliqué au critère de coupe). Le post-élagage, quant à lui, consiste, après la phase de construction du modèle complet,

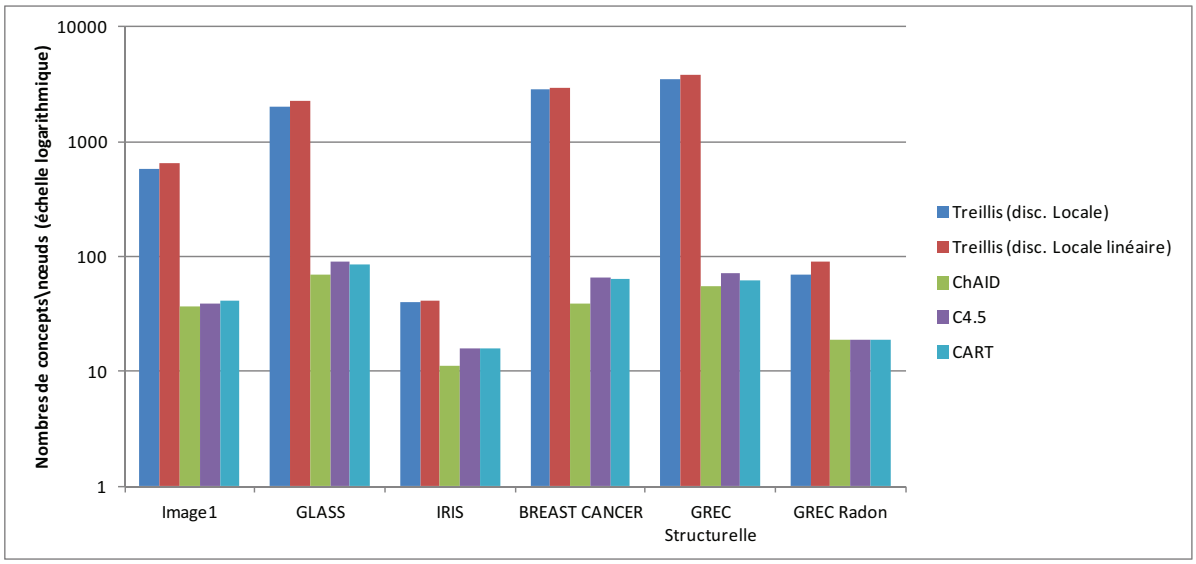


FIGURE 6.1.1: Comparaison nombres de concepts\nœuds par treillis et arbres sur les bases décrites dans la section 5.3.3

à supprimer des parties du modèle pour générer des sous-modèles de tailles différentes et choisir le meilleur, *i.e.* le modèle offrant le meilleur compromis entre taux d'erreur en classification et complexité structurelle. Le pré-élagage se fait généralement sur la base d'apprentissage, tandis que le post-élagage peut se faire sur une base de validation distincte de la base d'apprentissage et de la base de test, permettant ainsi d'estimer plus précisément la capacité de généralisation du modèle pour choisir le sous-arbre optimal (*cf.* section 2.4).

Ainsi **la discrétisation locale** telle que nous l'avons définie (*cf.* section 5.3) **peut induire un pré-élagage.** En effet, l'utilisation d'un critère d'arrêt permet de stopper la construction du contexte (et donc du treillis) en cours de génération. Dans notre cas, nous arrêtons la discrétisation lorsque le critère est nul cela indiquant soit que les sous-ensembles formés sont purs (*i.e.* ils contiennent des objets appartenant tous à la même classe), soit, lorsque les données ne sont pas séparables, qu'il n'y a plus de coupe pertinente pour séparer les sous-ensembles non purs. Cependant, en appliquant un seuil différent tel que proposé pour les arbres (*cf.* section 2.3.2, p 52), cela permettrait de limiter le nombre d'étapes, et donc induirait un pré-élagage.

En discrétisation globale, le même type de critère est utilisé comme critère d'arrêt, mais le modèle n'étant pas pris en compte puisque la discrétisation globale s'effectue en amont de la construction du modèle, on ne peut pas parler de pré-élagage.

Dans le cadre de la théorie des treillis, différentes études se sont intéressées à la simplification de la structure du treillis [Liquière 90, Anquetil 94, Godin 95a, Polaillon 98,

Ganter 99, Mineau 00, Kuznetsov 07b, Roth 08, Klimushkin 10, Falk 11]. Ces études sont principalement motivées par une diminution de la taille du treillis, qui peut être exponentielle en la taille des données utilisées, dans le pire des cas. Parmi ces travaux, certains ont pour but d'améliorer les performances du treillis lorsqu'il est utilisé en classification [Liquière 90, Godin 95a, Polaillon 98, Mineau 00, Kuznetsov 07b, Roth 08, Klimushkin 10, Falk 11].

La terminologie « d'élagage » étant généralement liée au phénomène de sur-apprentissage, et notre objectif n'étant pas seulement d'améliorer les performances de classification, nous privilégions la terminologie de « simplification de la structure » (aussi appelée *lattice filtering*).

Afin de réduire la taille du treillis construit tout en essayant de maintenir les performances en classification, nous nous intéressons donc à la mise en œuvre de ces simplifications de modèle.

Dans la section 6.2, **nous nous intéressons aux modèles d'élagage développés pour les arbres de classification, afin d'étendre aux treillis**, celui considéré comme le plus performant. Nous montrons que, du fait des spécificités structurelles du treillis, cette extension n'offre pas les résultats attendus et est notamment sujette à du sur-apprentissage. **Nous nous intéressons**, dans la section 6.3, **à une simplification des treillis déjà existante et basée sur l'indice de stabilité des concepts**. Comme nous le verrons, cette simplification est dépendante de la définition d'un seuil. Pour pallier à la difficulté de ce paramétrage, **nous proposons**, dans cette section 6.3, **un algorithme de définition automatique du seuil optimal**. Dans ces deux sections, nous appuyons nos propos sur différentes expérimentations.

6.2 Extension de l'élagage de type arbre aux treillis

6.2.1 Présentation et problématique

Rappelons que le principe général du post-élagage des arbres de classification est de construire une séquence de sous-arbres candidats, à partir de l'arbre maximal (*i.e.* arbre construit en appliquant des critères d'arrêt très permissifs au cours de la phase de division, *cf.* section 2.4). Puis de choisir, dans cette séquence de sous-arbres, celui qui offre les meilleures performances en classification selon différents critères. Les arbres élagués sont généralement construits en transformant en feuille un nœud interne de l'arbre à élaguer.

Comme mentionné dans le chapitre 2, selon Rakotomalala [Rakotomalala 05a], le post-élagage utilisé par CART est celui qui *intègre tous les "bons" ingrédients d'un apprentissage efficace,* en particulier il intègre *« une évaluation non biaisée de l'erreur pour déterminer le bon arbre ; une réduction de l'espace des hypothèses avec le principe des*

séquences d'arbres rangés à coût-complexité décroissant, limitant ainsi le risque de sur-apprentissage sur l'échantillon de validation ; une préférence donnée à la simplicité avec la règle de "l'écart-type" avant l'erreur minimale.

L'algorithme de post-élagage utilisé par CART (*CCP*) est basé sur **la minimisation de la mesure de coût-complexité**, notée σ (*cf.* section 2.4). Chaque étape construit un arbre DT_{i+1}, en transformant en feuille dans DT_i, les racines des branches ST_i minimisant σ. Rappelons que σ **est définie et calculée sur la base d'apprentissage** selon la formule suivante :

$$\sigma = \frac{\varepsilon(\tilde{DT}_i, BA) - \varepsilon(DT_i, BA)}{(L(ST_i) - 1)} \tag{6.2.1}$$

Avec un arbre à élaguer DT_i, dans lequel la racine de la branche ST_i est transformée en feuille pour obtenir le sous-arbre provisoirement élagué $\tilde{DT}_i$, et $L(ST_i)-1$ est la différence entre le nombre de feuilles de l'arbre avant élagage DT_i et l'arbre après élagage $\tilde{DT}_i$.

Le sous-arbre candidat DT_{i+1} est ajouté à la séquence, et l'opération est réitérée sur ce dernier. Une fois la séquence construite, **le meilleur sous-arbre** DT^* est choisi dans cette dernière, *i.e.* **celui minimisant un critère prenant en compte la taille de l'arbre et l'erreur sur une base de validation**. Ce critère est défini comme suit :

Le meilleur sous-arbre DT^*est celui possédant le moins de feuilles possibles et vérifiant la condition suivante :

$$\varepsilon(DT^*, BV) < \varepsilon^* + se(\varepsilon^*) \tag{6.2.2}$$

Avec

- $\varepsilon^* = \underset{DT_i \in \{DT_0, \ldots, t_1\}}{min} \varepsilon(DT_i, BV)$: l'erreur minimale observée pour les arbres de la séquence
- $se(\varepsilon^*)$: l'erreur standard associée à ε^*, $se(\varepsilon^*) = \sqrt{\frac{\varepsilon^* \times (1-\varepsilon^*)}{N}}$

Nous avons souhaité étendre ce procédé aux treillis de Galois. Cela consiste à transformer un à un les concepts prédécesseurs immédiats de concepts terminaux, pour obtenir une séquence de sous-structures, puis à choisir la meilleure sous-structure selon un critère prenant en compte la taille et les performances en classification.

Cependant, une différence importante existe entre la structure des arbres et celle des treillis. En effet, **dans un arbre, toute feuille a un unique père,** aussi transformer ce nœud interne (*i.e.* ce père) en feuille consiste à supprimer les nœuds et les branches fils de ce nœud. A contrario, **dans un treillis**, **un concept terminal peut avoir plusieurs pères.** Donc, transformer l'un de ces pères en concept terminal consiste à supprimer les arcs sortants de ce père. Pour autant, il n'est pas toujours possible de supprimer tous les concepts successeurs de ce père. En effet, ces derniers peuvent être atteignables par d'autres chemins de navigation. Les supprimer engendrerait une perdre

de la possibilité de navigation dans la structure (mais aussi de certaines propriétés des treillis dont la $\vee$-complémentarité).

Ainsi, il faut **adapter le processus de suppression des concepts en prenant en compte l'existence de chemins multiples** pour atteindre un même concept terminal. Pour ce faire, **nous proposons qu'un concept fils du concept à élaguer ne soit supprimé que lorsque tous ses pères ont été étiquetés comme terminaux,** *i.e.* lorsqu'il n'existe plus d'autre chemin de navigation pour atteindre ce concept.

Exemple. Prenons un exemple pour illustrer nos propos. Soit le treillis maximal de la Figure 6.2.1 (le contexte associé est présenté dans la Table 6.1), les concepts terminaux correspondent aux concepts en forme octogonale et verts. Supposons que l'on souhaite simplifier ce treillis en transformant le concept $C_1 = (\{o_6, o_7, o_8, o_{10}\}, \{I_4\})$ en concept terminal (ce dernier étant prédécesseur immédiat des concepts terminaux $C_2 = (\{o_6, o_7, o_8\}, \{I_2, I_4\})$ et $(\{o_{10}\}, \{I_3, I_4\})$). Sur la Figure 6.2.2, après transformation, C_1 est étiqueté avec sa classe majoritaire (*i.e.* la classe 3), les arcs sortants sont supprimés (représentés en pointillés gris).

Si le concept C_2 était supprimé, alors l'arc joignant ce dernier au concept $C_3 = (o_3, o_4, o_5, o_6, o_7, o_8\}, \{I_2\})$ (arc en tirets rouges dans la Figure 6.2.2) serait lui aussi supprimé. Cette suppression engendrerait la perte du second chemin permettant d'accéder à C_2 et donc un problème de navigation à partir de C_3 (en particulier pour les objets ne vérifiant pas le test de l'unique arc sortant de C_3 (arc plein et rouge)). En effet, rappelons que la navigation n'est stoppée que lorsqu'un concept terminal est atteint. Or, pour les objets ne vérifiant pas ce test, il n'y aurait pas de navigation possible après C_3, et donc la classification échouerait.

Pour éviter ce problème, le concept C_2 ne doit pas être supprimé. Seuls les arcs sortants de C_1 doivent l'être, tel qu'illustré sur la Figure 6.2.3. Le concept C_2 sera supprimé le cas échéant lorsque tous ses prédécesseurs auront été transformés en concepts terminaux.

Cette suppression est illustrée dans la Figure 6.2.4 où, lorsqu'un concept est supprimé, son étiquette est barrée en rouge, son contour est en rouge et il n'est relié à aucun autre concept. Sur cet exemple, lorsque le concept C_4 est transformé en concept terminal, il n'existe plus de chemin menant au concept C_5, ce dernier est donc supprimé.

Dés lors, la mesure de coût-complexité ne peut être appliquée telle quelle (*i.e.* simplement en remplaçant les feuilles par les concepts terminaux dans l'équation 6.2.1), car la différence entre le nombre de concepts terminaux du treillis avant simplification et celui du treillis simplifié peut être négative (*i.e.* si aucun fils d'un nouveau concept terminal n'est supprimé, alors le nombre de concepts terminaux augmente) ou être nulle (*i.e.* si un seul fils est supprimé, alors le nombre de concept terminaux reste le même).

Exemple. Pour illustrer ce phénomène propre aux treillis, reprenons l'exemple du treillis complet précédent. Les Figures 6.2.3 et 6.2.4 présentent deux simplifications possibles. Le nombre de concepts terminaux est indiqué sur chacune d'elle.

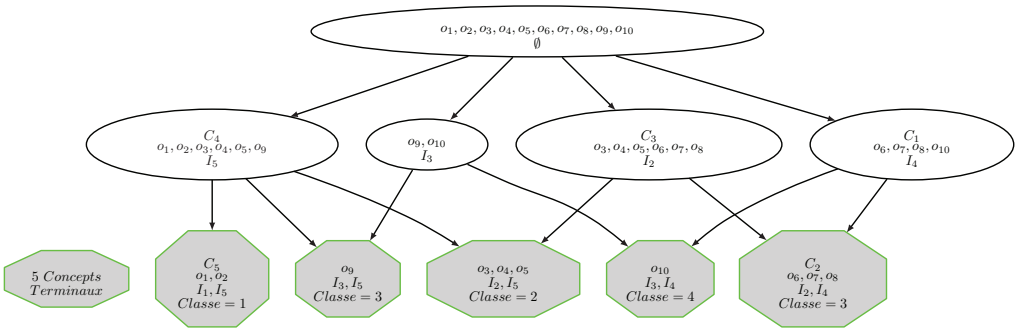


FIGURE 6.2.1: Treillis avant simplification

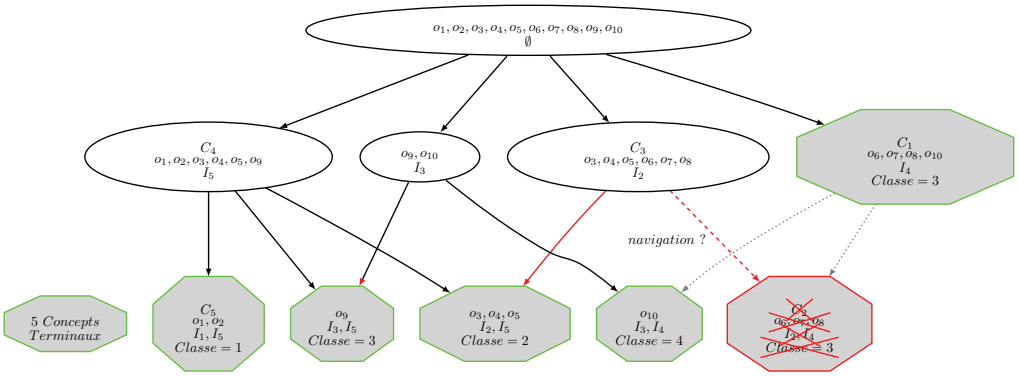


FIGURE 6.2.2: Treillis incorrectement simplifié engendrant une perte de la navigation

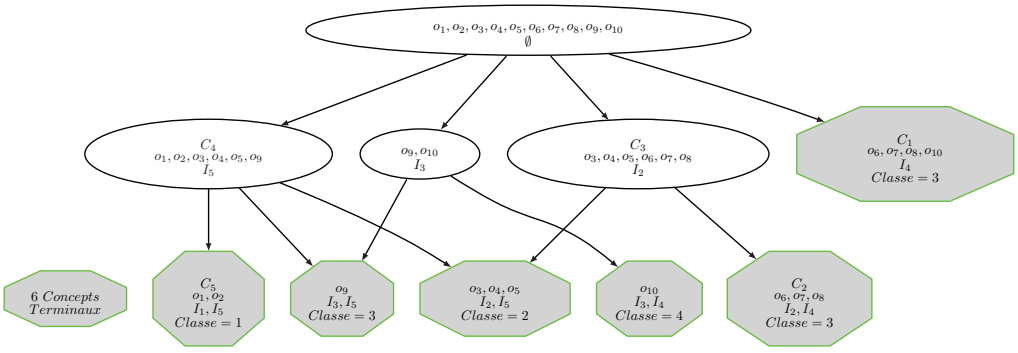


FIGURE 6.2.3: Treillis simplifié conservation des concepts terminaux et de la navigation

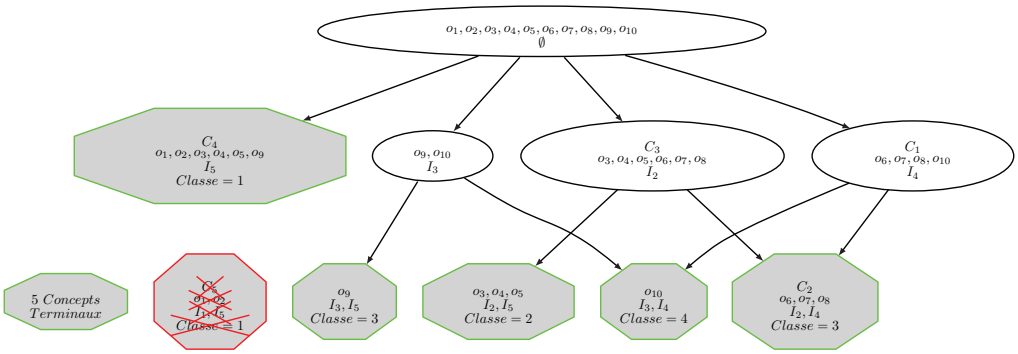


FIGURE 6.2.4: Treillis simplifié suppression d'un concept terminal et conservation de la navigation

O \ I	I_1	I_2	I_3	I_4	I_5	K
o_1	×			×		1
o_2	×			×		
o_3		×		×		2
o_4		×		×		
o_5		×		×		
o_6		×			×	3
o_7		×			×	
o_8		×			×	
o_9			×	×		
o_{10}			×		×	4

TABLE 6.1: Contexte formel du treillis de la Figure 6.2.1

La Figure 6.2.3 présente le cas où le nombre de concepts terminaux augmente entre le treillis complet et le treillis simplifié. Ce cas se présente lorsque le concept transformé est prédécesseur d'un concept terminal encore rattaché à un autre concept interne.

La Figure 6.2.4 **présente le cas où le nombre de concepts terminaux est stationnaire** entre le treillis complet et le treillis simplifié. Ce cas se présente lorsque le concept transformé est l'unique prédécesseur d'un concept terminal.

Pour pallier ce problème, nous pourrions envisager de considérer non pas l'évolution du nombre de concepts terminaux, mais l'évolution du nombre de concepts dans le treillis (*i.e.* sa taille). Malheureusement, cela ne change rien au problème présenté précédemment, si aucun concept n'est supprimé, comme c'est le cas sur l'exemple de la Figure **6.2.3**, alors la différence sera, là encore, nulle.

Ainsi la mesure de coût-complexité doit être adaptée. Nous proposons une solution dans la section suivante.

6.2.2 Proposition et adaptation

Comme nous l'avons vu précédemment, en choisissant de supprimer un concept uniquement lorsqu'il n'est plus relié à aucun concept interne non terminal, cela induit une modification de l'évolution du nombre de concepts terminaux comparée à l'évolution du nombre de feuilles lors de l'élagage d'un arbre. De ce fait, il est **nécessaire d'adapter la mesure de coût-compléxité**.

Aussi pour contourner ce problème, **nous modifions la manière dont le coût et la complexité sont pris en compte** lors de la simplification. Cette modification

présentée dans l'Algorithme 6.1 consiste à **considérer uniquement la partie coût pour la construction de la séquence de treillis, puis à considérer ensuite le coût et la complexité pour le choix du meilleur sous-treillis**. Ainsi la structure est simplifiée selon le procédé suivant :

1. **Construction d'une séquence de sous-treillis par minimisation du taux d'erreur en classification sur la base d'apprentissage (coût)**.
2. Choix du **meilleur** sous-treillis dans la séquence, **par minimisation d'un critère prenant en compte le nombre de concepts de la structure (complexité) et le taux d'erreur en classification sur une base de validation** (qui peut être la base d'apprentissage) **(coût).**

L'Algorithme 6.1 utilise la base d'apprentissage BA et une base de validation BV (qui peut être la base d'apprentissage). Notons que la première étape consiste à appliquer l'Algorithme 3.5, qui correspond à la simplification habituelle qui supprime tous les concepts non atteignables par navigation, ainsi que tous les arcs inutiles.

Ensuite, les prédécesseurs des concepts terminaux du treillis courant $\mathcal{L}_i$, sont étiquetés, un à un et de manière provisoire, comme terminaux. A chaque prédécesseur (A_i, B_i) transformé provisoirement, correspond un treillis simplifié $\tilde{\mathcal{L}_{i+1}}$, pour lequel le taux d'erreur est calculé sur la base d'apprentissage, *i.e.* $\varepsilon(\tilde{\mathcal{L}_{i+1}}, BA)$. Le prédécesseur (A_i, B_i), permettant de minimiser l'erreur sur la base d'apprentissage $\varepsilon(\tilde{\mathcal{L}_{i+1}}, BA)$, est conservé en mémoire, pour être ensuite définitivement transformé en concept terminal. Le treillis simplifié correspondant $\mathcal{L}_{i+1}$ est ajouté à la séquence, puis l'opération est réitérée sur ce dernier.

Lorsque le treillis courant $\mathcal{L}_i$ ne contient plus que le concept minimal, il n'y a plus de simplification possible, *i.e.* la séquence est complète. L'étape suivante consiste à choisir dans cette séquence, la meilleure structure simplifiée $\mathcal{L}^*$, *i.e.* celle ayant le plus faible taux d'erreur, tout en ayant le moins de concepts possible.

Il n'est pas nécessaire de mémoriser la séquence de structures simplifiées entièrement. C'est pourquoi dans l'Algorithme 6.1 seule la meilleure structure simplifiée, déjà calculée, est sauvegardée dans $\mathcal{L}^*$**.**

Remarque. Notons que l'utilisation de l'erreur simple lors du choix du concept à transformer au cours d'une étape, peut être rapprochée du critère *PEP* de Quinlan qui utilise uniquement le taux d'erreur pessimiste sur la base d'apprentissage et sans prise en compte de la taille de la structure [Quinlan 93] (*cf.* section 2.4). Cependant, à l'inverse des arbres de classification, le taux d'erreur n'évolue pas de la même manière au cours des étapes, puisqu'il n'est pas décroissant.

Ce phénomène propre aux treillis est illustré par la Figure 6.2.5 où l'évolution du taux d'erreur au cours des étapes de simplification est donnée en fonction du nombre de concepts de la structure (ce graphique est un extrait de la simplification du treillis construit après une discrétisation locale linéaire appliquée à la base de données Image1,

Algorithme 6.1 *Simplification_treillis_Coût_Complexité($\mathcal{L}$,BA,BV)*

Entrées :
- $\mathcal{L} = (\mathcal{C}, \prec)$ le treillis réduit à simplifier, *i.e.* le diagramme de Hasse d'un treillis ;
- BA la base d'apprentissage ;
- BV la base de validation ;

Sorties :
- $\mathcal{L}^*$ le treillis simplifié ;

DÉBUT
//Suppression des arcs et noeuds inutiles selon l'Algorithme 3.5, p 106
$Simplifier\_après\_Concepts\_Terminaux(\mathcal{L})$;

$taille\_ref \leftarrow |\mathcal{L}|$;
$taux\_ref \leftarrow \varepsilon(\mathcal{L}, BV)$;
//Simplification du treillis jusqu'à ce qu'il ne contienne plus que $\perp$
$\tilde{\mathcal{L}} \leftarrow \mathcal{L}$;
tant que $\tilde{\mathcal{L}} \neq \perp$ **faire**
  $pred\_terminal \leftarrow$ ensemble des prédécesseurs des concepts terminaux de $\tilde{\mathcal{L}}$;
  $taux\_min \leftarrow +\infty$;
  $concept\_à\_transformer \leftarrow \emptyset$;
  //Évaluation de l'apport de la transformation d'un prédécesseur d'un concept terminal
  pour chaque $pred\_courant \in pred\_terminal$ **faire**
    $treillis\_courant \leftarrow \tilde{\mathcal{L}}$ avec $pred\_courant$ étiqueté comme concept terminal ;
    //Mémorisation du prédécesseur permettant de minimiser l'erreur sur la base d'apprentissage
    si $\varepsilon(treillis\_courant, BA) < taux\_min$ **alors**
      $taux\_min \leftarrow \varepsilon(treillis\_courant, BA)$;
      $concept\_à\_transformer \leftarrow pred\_courant$;
    fin si
  fin pour
  //Transformation du meilleur prédécesseur
  Étiqueter $concept\_à\_transformer$ comme concept terminal dans $\tilde{\mathcal{L}}$;
  Supprimer, dans $\tilde{\mathcal{L}}$, les relations entre $concept\_à\_transformer$ et ses successeurs ;
  //Suppression des concepts qui ne sont plus liés à aucun autre
  Supprimer, dans $\tilde{\mathcal{L}}$, les concepts non atteignables ;
  //Mémorisation à la volée du meilleur sous-treillis $\mathcal{L}^*$
  si $|\tilde{\mathcal{L}}| \leq taille\_ref$ et $taux\_min \leq taux\_ref$ **alors**
    $\mathcal{L}^* \leftarrow \tilde{\mathcal{L}}$;
    $taille\_ref \leftarrow |\mathcal{L}^*|$;
    $taux\_ref \leftarrow taux\_min$;
  fin si
fin tant que
renvoyer $\mathcal{L}^*$;
FIN

avec pour critère de coupe et d'arrêt le χ^2, les taux d'erreur sont ceux observés sur la base d'apprentissage). Nous pouvons voir que les structures simplifiées contenant plus de 432 concepts ont un taux d'erreur nul, comme les plus petites contenant entre 263 et 258 concepts, tandis que celles contenant entre 432 et 263 concepts ont un taux d'erreur non nul. En effet, la suppression d'un chemin correct c_1 peut engendrer des erreurs de classement *via* l'utilisation d'un autre chemin c_2 menant à une classe différente, et ces erreurs peuvent disparaitre lorsque c_2 est supprimé au cours d'une nouvelle étape.

En suivant le modèle *REP* (*cf.* section *2.4)*, la séquence aurait été construite partiellement jusqu'à la structure contenant 432 concepts, alors qu'une structure plus petite permet d'avoir le même taux d'erreur. D'où la nécessité de construire la séquence entière pour le traitement des treillis.

Un autre point distingue notre approche de la méthode *REP*. En effet *REP* ne prend pas en compte la complexité de la structure or si celle-ci était considérée, cela permettrait de choisir la plus petite structure parmi celles ayant le même taux d'erreur.

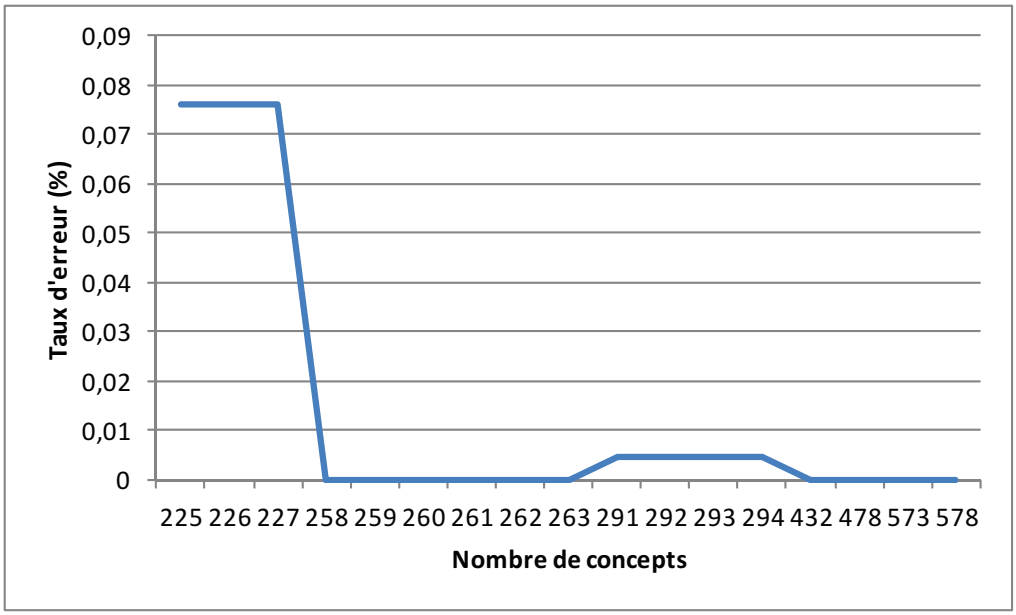


FIGURE 6.2.5: Évolution du taux d'erreur en fonction du nombre de concepts

Exemple. Pour illustrer le processus proposé, appliquons l'Algorithme 6.1 au treillis de la Figure 6.2.6 dont le contexte formel est présenté dans la Table 6.2. Nous appelons l'Algorithme 6.1 en utilisant comme base de validation ce même ensemble de données (*i.e.* la base d'apprentissage). Les différentes structures simplifiées construites au fur et à mesure des étapes sont présentées dans les Figures 6.2.7, 6.2.8, 6.2.9, 6.2.10, et 6.2.11. Les concepts internes sont progressivement (*i.e.* étape par étape) transformés en concepts terminaux, jusqu'à ce que la structure ne contienne plus que le concept minimal.

Dans cette séquence de structures simplifiées, c'est **celle obtenue lors de l'étape 2** qui **est choisie comme étant la meilleure**. La Table 6.3 présentent l'évolution du nombre de concepts de la structure simplifiée au fur et à mesure des étapes ainsi que le taux d'erreur associé à chacune d'entre elles. La structure simplifiée choisie lors de l'étape 2 est sélectionnée car elle a un taux d'erreur nul, *i.e.* le plus faible taux d'erreur possible et elle a le plus petit nombre de concepts parmi les structures obtenant le même taux d'erreur.

	Attributs - $\mathcal{I}$				K
O	I_1	I_2	I_3	I_4	
o_1		×		×	1
o_2		×		×	
o_3		×	×		2
o_4	×		×		3
o_5	×		×		
o_6	×			×	4

TABLE 6.2: Contexte binaire du treillis de la Figure 6.2.6

Treillis	Objets mal classés	Taux d'erreur	Nb de concepts	Meilleure structure ?
Maximal	$\emptyset$	0%	10	
Étape 1	$\emptyset$	0%	9	
Étape 2	$\emptyset$	0%	8	×
Étape 3	o_6	16%	7	
Étape 4	o_3, o_6	33%	5	
Étape 5	o_1, o_2, o_3, o_6	66%	1	

TABLE 6.3: Évolution par étape du nombre de concepts et du taux d'erreur

6.2.3 Expérimentations

Nous avons réalisé deux types d'expérimentations avec cet algorithme de simplification. Les premières utilisent une base de validation indépendante de la base d'apprentissage tandis que les secondes utilisent la base d'apprentissage comme base de validation.

Base de validation indépendante de la base d'apprentissage

Dans ces expérimentations, les bases d'apprentissage utilisées précédemment (chapitre 4) ont été divisées pour former une base d'apprentissage et une base de validation indépendantes selon le ratio 2/3-1/3. Les deux bases formées sont constituées de **données**

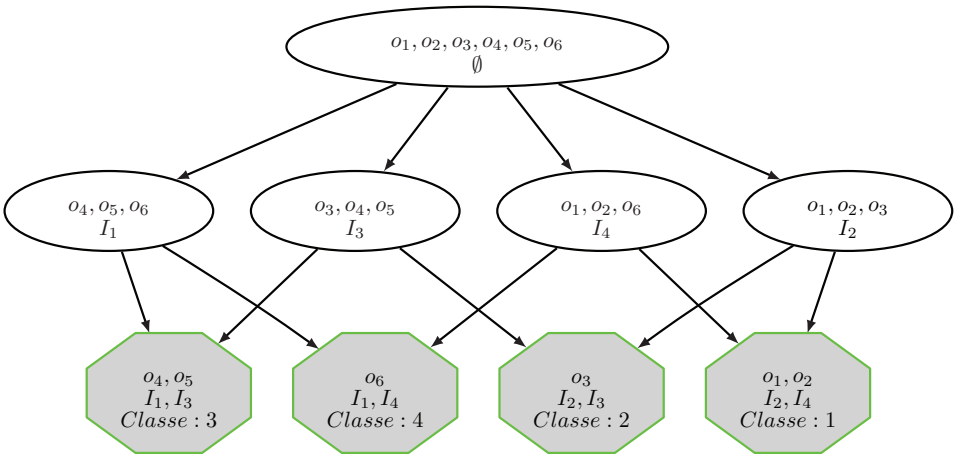


FIGURE 6.2.6: Treillis maximal

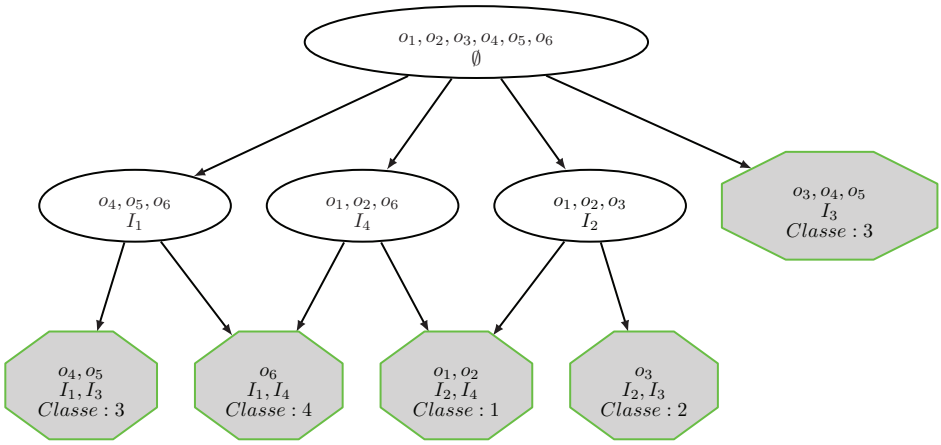


FIGURE 6.2.7: Étape 1 - Treillis simplifié

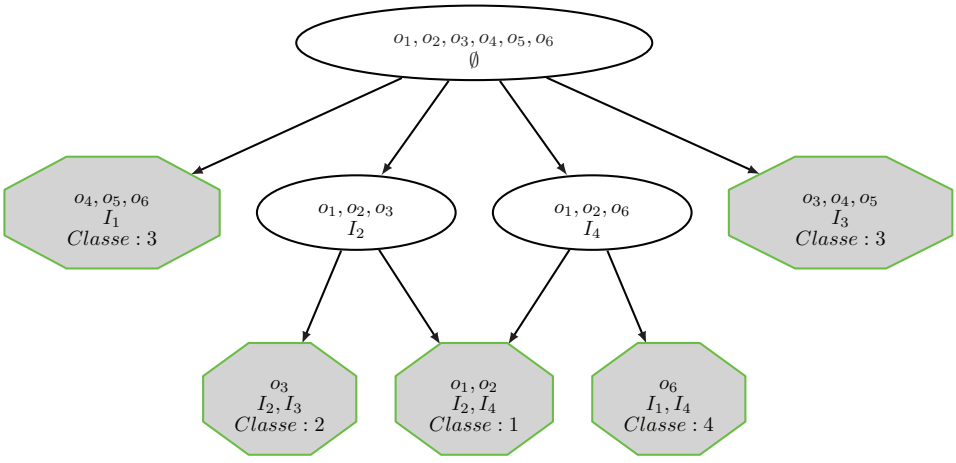


FIGURE 6.2.8: Étape 2 - Treillis simplifié

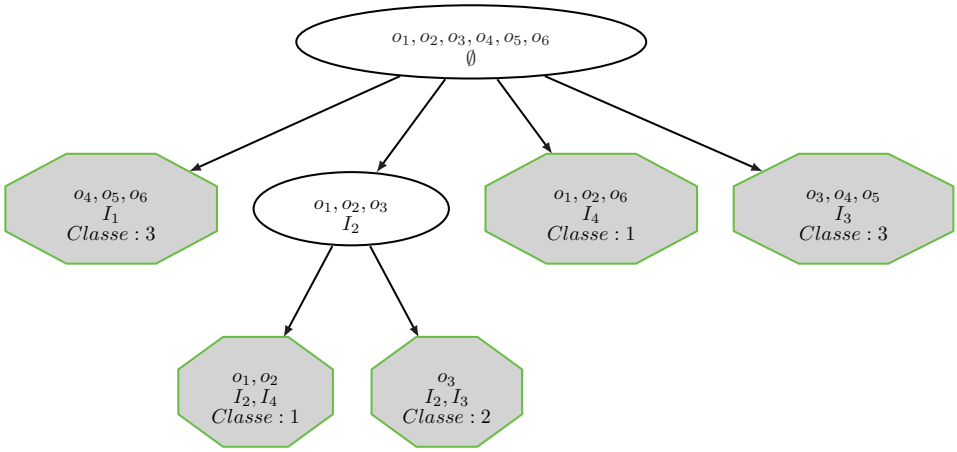


FIGURE 6.2.9: Étape 3 - Treillis simplifié

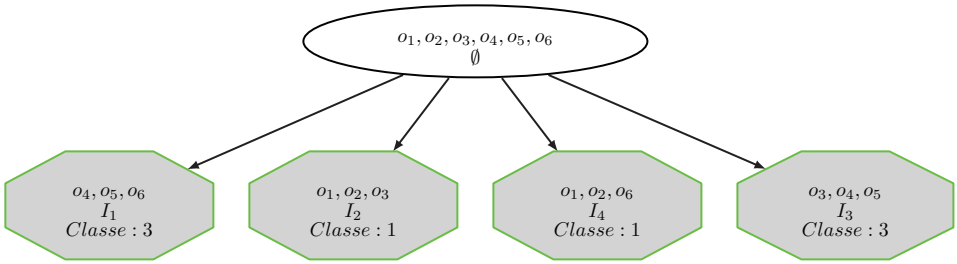


FIGURE 6.2.10: Étape 4 - Treillis simplifié

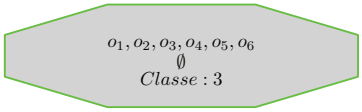


FIGURE 6.2.11: Étape 5 - Treillis simplifié

indépendantes et identiquement distribuées.

La Table 6.4 présente les taux de reconnaissance [1] calculés sur la base de test (associée à chaque ensemble de données) à partir des treillis construits après une discrétisation locale linéaire, sans simplification de leur structure, et avec simplification de leur structure, selon l'Algorithme 6.1. Nous rappelons dans la dernière colonne les taux de reconnaissance obtenus sur la même base de test avec les treillis construits à partir des bases d'apprentissage complètes (*cf.* section 5.3.3).

	Treillis de Galois (discrétisation locale linéaire)		
	BA et BV indépendantes		BA complète
Ensemble de données	Sans simplification	Après simplification	Sans simplification
Image1	87,61	86,38	**91,71**
GLASS	65,36 ±14,23	61,83 ±12,31	**71,09 ±10,53**
IRIS	96,00 ±4,42	**96,67 ±3,34**	95,33 ±5,21
BREAST CANCER	**94,45 ±3,12**	93,88 ±3,19	94,43 ±2,05
GREC structurelle	68,74	68,09	**73,68**
GREC Radon	87,75 ±1,39	86,82 ±2,10	**90,69 ±4,33**
Moyenne	83,29	82,28	**86,15**

TABLE 6.4: Taux de reconnaissance avec et sans simplification de la structure

Nous pouvons voir que la simplification proposée permet d'améliorer les performances en classification seulement pour Iris. Pour les autres bases, une diminution des performances en généralisation est observée. Ainsi, bien que la diminution du taux de reconnaissance soit en général faible, **la simplification réduit les performances en généralisation des treillis.**

Le point le plus important, observable sur ces expérimentations, est que l'utilisation d'une base d'apprentissage réduite engendre la construction de treillis moins performants que ceux définis avec une base d'apprentissage complète (exception faite à 0,02 % de BREAST CANCER). Cela s'explique facilement par la diminution du nombre d'exemples appris. **Le processus de simplification de la structure, utilisant la base de validation indépendante, ne permet pas de pallier à cette diminution.**

Ces observations sont proches de celles faites par Quinlan qui indiquait dans [Quinlan 87b], que l'utilisation d'une base de validation indépendante pouvait être contraignante, *i.e.* faire diminuer les performances en classification.

1. La variance des taux de reconnaissance est indiquée pour les bases où une validation croisée est utilisée.

Sur ce constat, nous avons choisi d'utiliser dans de nouvelles expérimentations, les bases d'apprentissage complètes et de procéder à la simplification de la structure à partir de ces mêmes bases (*i.e.* BV=BA).

Base de validation égale à la base d'apprentissage

Sur la même démarche que Quinlan (critère *EBP* [Quinlan 93]), nous avons choisi d'utiliser la base d'apprentissage complète pour la construction du treillis maximal, mais aussi comme base de validation pour sa simplification. Cela nous permet de construire le treillis maximal à partir de plus d'exemples. Ces expérimentations ont été menées avec l'Algorithme 6.1, et en paramètre BV = BA.

La Table 6.5 présente les taux de reconnaissance obtenus. La Table 6.6 présente le nombre de concepts du treillis maximal (*i.e.* sans simplification), celui du treillis simplifié (*i.e.* après simplification) ainsi que le pourcentage de concepts supprimés.

	Treillis de Galois (discrétisation locale linéaire)	
	BA complète	
Ensemble de données	Sans simplification	Après simplification
Image1	**91,71**	81,33
GLASS	**71,09 ±10,53**	59,78 ±10,23
IRIS	**95,33 ±5,21**	94,66 ±0,7
BREAST CANCER	**94,43 ±2,05**	93,72 ±2,54
GREC structurelle	**73,68**	67,57
GREC Radon	**90,69 ±4,33**	87,16 ±7,20
Moyenne	86,15	80,70

Table 6.5: Taux de reconnaissance avec et sans simplification de la structure

	Treillis de Galois (discrétisation locale linéaire)		
	BA complète		
Ensemble de données	Sans simplification	Après simplification	Différence
Image1	649	**448**	- 31 %
GLASS	2267	**1129**	- 50 %
IRIS	41	**35**	- 15 %
BREAST CANCER	2961	**679**	- 77 %
GREC structurelle	3851	**1120**	- 71 %
GREC Radon	90	**81**	- 10 %

Table 6.6: Nombre de concepts avec et sans simplification de la structure

Nous pouvons voir dans la Table 6.6 que la simplification proposée permet de diminuer la taille de la structure de manière intéressante avec au minimum 10% de concepts

supprimés et jusqu'à 77%. Cependant, **la simplification réduit les performances en généralisation des treillis.**

6.2.4 Discussion

Les expérimentations menées avec l'Algorithme 6.1 montrent que lors de l'utilisation d'une base de validation indépendante de la base d'apprentissage, **la simplification ne permet pas de compenser la diminution du taux de reconnaissance entre le treillis défini à partir de la base d'apprentissage complète et le treillis défini avec un sous-ensemble de celle-ci**. Ensuite, lors de l'utilisation de la base d'apprentissage complète pour les deux étapes d'apprentissage et de simplification, nous avons pu observer que la simplification génère des structures simplifiées moins performantes en généralisation que les treillis maximaux. **La simplification ne supprimant pas toujours de concept d'une étape à une autre, mais uniquement des arcs, la robustesse du treillis est réduite.**

Ainsi, avec une diminution intéressante de la taille des treillis, les taux de reconnaissance obtenus avec ces treillis simplifiés sont plus faibles que ceux obtenus avec les treillis complets. Nous pouvons en conclure que **le processus ascendant de suppression de concepts n'est pas adapté à la structure du treillis, *i.e.* certains concepts ou chemins sont supprimés malgré leur apport en classification.**

Ce constat nous a orientés vers des indices propres à la théorie des treillis. Ces indices sont des indicateurs de pertinence des concepts du treillis. A l'inverse du taux de reconnaissance qui est un critère qui pourrait être qualifié de **critère global**, ces indices s'appliquent à chaque concept ; de fait ils peuvent être qualifiés de **critères locaux**.

De plus, il existe une distinction entre les différents indices locaux existants, certains sont **monotones** et d'autres sont **non-monotones**. Un indice est dit monotone lorsque sa valeur décroit d'un concept à ses successeurs sinon il est dit non-monotone. L'utilisation d'**un indice monotone induit** des treillis spécifiques que sont les sup-demi-treillis (*i.e.* treillis où toute paire d'éléments admet une borne supérieure ; *cf.* section 3.2.2, p 78), un concept interne n'étant jamais supprimé si ses successeurs ne l'ont pas été, il s'agit d'**une simplification ascendante**. Tandis que l'utilisation d'**un indice non-monotone permet de supprimer des concepts internes sans supprimer leurs successeurs**. Dans ce cas, la structure simplifiée obtenue ne vérifie pas toujours la propriété de treillis. [Roth 06, Kuznetsov 07b, Roth 08, Klimushkin 10].

Notre adaptation de la mesure de coût-complexité peut être considérée comme une mesure monotone car, bien que ce soit une mesure globale, la simplification associée permet d'obtenir des sup-demi-treillis.

Dans la section suivante, après une présentation générale des indices propres aux treillis, nous nous focalisons sur un indice non-monotone, qui permet une simplification différente de la précédente. Puis, pour pallier au paramétrage (toujours complexe) de la

simplification basée sur cet indice, nous proposons un processus de simplification mêlant cet indice avec le critère global de performance du treillis en classification. Enfin, nous présentons les expérimentations démontrant l'apport de cette méthode.

6.3 Indices locaux pour la simplification des treillis de Galois

6.3.1 Présentation et problématique

Comme mentionné dans le chapitre 3, certaines méthodes de classification basées sur les treillis de Galois n'utilisent pas la structure complète du treillis, mais seulement une sélection de concepts. Cette sélection est généralement réalisée selon une mesure de pertinence des concepts du treillis. Dans la littérature, différentes mesures, généralement statistiques, ont été développées et en particulier des **indices locaux**. Parmi les plus connus, se trouvent les indices de stabilité, de séparation et de probabilité [Kuznetsov 90, Kuznetsov 07a, Klimushkin 10], mais aussi l'indice de support [Liquière 90, Brin 97, Stumme 01]. Tous ces indices sont définis de manière non supervisée, leur valeur dépend du contenu de chaque concept (*i.e.* de leur intension et de leur extension), l'appartenance des objets à une classe n'étant pas prise en compte.

L'indice de support a été utilisé pour définir des treillis iceberg (*i.e.* sup-demi-treillis de concepts fréquents ; *cf.* section 3.2.2) [Stumme 01]. Il s'agit d'**un critère monotone**, car le support (nombre d'objets du concept) décroît avec la relation $\leq$.

Les indices de stabilité, de séparation et de probabilité ont été utilisés pour supprimer des concepts particuliers du treillis tout en préservant sa structure arborescente [Roth 06, Kuznetsov 07b, Roth 08, Klimushkin 10, Falk 11]. **L'indice de stabilité** (*cf.* définition 210)**, en particulier, est un indice non-monotone.**

De manière générale, les simplifications basées sur ces indices consistent à **définir un seuil**, et à **supprimer les concepts dont l'indice est inférieur à ce seuil**.

Nous avons vu dans la section précédente que l'adaptation de la mesure de coût-complexité, mesure monotone, n'est pas efficace. La monotonie n'étant pas indispensable pour une classification par navigation, **nous nous sommes donc intéressés plus particulièrement à l'indice de stabilité, ce dernier étant non-monotone**.

6.3.1.1 L'indice de stabilité

Définition

L'indice de stabilité des concepts formels a été défini par Kuznetsov dans le cadre de la théorie des treillis [Kuznetsov 90, Kuznetsov 03, Kuznetsov 07a]. Sa définition est la

suivante :

Définition 19. Soit un concept $(A, B) \in \mathcal{C}$ d'un treillis de Galois $\mathcal{L} = (\mathcal{C}, \leq)$, l'indice de stabilité de ce concept est défini par :

$$\delta((A, B)) = \frac{|\{\tilde{A} \subseteq A \ tq \ \alpha(\tilde{A}) = B\}|}{2^{|A|}} \tag{6.3.1}$$

Cet indice traduit la proportion de sous-ensembles $\tilde{A} \subseteq A$ qui ont pour intension exactement B, c'est à dire la dépendance entre les attributs binaires et les objets contenus dans un concept. Plus un concept est stable, *i.e.* plus son indice de stabilité est élevé, moins il est dépendant d'un objet particulier qu'il contient [Roth 06].

Il existe une définition duale de la stabilité des concepts à partir de leur intension (ensemble d'attributs) plutôt que de leur extension (ensemble d'objets) [Kuznetsov 07a, Kuznetsov 07b, Roth 08]. **Nous nous intéressons plus particulièrement à la stabilité sur l'extension.** En effet, la navigation dans le treillis a pour objectif d'atteindre un concept terminal, *i.e.* un sous-groupe d'objets proches. De plus, un concept qui contient des objets bruités risque d'avoir un indice de stabilité sur les objets (*i.e.* sur l'extension) faible, car la suppression de l'un de ces objets a de forte chance de modifier l'intension du concept. Cette information nous intéresse donc tout particulièrement. Aussi, c'est la stabilité sur les objets et donc sur les attributs qui est la plus pertinente pour nous.

Algorithmes

Algorithme de calcul des indices de stabilité

Le calcul de la stabilité d'un concept (A, B) nécessite de tester toutes les sous-parties de A. Dans son article [Kuznetsov 07a], Kuznetsov a démontré que ce calcul est NP-complet **avec un contexte en entrée**. Pour pallier ce problème, il propose une heuristique polynomiale calculant approximativement l'indice de stabilité d'un concept à partir du contexte formel.

Dans [Roth 06], Roth *et al.* proposent un algorithme exact et polynomial, de calcul de tous les indices de stabilité des concepts d'un treillis, **en s'appuyant sur le diagramme de Hasse d'un treillis en entrée**. Cet algorithme a été utilisé de manière efficace pour la classification non supervisée [Falk 11], nous l'avons donc utilisé dans nos expérimentations (*cf.* section 6.3.3).

L'Algorithme 6.2 présente cet algorithme qui prend en paramètre le diagramme de Hasse d'un treillis $(\mathcal{C}, \prec)$, puis qui calcule l'indice de stabilité des concepts de la structure. Le calcul de l'indice de stabilité d'un concept est fonction des sous-ensembles d'objets existants dans ses successeurs.

Exemple. En appliquant cet Algorithme 6.2 au treillis présenté précédemment (*cf.* Figure 6.2.6), on obtient le treillis marqué par les indices de stabilité de chaque concept,

Algorithme 6.2 *Calcul_Indices_Stabilité($\mathcal{L}$)*

Entrées :
- $\mathcal{L} = (\mathcal{C}, \prec)$ le diagramme de Hasse d'un treillis ;

Sorties :
- $(\mathcal{C}, \prec, \Delta)$ le diagramme de Hasse du treillis marqué avec les indices de stabilité ;

DÉBUT
//Initialisation
pour chaque $(A, B) \in \mathcal{C}$ **faire**
  $sous\_ensembles[(A, B)] = 2^{|A|}$;
fin pour
$liste\_concepts \leftarrow$ tri topologique de $\mathcal{C}$ selon $\prec$;
//Calcul des indices en plusieurs étapes
tant que $liste\_concepts \neq \emptyset$ **faire**
  Soit (C, D) le premier concept de $liste\_concepts$;
  //Calcul de l'indice de stabilité de chaque concept en fonction du nombre de sous-ensembles d'objets vérifiant exactement D ;
  $\delta[(C, D)] = \frac{sous\_ensembles[(C,D)]}{2^{|C|}}$;
  Marquer (C, D) avec son indice $\delta[(C, D)]$;
  Supprimer (C, D) de $liste\_concepts$;
  pour chaque $(A, B) < (C, D)$ **faire**
    $sous\_ensembles[(A, B)] = sous\_ensembles[(A, B)] - sous\_ensembles[(C, D)]$;
  fin pour
fin tant que

renvoyer $(\mathcal{C}, \prec, \Delta)$ marqué ;
FIN

présenté dans la Figure 6.3.1.1. Pour faciliter la présentation des étapes de calcul dans la Table 6.7, les étiquettes des concepts de cette figure ont été modifiées. Ainsi, l'ensemble des attributs et la classe majoritaire des concepts terminaux ont été retirés, les étiquettes contiennent un identifiant, l'ensemble des objets et l'indice de stabilité. De plus, afin d'observer que l'indice de stabilité du concept maximal est égal à 1, nous l'avons représenté sur cette figure. La Table 6.7 présente l'évolution étape par étape, de l'Algorithme 6.2 permettant de calculer les indices de stabilité δ pour chaque concept. La variable sous-ensemble est tout d'abord initialisée avec la valeur $2^{|A|}$ pour chaque concept (A, B) du treillis. Puis lorsque ces variables sont modifiées, elles sont indiquées en gras, *i.e.* à chaque fois que l'indice de stabilité est calculé pour l'un des successeurs du concept. Les lignes grises indiquent les valeurs des indices de stabilité pour chaque concept.

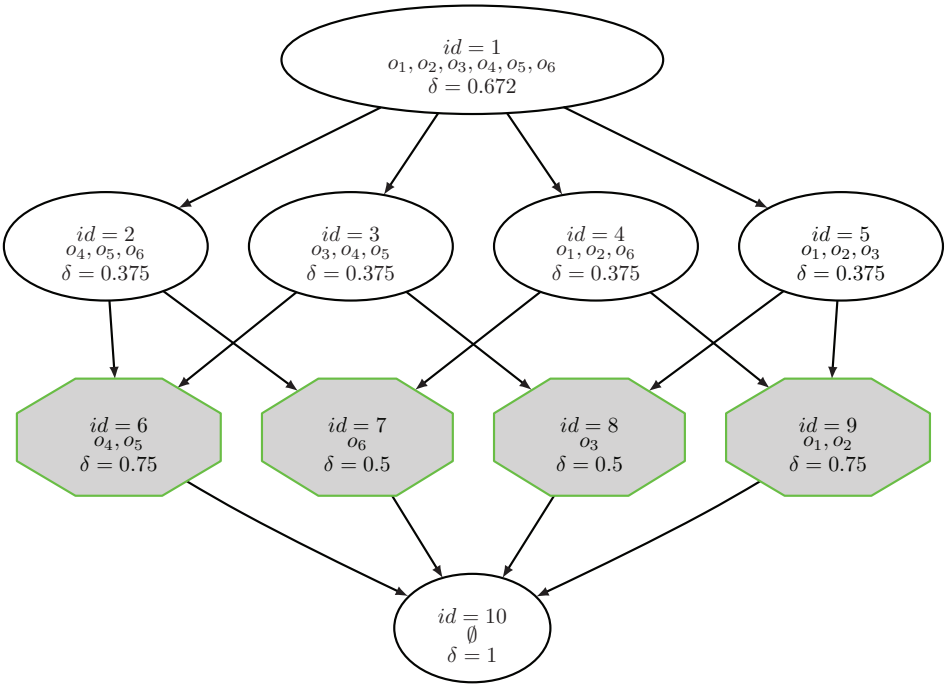


FIGURE 6.3.1: Indice de stabilité des concepts du treillis de la Figure 6.2.6 (construit à partir du contexte de la Table 6.2)

Algorithme de simplification basée sur l'indice de stabilité

Comme mentionné précédemment, l'indice de stabilité a été utilisé pour simplifier la structure de treillis tout en conservant une structure arborescente [Kuznetsov 07b, Roth 08]. L'algorithme correspondant est présenté dans l'Algorithme 6.3.

Cette simplification nécessite la définition d'un seuil η sur la stabilité et est appliquée au diagramme de Hasse d'un treillis marqué par ses indices de stabilité $\mathcal{L} = (\mathcal{C}, \prec, \Delta)$.

ID concept	10	7	8	9	4	5	6	3	2	1
Initialisation										
sous_ensembles ($2^{\|A\|}$)	1	2	2	4	8	8	4	8	8	64
Étape 1 - traitement du concept 10										
sous_ensembles	**1**	**1**	**1**	**3**	**7**	**7**	**3**	**7**	**7**	**63**
δ	**1**									
Étape 2 - traitement du concept 7										
sous_ensembles	1	1	1	3	**6**	7	3	7	**6**	**62**
δ	1	**0.5**								
Étape 3 - traitement du concept 8										
sous_ensembles	1	1	1	3	6	**6**	3	**6**	6	**61**
δ	1	0.5	**0.5**							
Étape 4 - traitement du concept 9										
sous_ensembles	1	1	1	3	**3**	**3**	3	**6**	6	**58**
δ	1	0.5	0.5	**0.75**						
Étape 5 - traitement du concept 4										
sous_ensembles	1	1	1	3	3	3	3	6	6	**55**
δ	1	0.5	0.5	0.75	**0.375**					
Étape 6 - traitement du concept 5										
sous_ensembles	1	1	1	3	3	3	3	**3**	**3**	**52**
δ	1	0.5	0.5	0.75	0.375	**0.375**				
Étape 7 - traitement du concept 6										
sous_ensembles	1	1	1	3	3	3	3	3	3	**49**
δ	1	0.5	0.5	0.75	0.375	0.375	**0.75**			
Étape 8 - traitement du concept 3										
sous_ensembles	1	1	1	3	3	3	3	3	3	**46**
δ	1	0.5	0.5	0.75	0.375	0.375	0.75	**0.375**		
Étape 9 - traitement du concept 2										
sous_ensembles	1	1	1	3	3	3	3	3	3	**43**
δ	1	0.5	0.5	0.75	0.375	0.375	0.75	0.375	**0.375**	
Étape 10 - traitement du concept 1										
sous_ensembles	1	1	1	3	3	3	3	3	3	**43**
δ	1	0.5	0.5	0.75	0.375	0.375	0.75	0.375	0.375	**0.672**

TABLE 6.7: Les différentes étapes du calcul des indices de stabilité par l'Algorithme 6.2

L'algorithme parcourt l'ensemble des concepts du treillis (*i.e.* $C \in \mathcal{C}$). Si la valeur $\delta(C)$ du concept courant C est inférieure au seuil η, alors les relations/arcs liants C à ses prédécesseurs et successeurs sont mis à jour. Cette mise à jour consiste à supprimer les relations joignant C à ses prédécesseurs $pred_C$ et à ses successeurs $succ_C$. Et, s'il n'existe pas d'autre chemin joignant un successeur $currentSucc$ à un prédécesseur $currentPred$ dans $\mathcal{L}$, il y a aussi création d'une relation entre $currentPred$ et $currentSucc$. Une fois cette mise à jour réalisée, ce concept C est supprimé de la structure.

Afin de conserver la possibilité d'une classification par navigation, le concept minimal étant le point d'entrée de cette dernière, alors **nous avons choisi, lorsque nous utilisons cette simplification, de ne jamais supprimer le concept minimal**, même si son indice est inférieur au seuil défini.

La structure résultante ne vérifie plus la définition de treillis (*cf.* définition 9) car la simplification supprime des concepts au milieu du treillis, ce qui engendre la suppression de concepts bornes inférieures ou bornes supérieures d'autres concepts. **La structure obtenue est alors d'une structure hybride entre treillis et arbre de classification.** Elle est composée de concepts reliés par la relation d'ordre $\prec$, et il est possible de l'utiliser par navigation pour la classification de nouveaux objets (*cf.* section 3.4.2).

Exemple. Les Figures 6.3.2 et 6.3.3 présentent un exemple de cette simplification. La Figure 6.3.2 présente le diagramme de Hasse d'un treillis complet (*i.e.* avant simplification), ses concepts sont étiquetés avec un identifiant, une liste d'objets et leur indice de stabilité. Les concepts terminaux sont représentés par les concepts en forme octogonale.

La Figure 6.3.3 présente ce même treillis après simplification par l'Algorithme 6.3 avec un seuil $\eta = 0.375$.

Tous les concepts ayant une stabilité inférieure ou égale à 0.375 ont été supprimés et les arcs ont été mis à jour selon l'Algorithme 6.3 (ces concepts et ces arcs sont en rouge dans la Figure 6.3.2). La Table 6.8 présente le récapitulatif des concepts supprimés et de l'arc créé (marqué et de couleur rouge dans la Figure 6.3.3).

Les concepts supprimés	Arc créé
3	$1 \rightarrow 9$
5	
11	

TABLE 6.8: Simplification du treillis de la Figure 6.3.2 - Stabilité

L'indice de stabilité a été utilisé, pour réduire la taille des treillis et conserver uniquement les informations les plus stables au sein d'une structure arborescente, *i.e.* seuls sont conservés les sous-groupes (concepts) d'objets formés par le treillis, qui sont suffisamment

Algorithme 6.3 *Simplification_treillis_stabilité*($\mathcal{L} = (\mathcal{C}, \prec)$,η)

Entrées :
- $\mathcal{L} = (\mathcal{C}, \prec, \Delta)$ le diagramme de Hasse d'un treillis marqué avec les indices de stabilité, à simplifier ;
- η le seuil ;

Sorties :
- $\mathcal{L}$ la structure simplifiée ;

DÉBUT
//Suppression des arcs et concepts inutiles selon l'Algorithmes 3.5
$Simplifier\_après\_Concepts\_Terminaux(\mathcal{L})$;
//Suppression des concepts dont l'indice de stabilité est inférieur à η
pour chaque $C \in \mathcal{C}$ **faire**
  si $\delta(C) \leq \eta$ **alors**
    //Mise à jour des relations dans la structure
    $pred_C \leftarrow$ ensemble des prédécesseurs de C ;
    $succ_C \leftarrow$ ensemble des successeurs de C ;
    pour chaque $pred\_courant \in pred_C$ **faire**
      Supprimer dans $\mathcal{L}$ l'arc joignant C à $pred\_courant$;
    fin pour
    pour chaque $succ\_courant \in succ_C$ **faire**
      Supprimer dans $\mathcal{L}$ l'arc joignant C à $succ\_courant$;
    fin pour
    pour chaque $(pred\_courant, succ\_courant) \in pred_C \times succ_C$ **faire**
      si il n'existe pas de chemin entre $pred\_courant$ et $succ\_courant$ dans $\mathcal{L}$ **alors**
        Ajouter un arc entre $pred\_courant$ et $succ\_courant$ dans $\mathcal{L}$;
      fin si
    fin pour
    Supprimer C de $\mathcal{L}$;
  fin si
fin pour
renvoyer $\mathcal{L}$;

FIN

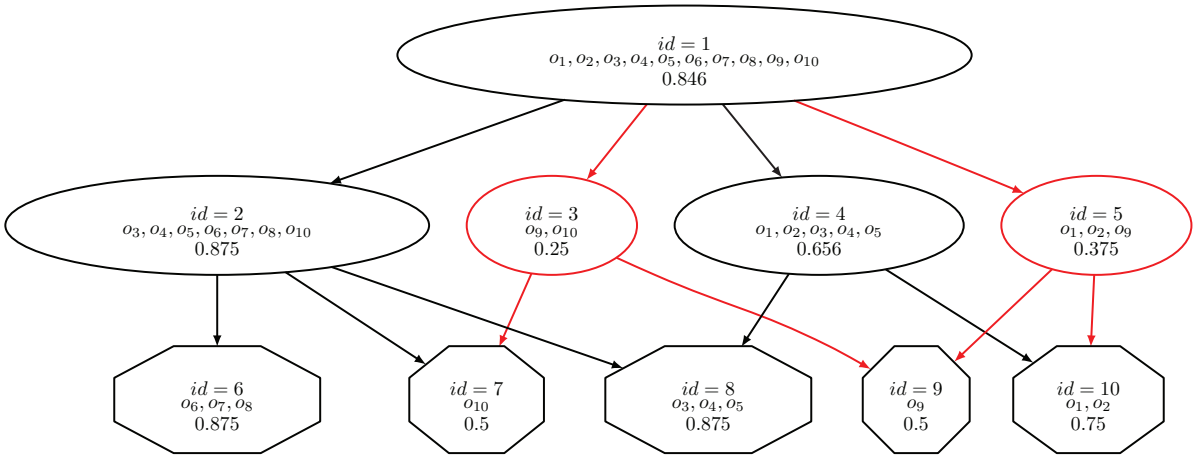


FIGURE 6.3.2: Treillis original

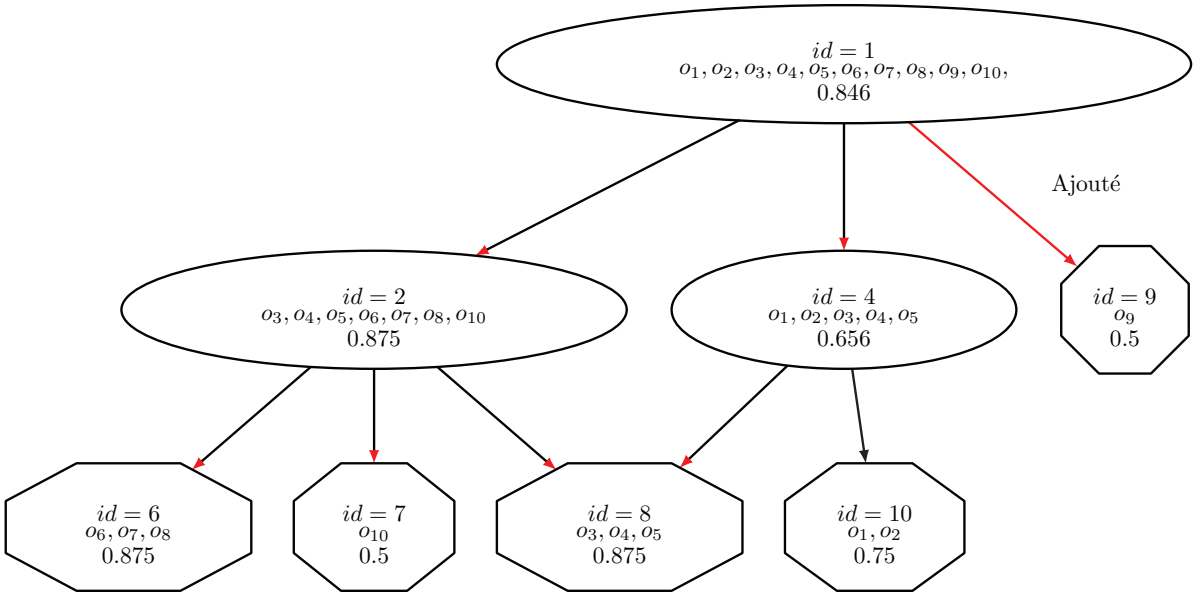


FIGURE 6.3.3: Treillis simplifié selon les indices de stabilité - seuil=0.375

stables (persistants) lors de la suppression de quelques objets [Roth 06, Kuznetsov 07b, Roth 08, Klimushkin 10, Falk 11].

La notion de *suffisamment stable* reste abstraite. De fait, **il est difficile de fixer un seuil pour l'indice de stabilité** permettant de dire que tous les concepts en deçà de ce seuil doivent être supprimés.

6.3.2 Proposition

Comme nous l'avons mentionné précédemment, nous souhaitons simplifier la structure du treillis pour obtenir une sous-structure au moins aussi performante en classification, mais aussi plus lisible, que le treillis complet. La simplification à base d'indice de stabilité permet de supprimer des concepts et des relations dans le treillis, et d'obtenir une sous-structure hiérarchique. Cette simplification a déjà été utilisée de manière efficace pour simplifier les treillis [Roth 06, Klimushkin 10, Falk 11], dans ces articles, les auteurs testent différents seuils fixés aléatoirement. Pour éviter cet aspect aléatoire, nous proposons **une définition automatique du seuil optimal** pour la simplification à base d'indice de stabilité **par optimisation du critère global de taux d'erreur**, comme cela est fait lors de l'élagage des arbres de classification. Ainsi, le processus proposé est hybride entre l'élagage d'arbre et la simplification de treillis à base d'indice de stabilité.

L'Algorithme 6.4 présente le processus de simplification du treillis. Cet algorithme prend en paramètre le diagramme de Hasse d'un treillis et une base de validation.

Comme mentionné dans la section précédente, il est possible de mémoriser à chaque étape le meilleur modèle en termes de classification, ainsi la séquence totale n'est pas mémorisée.

Comme pour l'élagage de type arbre, la première étape consiste à appliquer l'Algorithme 3.5 qui supprime tous les concepts non atteignables par navigation, ainsi que tous les arcs inutiles. Puis, après avoir ordonné les indices de stabilité des concepts par ordre croissant, le processus consiste à parcourir cette liste ordonnée en considérant chaque valeur, une à une, comme seuil candidat pour la simplification du treillis (simplification selon l'Algorithme 6.3).

Chaque treillis ainsi simplifié est évalué, celui offrant le plus petit taux d'erreur sur la base de validation est renvoyé.

Exemple. La Figure 6.3.3 correspond à la simplification du treillis de la Figure 6.3.2 par application de cet Algorithme 6.4. **Le seuil de** 0.375 **est le seuil optimal** lorsque la base d'apprentissage complète est utilisée comme base de validation.

Algorithme 6.4 *Simplification_hybride($\mathcal{L}$,BV)*

Entrées :
- $\mathcal{L} = (\mathcal{C}, \prec)$ le diagramme de Hasse à simplifier ;
- BV la base de validation ;

Sorties :
- $\mathcal{L}^*$ la structure hybride (*i.e.* le treillis simplifié) ;

DÉBUT
//Calcul des indices de stabilité selon Algorithme 6.2
$\mathcal{L} \leftarrow (\mathcal{C}, \prec, \Delta)$ = Calcul_Indices_Stabilité($\mathcal{L}$) ;
$\tilde{\mathcal{L}} \leftarrow \mathcal{L}$;
$taux\_min \leftarrow \varepsilon(\mathcal{L}, BV)$;
$liste\_indices \leftarrow$ liste ordonnée des indices de stabilité enregitrés dans Δ ;
pour chaque $\eta \in liste\_indices$ **faire**
  //Simplification du treillis selon l'Algorithme 6.3
  $\tilde{\mathcal{L}} \leftarrow$ Simplification_treillis_stabilité($\tilde{\mathcal{L}}, \eta$) ;
  si $\varepsilon(\tilde{\mathcal{L}}, BV) \leq taux$ **alors**
    $\mathcal{L}^* \leftarrow \tilde{\mathcal{L}}$;
    $taux\_min \leftarrow \varepsilon(\mathcal{L}^*, BV)$;
  fin si
fin pour
renvoyer $\mathcal{L}^*$;

FIN

Optimisation de l'algorithme

Comme évoqué précédemment, certains algorithmes d'élagage d'arbres de classification construisent la séquence des sous-arbres de manière partielle seulement. Ainsi, l'algorithme *REP* de Quinlan s'arrête lorsqu'aucun sous-arbre ne permet de diminuer ou de conserver le taux d'erreur obtenu avec l'arbre courant.

Dans l'Algorithme 6.4, une question qui se pose est : pourquoi parcourir tous les seuils possibles et ne pas stopper la simplification lorsque le seuil appliqué fait augmenter le taux d'erreur comparé au seuil précédent ?

Une seconde question se pose sur l'évolution de la complexité de la structure au fur et à mesure des étapes. En effet dans l'Algorithme 6.4, la taille de la structure n'est jamais utilisée pour choisir la meilleure, pourquoi ?

Concernant l'évolution du nombre de concepts de la structure au cours de cette simplification, celle-ci est présentée sur la Figure 6.3.4. Clairement, la taille de la structure décroît au cours du processus de simplification car des concepts sont supprimés à chaque nouveau seuil parcouru. Nous pouvons aussi voir que certains seuils permettent de supprimer beaucoup de concepts tandis que d'autres n'en suppriment que très peu.

Le taux d'erreur, quant à lui, n'est pas monotone en fonction des seuils (tel que représenté sur la Figure 6.3.4), contrairement à celui des arbres qui décroît constamment ce qui permet d'arrêter le processus avant la fin.

Ainsi, pour la simplification des treillis, il est nécessaire de parcourir l'ensemble des indices de stabilité. De plus, la structure est de plus en plus simple au cours des étapes, il n'est donc pas nécessaire d'utiliser la taille de la structure au cours de la simplification.

La Figure 6.3.4 est un extrait de la simplification appliquée au treillis construit après la discrétisation locale linéaire appliquée à l'ensemble de données Image1. Nous avons indiqué le seuil optimal choisi pour cet exemple en particulier, *via* l'axe vert.

6.3.3 Expérimentations

Les tables 6.9, et 6.10 présentent les résultats obtenus sur les bases utilisées dans le chapitre précédent. Les treillis sont construits en utilisant la discrétisation locale linéaire avec pour critère de coupe et d'arrêt le χ^2. Puis, ils sont simplifiés selon l'Algorithme 6.4 en utilisant la base d'apprentissage comme base de validation pour obtenir le modèle hybride. Les taux de reconnaissance obtenus avec le treillis avant simplification et avec le modèle hybride (*i.e.* après simplification) sont présentés dans la Table 6.9, ainsi que le pourcentage de différence entre ces taux. Le nombre de concepts de chaque structure est présenté dans la Table 6.10, ainsi que le pourcentage de concepts supprimés.

Nous pouvons voir sur ces expérimentations que **les taux de reconnaissances sont stables entre treillis complets et modèles hybride**s, avec une différence de $\pm 1\%$,

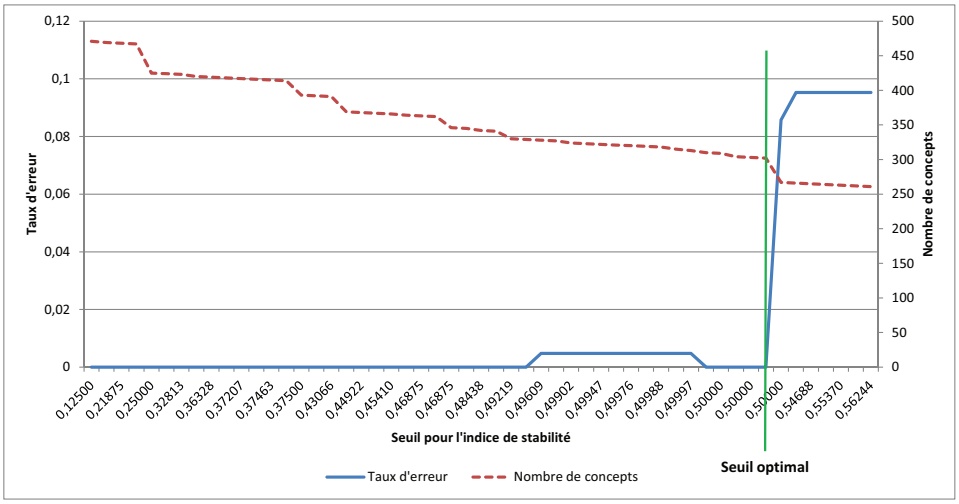


FIGURE 6.3.4: Évolution du taux d'erreur et du nombre de concepts en fonction du seuil au cours de la simplification (stabilité)

Ensemble de données	Treillis non simplifié	Modèle Hybride	Différence
Image1	**91,71**	90,95	- 0,83 %
GLASS	71,09 ±10,53	**71,83 ±13,37**	+ 1,05 %
IRIS	**95,33 ±5,21**	**95,33 ±5,21**	=
BREAST CANCER	94,43 ±2,05	**95,01 ±1,78**	+ 0,61 %
GREC Structurelle	**73,68**	72,96	- 0,98 %
GREC Radon	90,69 ±4,33	**90,73 ±4,43**	+ 0,04 %
Moyenne	86,15	86,14	

TABLE 6.9: Taux de reconnaissance - Treillis de Galois *vs* Modèle Hybride (indice de stabilité)

et cela **malgré une diminution de la taille de la structure généralement importante**, de 25% à 50% de concepts supprimés (sauf pour la base Iris avec seulement 12%, ce qui n'est pas surprenant car les treillis construits sur cette base sont déjà de petites tailles avant simplification).

Cependant, le processus proposé venant après la construction du treillis, comme nous pouvons le voir dans la Table 6.11, les temps de calculs en apprentissage sont allongés. En effet, le nombre de structures simplifiées à évaluer correspond au nombre de seuils, majoré par le nombre de concepts. Cette augmentation est contrebalancée par un temps en classification plus court, telle que présenté dans la Table 6.12. Dans cette table, les temps nécessaire à la classification des bases de test sont indiqués lorsque la structure complète est utilisée et lorsque le modèle hybride est utilisé. Le modèle hybride étant plus petit, le temps de calcul en classification est systématiquement réduit, tel qu'indiqué

Ensemble de données	Treillis non simplifié	Modèle hybride	Différence
Image1	649	**363**	- 44 %
GLASS	2267	**1127**	- 50 %
IRIS	41	**36**	- 12 %
BREAST CANCER	2961	**1939**	- 35 %
GREC Structurelle	3851	**1748**	- 55 %
GREC Radon	90	**68**	- 25 %

TABLE 6.10: Nombre de concepts - Treillis de Galois *vs* Modèle Hybride (indice de stabilité)

Ensemble de données	Treillis non simplifié	Modèle Hybride	Ratio
Image1	**359**	781	×2,17
GLASS	**221,9**	46021,9	×207,39
IRIS	**12,6**	**12,6**	×1
BREAST CANCER	**14632,5**	253323	×17,31
GREC Structurelle	**238**	575948	×2419,94
GREC Radon	**29,6**	30,2	×1,02

TABLE 6.11: Temps de calcul en apprentissage - secondes

par le ratio donné dans la dernière colonne de la table.

Bien que les temps de classification soient déjà faibles comparés aux temps d'apprentissage, la diminution du temps nécessaire à la classification reste importante cette étape étant généralement réalisée en ligne, contrairement à l'étape d'apprentissage.

Ensemble de données	Treillis non simplifié	Modèle Hybride	Ratio
Image1	433	**251**	÷1,72
GLASS	14,3	**8,4**	÷1,70
IRIS	0,7	**0,4**	÷1,75
BREAST CANCER	51,1	**36,1**	÷1,41
GREC Structurelle	1378	**270**	÷5,10
GREC Radon	36,9	**30,9**	÷1,19

TABLE 6.12: Temps de calcul en classification - millisecondes

6.4 Conclusion

Dans ce chapitre, nous avons présenté deux processus de simplification de la structure du treillis.

Le premier processus est une adaptation de l'élagage de CART qui utilise la mesure de coût-complexité (*CCP*), ce dernier ne permettant pas d'améliorer les performances du treillis, car il est peu adapté à sa structure car monotone.

Le second processus se base sur la simplification des treillis à base d'indices de stabilité, tout en utilisant les performances des structures en classification. **Ainsi, nous avons utilisé ce critère global supervisé pour définir automatiquement le seuil optimal, nécessaire à la simplification à base d'indices de stabilité.**

Nous avons choisi de tester deux processus de simplifications qui se distinguent par deux différences majeures :

1. Monotonie du critère : l'adaptation de la mesure de cout-complexité est un critère monotone, tandis que l'indice de stabilité est un critère non-monotone. (Rappelons que la monotonie n'est pas une propriété nécessaire pour obtenir une structure dans laquelle une navigation est possible, mais qu'elle est nécessaire seulement pour conserver la propriété de treillis).
2. Conservation de la structure de treillis : la simplification de type élagage d'arbres, permet d'obtenir un inf-demi-treillis, tandis que, comme mentionné précédemment, la simplification basée sur l'indice de stabilité des concepts, génère une structure hybride qui ne vérifie plus la propriété de treillis (en raison de sa non-monotonie).

Enfin, nous avons mené des expérimentations sur ces deux simplifications permettant de mettre en avant l'apport de la seconde approche.

Nous obtenons alors une structure hybride entre arbre de classification et treillis de Galois. Cette dernière offre l'avantage d'être plus stable que l'arbre et moins complexe que le treillis, tout en ayant des taux de reconnaissance similaires au treillis de Galois.

Points clés

Positionnement

- ❏ A partir du processus de post-élagage mis en œuvre pour les arbres de classification, nous avons exploré différentes possibilités pour simplifier la structure du treillis.
- ❏ Nous avons utilisé, en plus du post-élagage des arbres, des indices propres aux treillis de Galois.

Contributions

- ❏ Nous avons adapté à la structure particulière du treillis, la définition du critère d'élagage *CCP*.
- ❏ Nous avons mené des expérimentations permettant de mettre en avant que ce type d'élagage ascendant n'est pas adapté à la structure de treillis.
- ❏ **En s'inspirant du processus d'élagage des arbres, nous avons proposé un algorithme de définition automatique du seuil optimal, pour la simplification de la structure basée sur l'indice de stabilité des concepts.**
- ❏ Nous avons développé donc une simplification automatique et efficace de la structure de treillis.
- ❏ **La structure hybride obtenue a plus d'avantages en termes de classification que le treillis complet car elle l'égale en termes de taux de reconnaissance, tout en ayant une structure moins complexe permettant une classification plus rapide.**

Conclusion et perspectives

Rappel des objectifs

Les travaux exposés dans ce manuscrit sont tournés vers la classification et l'indexation d'images, et plus particulièrement vers la création d'un modèle de classification symbolique capable de traiter efficacement des données quantitatives. En effet les données extraites d'images pour leur analyse sont fréquemment quantitatives. De plus, avec l'émergence entre autres des applications interactives, il est de plus en plus nécessaire de proposer des modèles de classification qui soient compréhensibles et lisibles pour tout un chacun, cela afin que les résultats fournis par le modèle soient facilement interprétables et justifiables. Nous nous sommes intéressés à deux modèles de classification symboliques reconnus pour leur lisibilité, à savoir l'arbre de classification et le treillis de Galois.

Le premier objectif de cette thèse était de comparer ces deux modèles dans le cadre de la classification supervisée d'images. Nous voulions mettre en avant les avantages et les inconvénients de chacun.

Le second objectif était de proposer un modèle hybride entre ces deux derniers, qui combine les avantages des deux modèles d'origine (*i.e.* le faible espace mémoire de l'arbre de classification mais aussi son traitement local des données quantitatives d'une part, et d'autre part la robustesse du treillis de Galois pour la classification des données bruitées) tout en conservant la lisibilité des deux modèles.

Contributions et bilan des travaux effectués

Nous avons développé un modèle de classification hybride entre arbre de classification et treillis de Galois.

Tout d'abord, nous nous sommes intéressés aux points communs mais aussi aux différences qui existent entre arbres de classification et treillis de Galois.

Le chapitre 4 présente l'étude des points communs. Après avoir mis en avant **les liens en utilisation pour la classification par navigation** dans chaque structure, nous avons démontré **les liens structurels existants entre les deux modèles** (inclusion et fusion).

Concernant les liens en utilisation, rappelons que chacun des modèles est un graphe dans lequel une classification par navigation est possible. Dans chaque structure, la navigation est réalisée à partir du nœud « le plus général » (*i.e.* la racine de l'arbre ou le concept minimal du treillis) jusqu'à atteindre un nœud étiqueté par sa classe (*i.e.* les feuilles de l'arbre ou les concepts terminaux du treillis).

Concernant les liens structurels, rappelons que le treillis est défini de manière unique à partir d'une table de données qualitatives, tandis que plusieurs arbres peuvent être construits à partir d'une même table. Le lien d'inclusion correspond au fait que, lorsque les structures sont définies à partir de la même table de donnée qualitatives, tout arbre est inclus dans l'unique treillis. Le lien de fusion, quant à lui, est lié à la définition des **treillis dichotomiques**, il correspond au fait que ce treillis est la fusion de tous les arbres constructibles à partir de la même table de donnée qualitatives. Les treillis dichotomiques ont été définis selon une propriété de complémentarité portée par la table de données dans [Guillas 07, Bertet 09]. **Nous en avons donné une définition formelle et nous avons démontré ses propriétés que sont la $\vee$-complémentarité et la coatomicité.** La $\vee$-complémentarité est liée à la complémentarité des attributs qualitatifs considérés, sachant que cette complémentarité est vérifiée en particulier par les attributs qualitatifs issus de la discrétisation en intervalles disjoints d'attributs quantitatifs. La coatomicité permet, quant à elle, d'identifier les concepts terminaux.

Malgré ces liens forts, des différences existent entre ces deux modèles, en particulier pour le traitement des données quantitatives. En effet, alors que les arbres de classification utilisent une discrétisation locale pour transformer les données du type quantitatif vers le type qualitatif, jusqu'alors, une discrétisation globale était généralement utilisée avant toute construction de treillis de Galois.

Certaines études [Dougherty 95, Quinlan 93, Quinlan 96b] ayant montré que l'utilisation d'une discrétisation locale permet d'améliorer les performances en classification des arbres, comparée à une discrétisation globale, **nous avons proposé une discrétisation locale pour les treillis de Galois** (*cf.* chapitre 5). **L'algorithme développé s'inspire de la discrétisation locale des arbres de classification, tout en utilisant les propriétés structurelles des treillis dichotomiques**. En effet similairement aux arbres, la discrétisation locale est réalisée jusqu'à ce que les concepts terminaux (équivalents des feuilles de l'arbre) satisfassent un critère d'arrêt. Dans le cas des treillis dichotomiques, la coatomicité permet d'identifier ces concepts terminaux : ce sont les co-atomes.

A l'inverse de l'arbre de classification pour lequel la discrétisation locale engendre généralement une complexification de la structure [Dougherty 95], **la structure des treillis construits en utilisant notre discrétisation locale est simplifiée** comparée à celle des treillis définis à partir d'une discrétisation globale. De plus, le nombre d'étapes de discrétisation nécessaires pour aboutir à cette structure moins complexe est bien plus faible.

Enfin, comme pour l'arbre de classification [Quinlan 96b], **les performances en classification** des treillis définis avec une discrétisation locale **sont meilleures** que celles des treillis définis après une discrétisation globale.

Le traitement des données quantitatives par les treillis de Galois a donc été amélioré.

Nous avons montré sur différentes bases (et en particulier des bases d'images) que les performances en classification des treillis (et en particulier lorsqu'une discrétisation locale est utilisée) sont meilleures que celles des arbres. Cela provient certainement du fait que le treillis de Galois est plus robuste que les arbres de classification pour le traitement de données bruitées comme les caractéristiques issues d'images [Guillas 07]. Cela est dû en particulier à la multiplicité des chemins existants dans le treillis pour naviguer du concept minimal jusqu'à un concept terminal. Cependant, de ce fait la structure du treillis est aussi beaucoup plus complexe que la structure d'arbre.

Dans le chapitre 6, nous avons donc étudié le processus d'élagage existant pour les arbres de classification ainsi que les simplifications existantes pour la structure de treillis.

Tout d'abord, **nous avons adapté la mesure de coût-complexité** [Breiman 84] (considérée comme la plus efficace pour l'élagage des arbres [Rakotomalala 97]) afin de l'étendre à la simplification de la structure de treillis. Si cette mesure engendre mécaniquement une simplification de la structure de l'arbre, il n'en est pas de même pour les treillis, du fait de son caractère monotone. **Nous avons donc choisi d'utiliser un indice propre aux treillis** : l'indice de stabilité des concepts [Kuznetsov 90, Kuznetsov 03, Kuznetsov 07a].

Plus précisément, nous avons choisi cet indice car contrairement à l'adaptation de la mesure de coût-complexité précédente, l'indice de stabilité a la particularité d'être non-monotone. Or, la monotonie du critère utilisé pour la simplification a un impact fort sur la structure simplifiée obtenue. Lorsque le critère est monotone, la structure simplifiée est un $\vee$-demi-treillis, les concepts sont supprimés au fur et à mesure en commençant par le concept maximal et en remontant vers le concept minimal, ce qui préserve la structure de treillis mais n'engendre pas nécessairement une diminution du nombre de concepts entre deux étapes consécutives. A l'inverse, lorsqu'un critère non-monotone est utilisé, la structure simplifiée ne vérifie plus la propriété de treillis, les concepts pouvant être supprimés n'importe où dans la structure. La structure simplifiée résultante compte donc moins de concepts que la structure initiale.

L'utilisation de l'indice de stabilité nécessitant la définition d'un seuil, **nous avons proposé un algorithme de sélection automatique du seuil optimal**, *i.e.* l'algorithme sélectionne le meilleur seuil qui permet de diminuer le nombre, potentiellement exponentiel, de concepts dans la structure tout en conservant les performances en classification ainsi que la possibilité d'une classification par navigation. **En associant les**

indices de stabilité des concepts et le taux d'erreur du modèle, nous avons mis en œuvre une simplification structurelle du treillis aboutissant à un modèle hybride entre arbre de classification et treillis.

Apports, limites et perspectives

Le modèle proposé dans cette thèse profite des avantages de chaque modèle étudié :
- C'est un modèle arborescent comme l'arbre de classification et le treillis de Galois, la lisibilité des deux modèles est donc conservée ;
- La discrétisation locale, qui constituait un avantage de l'arbre de classification sur le treillis de Galois, a été développée pour ce dernier. Le modèle ainsi construit est plus performant que le treillis défini avec une discrétisation globale, tout en conservant la robustesse du treillis par rapport aux données bruitées et en diminuant la complexité (en nombre d'itérations) de la discrétisation et la complexité (en mémoire) de la structure de treillis ;
- Grâce à la simplification de la structure de treillis pour laquelle nous avons défini un seuillage automatique, le modèle simplifié reste plus complet que l'arbre de classification et plus simple que le treillis initial. Le défaut de complexité du treillis a donc été en partie supprimé.

Cependant, notre modèle a certaines limites que nous décrivons dans la suite de ce paragraphe. Les pistes d'améliorations proposées sont autant de perspectives intéressantes pour la suite de nos travaux.

Notre approche a été définie pour le traitement de données où les attributs explicatifs sont quantitatifs. Cependant la généralisation du processus, pour une prise en compte d'attributs explicatifs qualitatifs, reste simple et directe. En effet, le traitement de données hétérogènes reposeraient uniquement sur les définitions existantes : pour la phase d'expansion, les calculs des critères (de division, de coupe ou d'arrêt) sont déjà existants pour des données où les attributs explicatifs sont qualitatifs. Ainsi, pour les attributs qualitatifs de modalités mutuellement exclusives, notre adaptation des critères aux treillis reste donc inchangée. Les treillis construits seront aussi dichotomiques. Vu l'existence de nombreux ensembles de données de ce type, il serait intéressant de compléter le logiciel *Navigala 2012* pour qu'il traite ce type de données.

Concernant la discrétisation, l'algorithme proposé dans ce manuscrit peut être qualifié de glouton, ce dernier évaluant à chaque étape tous les points de coupe de chaque inf-irréductible. Aussi, comme perspectives de travail, il peut être intéressant de rechercher une optimisation de l'algorithme proposé. Par exemple, on pourrait proposer une mise à jour de l'ensemble des critères de gain des inf-irréductibles, plutôt que de tous les recalculer à chaque étape.

De plus, une différence réside encore entre la discrétisation locale menée dans les arbres de classification et celle proposée pour les treillis. En effet, l'arbre de classification est construit au cours de cette discrétisation, *i.e.* de manière incrémentale, tandis que le treillis est construit seulement une fois la discrétisation locale achevée.

Une étude de la possibilité d'une construction incrémentale nous semble une piste de recherche intéressante. En effet, il nous semble qu'un processus de duplication des concepts contenant l'intervalle discrétisé (*i.e.* duplication des concepts contenus entre le sup-irréductible associé à l'attribut discrétisé et le concept maximal), permettrait une construction incrémentale du treillis au fur et à mesure de la discrétisation locale.

Concernant la simplification de la structure, les points abordés dans ce manuscrit ont également permis de dégager des pistes de recherche. Pour l'extension de l'élagage de type arbre, nous avons vu que la taille du treillis n'évolue pas comme celle d'un arbre au cours des étapes d'élagage. En effet, celle-ci peut diminuer, mais aussi stagner, en raison de l'existence de plusieurs pères pour un même concept et de la monotonie des critères d'élagage utilisés pour les arbres. Sur ce constat, il serait intéressant dans le critère d'élagage, de ne pas considérer uniquement le nombre de concepts comme indicateur de la complexité structurelle, mais aussi le nombre d'arcs de la structure, ce dernier ne pouvant pas stagner d'une étape à l'autre.

Quant à la simplification à base d'indices, comme nous l'avons vu, la définition automatique du seuil optimal pour la stabilité est coûteuse en temps de calcul. La recherche d'une optimisation de cet algorithme nous semble être une piste intéressante. Cela permettrait ensuite une comparaison d'un plus grand nombre de critères, qu'ils soient monotone ou non (critère de support, indices de séparation, de probabilité, ...). Une optimisation permettrait aussi l'utilisation d'une combinaison linéaire des indices de stabilité et de probabilité, telle que le suggèrent les articles [Klimushkin 10, Falk 11].

Pour terminer, la génération du treillis à la demande ayant permis de diminuer les temps de calcul au cours de la phase d'apprentissage [Guillas 07], il nous semble intéressant d'étudier la possibilité d'une telle construction pour le modèle hybride. Dans ce cas, la génération à la demande serait dirigée localement par l'indice de stabilité et le taux d'erreur de la structure. En effet, le calcul de l'indice de stabilité peut se faire sans connaissance des successeurs. Ainsi, les concepts non pertinents ou induisant une erreur supplémentaire seraient supprimés juste après leur évaluation. Cela pallierait de plus, le surcoût de temps de calcul observé lors de la simplification basée sur cet indice.

Liste des symboles

(α, β) Une correspondance de Galois

(A, B) Un concept formel $A \subseteq O$ et $B \subseteq \mathcal{I}$

$(O, \mathcal{I}, (\alpha, \beta))$ Un contexte formel avec O un ensemble d'objets et $\mathcal{I}$ un ensemble d'attributs binaires

α Un application de $\mathcal{P}(O)$ vers $\mathcal{P}(\mathcal{I})$

$\bar{I}_j$ La famille non vide de parties d'attributs complémentaires de l'attribut binaire I_j

β Un application de $\mathcal{P}(\mathcal{I})$ vers $\mathcal{P}(O)$

$\mathcal{I}$ Un ensemble d'attributs qualitatifs

$\mathcal{V}$ Un ensemble d'attributs quantitatifs

$\mathcal{Y}$ Un ensemble d'attributs ($\mathcal{Y} = \mathcal{V} \cup \mathcal{I}$)

$\mathcal{C}$ Un ensemble de concepts

$\mathcal{F}$ Une famille de Moore

$\mathcal{I}_{\mathcal{N}}$ L'ensemble des attributs qualitatifs contenu dans le noeud $\mathcal{N}$

$\mathcal{L} = (\mathcal{C}, \leq)$ Un treillis de Galois sur l'ensemble des concepts $\mathcal{C}$ muni de la relation d'ordre $\leq$

$\mathcal{N}$ Le noeud courant d'un arbre de décision

$\mathcal{P}(\mathcal{X})$ L'ensemble des parties d'un ensemble $\mathcal{X}$

$\leq$ Une relation d'ordre

$\Omega = (O, \mathcal{I}, K)$ Un ensemble de données étiquetées contenant uniquement des attributs explicatifs qualitatifs ($\mathcal{I}$) et avec un attribut à expliquer qualitatif (K)

$\Omega = (O, \mathcal{V}, K)$ Un ensemble de données étiquetées contenant uniquement des attributs explicatifs quantiatifs ($\mathcal{V}$) et un attribut à expliquer qualitatif (K)

$\Omega = (O, \mathcal{Y}, K)$ Un ensemble de données étiquetées avec un attribut à expliquer qualitatif (K)

$\Omega_{\mathcal{N}}$ L'ensemble de données contenu dans le noeud $\mathcal{N}$

$\phi(.)$ Un modèle de classification

$\prec$ Une relation de couverture

$\varepsilon(\phi(.), BA)$ Le taux d'erreur en resubstitution du modèle $\phi(.)$

$\varepsilon(\phi(.), BT)$ Le taux d'erreur du modèle $\phi(.)$ calculé sur la base de test BT

$\varepsilon(\phi(.), BV)$ Le taux d'erreur du modèle $\phi(.)$ calculé sur la base de validation BV

$\vec{o_n}$ Le vecteur des attributs associé à l'objet o_n

$\vee$ La borne supérieure

$\wedge$ La borne inférieure

BA La base d'apprentissage

BT La base de test

BV La base de validation

c_V^i Un point de coupe possible pour l'attribut quantitatif $V \in \mathcal{V}$, $c_V^i \in C_V$

C_V L'ensemble des points de coupe possibles pour l'attribut quantitatif $V \in \mathcal{V}$

DT_0 L'arbre de décision de taille maximale

I Un attribut qualitatif parmi l'ensemble $\mathcal{I}_{\mathcal{N}}$

$I_{.z}$ Un attribut qualitatif parmi l'ensemble $\mathcal{I}$

Inf_x L'ensemble des minorants/prédecesseurs d'un élément x

K Un attribut à expliquer qualitatif

k_{o_n} La valeur de l'attribut à expliquer prise par l'objet o_n

$L(DT_i)$ Le nombre de feuilles de l'arbre DT_i

m_{nz} La modalité de l'attribut qualitatif $I_{.z}$ vérifiée par l'objet o_n

$MC(\phi(.), BT)$ Le nombre d'objets de la base de test BT, mal classés par le modèle $\phi(.)$

O Un ensemble d'objets

o_n Un objet de O

$O_{\mathcal{N}}$ L'ensemble des objets contenu dans le noeud $\mathcal{N}$

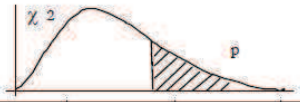
ST_i Une branche d'un arbre de racine $\mathcal{N}$

Sup_x	L'ensemble des majorants/successeurs d'un élément x
t_1	L'arbre ne contenant que la racine
$V_{.h}$	Un attribut quantitatif parmi l'ensemble $\mathcal{V}$
v_{nh}	La valeur prise par l'objet o_n pour l'attribut quantitatif $V_{.h}$
$<$	Une relation d'ordre strict

Annexe A

Table du χ^2

TABLE DU CHI-DEUX : $\chi^2(n)$



n p	0.90	0.80	0.70	0.50	0.30	0.20	0.10	0.05	0.02	0.01
1	0,0158	0,0642	0,148	0,455	1,074	1,642	2,706	3,841	5,412	6,635
2	0,211	0,446	0,713	1,386	2,408	3,219	4,605	5,991	7,824	9,210
3	0,584	1,005	1,424	2,366	3,665	4,642	6,251	7,815	9,837	11,341
4	1,064	1,649	2,195	3,357	4,878	5,989	7,779	9,488	11,668	13,277
5	1,610	2,343	3,000	4,351	6,064	7,289	9,236	11,070	13,388	15,086
6	2,204	3,070	3,828	5,348	7,231	8,558	10,645	12,592	15,033	16,812
7	2,833	3,822	4,671	6,346	8,383	9,803	12,017	14,067	16,622	18,475
8	3,490	4,594	5,527	7,344	9,524	11,030	13,362	15,507	18,168	20,090
9	4,168	5,380	6,393	8,343	10,656	12,242	14,684	16,919	19,679	21,666
10	4,865	6,179	7,267	9,342	11,781	13,442	15,987	18,307	21,161	23,209
11	5,578	6,989	8,148	10,341	12,899	14,631	17,275	19,675	22,618	24,725
12	6,304	7,807	9,034	11,340	14,011	15,812	18,549	21,026	24,054	26,217
13	7,042	8,634	9,926	12,340	15,119	16,985	19,812	22,362	25,472	27,688
14	7,790	9,467	10,821	13,339	16,222	18,151	21,064	23,685	26,873	29,141
15	8,547	10,307	11,721	14,339	17,322	19,311	22,307	24,996	28,259	30,578
16	9,312	11,152	12,624	15,338	18,418	20,465	23,542	26,296	29,633	32,000
17	10,085	12,002	13,531	16,338	19,511	21,615	24,769	27,587	30,995	33,409
18	10,865	12,857	14,440	17,338	20,601	22,760	25,989	28,869	32,346	34,805
19	11,651	13,716	15,352	18,338	21,689	23,900	27,204	30,144	33,687	36,191
20	12,443	14,578	16,266	19,337	22,775	25,038	28,412	31,410	35,020	37,566
21	13,240	15,445	17,182	20,337	23,858	26,171	29,615	32,671	36,343	38,932
22	14,041	16,314	18,101	21,337	24,939	27,301	30,813	33,924	37,659	40,289
23	14,848	17,187	19,021	22,337	26,018	28,429	32,007	35,172	38,968	41,638
24	15,659	18,062	19,943	23,337	27,096	29,553	33,196	36,415	40,270	42,980
25	16,473	18,940	20,867	24,337	28,172	30,675	34,382	37,652	41,566	44,314
26	17,292	19,820	21,792	25,336	29,246	31,795	35,563	38,885	42,856	45,642
27	18,114	20,703	22,719	26,336	30,319	32,912	36,741	40,113	44,140	46,963
28	18,939	21,588	23,647	27,336	31,391	34,027	37,916	41,337	45,419	48,278
29	19,768	22,475	24,577	28,336	32,461	35,139	39,087	42,557	46,693	49,588
30	20,599	23,364	25,508	29,336	33,530	36,250	40,256	43,773	47,962	50,892

Pour n > 30, on peut admettre que $\sqrt{2\chi^2} - \sqrt{2n-1} \approx N(0,1)$

Annexe B

Fichier de description d'un graphe pour Graphviz

L'utilisation du logiciel Graphviz repose sur la définition textuelle d'un graphe selon le format *DOT*[1] et dans un fichier '.dot'. Ce fichier doit contenir, tout d'abord la description des nœuds puis celle des arcs qui lient ces nœuds. Le cadre ci-dessous présente un exemple de fichier décrivant le diagramme de Hasse du treillis de Galois associé au contexte formel de la table B.1. Le dessin résultat est présenté dans la Figure B.0.1.
Notons que les notations indice et exposant, telles que dans la Figure B.0.1, ne sont pas prises en charge par le logiciel Graphviz. Nous avons utilisé le package LaTex *dot2texi*[2], développé par Fauske. Ce package permet de générer des graphes contenant ce type de notations.

O \ $\mathcal{I}$	I_1	I_2	I_3	I_4
o_1	×	×		
o_2		×		
o_3		×	×	
o_4			×	×

Table B.1: Exemple de contexte formel

1. Langage *DOT* : `http://www.graphviz.org/content/dot-language`
2. Package dot2texi : `http://www.fauskes.net/code/dot2tex/documentation/`

monGraphe.dot

```
digraph G{
/*Description des nœuds ou concepts*/
1 [label="o_1,o_2,o_3,o_4\n \\emptyset"];
2 [label="o_1,o_2,o_3\n I_2"];
3 [label="o_3,o_4\n I_3"];
4 [label="o_1\n I_1,I_2"];
5 [label="o_3\n I_2,I_3"];
6 [label="o_4\n I_3,I_4"];
7 [label="\\emptyset\n I_1,I_2,I_3,I_4"];
/*Description des relations entre concepts*/
1 -> 2; 1 -> 3;
2 -> 4; 2 -> 5;
3 -> 5; 3 -> 6;
4 -> 7; 5 -> 7; 6 -> 7;
}
```

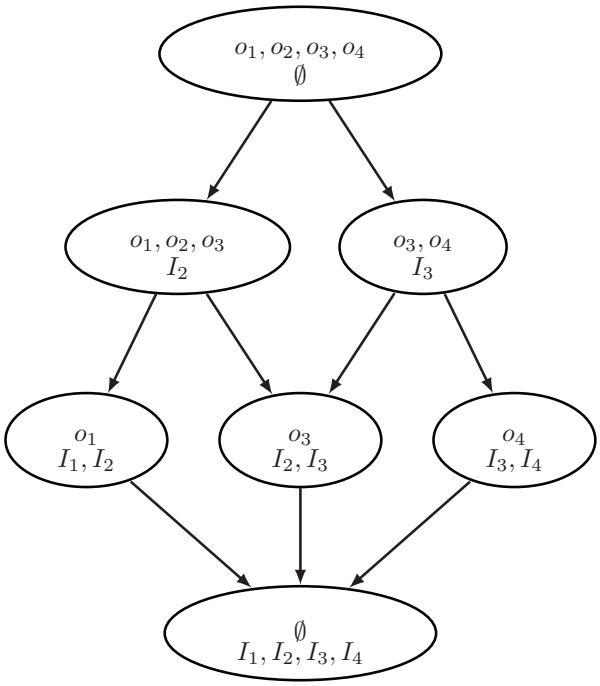


FIGURE B.0.1: Diagramme de Hasse du treillis de Galois de la table B.1 - Graphviz, dot, dot2texi

Annexe C

Critères de coupe - formules simplifiées

Soit un point de coupe c_V défini pour l'attribut quantitatif V et un ensemble de données Ω. Soit l'attribut qualitatif I correspondant, *i.e.* l'attribut à deux modalités $[min_V, c_V]$ et $]c_V, max_V]$. La table de contingence associée à cette discrétisation est présentée dans la table C.1.

K	I		$\sum$
	$[min_V, c_V]$	$]c_V, max_V]$	
q_1	n_{11}	n_{12}	$n_{1.}$
$\vdots$			
q_h	n_{h1}	n_{h2}	$n_{h.}$
$\vdots$			
q_H	n_{H1}	n_{H2}	$n_{H.}$
$\sum$	$n_{.1}$	$n_{.2}$	N

TABLE C.1: Table de contingence pour un attribut I qualitatif

Les critères de coupe usuels sont définis comme suit.

– Critère d'Entropie :

$$gain(V, c_V, \Omega) = gain_{Entropie}(V, c_V, \Omega) = E(K) - E_c(K/V)$$

Avec $E(K) = -\sum_{h=1}^{H} p_{h.} log_2(p_{h.})$ et
$E_c(K/V) = -\Big(p_{.1} \sum_{h=1}^{H} p_{h1}\, log_2(p_{h1}) + p_{.2} \sum_{h=1}^{H} p_{h2}\, log_2(p_{h2})\Big)$.

– Critère du χ^2 :

$$gain(V, c_V, \Omega) = gain_{\chi^2}(V, c_V, \Omega) = \sum_{h=1}^{H} \left(\frac{(n_{h1} - \frac{n_{h.} \times n_{.1}}{N})^2}{\frac{n_{h.} \times n_{.1}}{N}} + \frac{(n_{h2} - \frac{n_{h.} \times n_{.2}}{N})^2}{\frac{n_{h.} \times n_{.2}}{N}} \right)$$

– Critère de Gini :

$$gain(V, c_V, \Omega) = gain_{Gini}(V, c_V, \Omega) = Gini(K) - Gini_c(K/V)$$

Avec $Gini_c(K/VI) = -\left(p_{.1} \times (1 - \sum_{h=1}^{H} p_{h1}^2) + p_{.2} \times (1 - \sum_{h=1}^{H} p_{h2}^2)\right)$ et $Gini(K) = 1 - \sum_{h=1}^{H} p_{h.}^2$.

Annexe D

Notices des bases de données utilisées

Les sections suivantes présentent les informations utiles sur les bases utilisées au cours de nos expérimentations. Les descriptions complétes sont disponibles sur internet.

D.1 Breast Cancer Database

Description compléte ici.

Citation Request :

This breast cancer databases was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg :

1. O. L. Mangasarian and W. H. Wolberg : "Cancer diagnosis via linear programming", SIAM News, Volume 23, Number 5, September 1990, pp 1 & 18.
2. William H. Wolberg and O.L. Mangasarian : "Multisurface method of pattern separation for medical diagnosis applied to breast cytology", Proceedings of the National Academy of Sciences, U.S.A., Volume 87, December 1990, pp 9193-9196.

Title :

Wisconsin Breast Cancer Database (January 8, 1991)

Sources :

- Dr. WIlliam H. Wolberg (physician) University of Wisconsin Hospitals Madison, Wisconsin USA
- Donor : Olvi Mangasarian (mangasarian@cs.wisc.edu) Received by David W. Aha (aha@cs.jhu.edu)
- Date : 15 July 1992

Past Usage :

Attributes 2 through 10 have been used to represent instances. Each instance has one of 2 possible classes : benign or malignant.

1. Wolberg, W. H., and Mangasarian, O. L. (1990). Multisurface method of pattern separation for medical diagnosis applied to breast cytology. In *Proceedings of the National Academy of Sciences*, *87*, 9193–9196.
   a) Size of data set : only 369 instances (at that point in time)
   b) Collected classification results : 1 trial only
   c) Two pairs of parallel hyperplanes were found to be consistent with 50% of the data
      – Accuracy on remaining 50% of dataset : 93.5%
   d) Three pairs of parallel hyperplanes were found to be consistent with 67% of data
      – Accuracy on remaining 33% of dataset : 95.9%
2. Zhang, J. (1992). Selecting typical instances in instance-based learning. In *Proceedings of the Ninth International Machine Learning Conference* (pp. 470–479). Aberdeen, Scotland : Morgan Kaufmann.
   a) Size of data set : only 369 instances (at that point in time)
   b) Applied 4 instance-based learning algorithms
   c) Collected classification results averaged over 10 trials
   d) Best accuracy result :
      – 1-nearest neighbor (trained on 200 instances, tested on the other 169) : 93.7%
   e) Also of interest :
      – Using only typical instance (trained on 200 instances, tested on the other 169) : 92.2% (storing only 23.1 instances)

Relevant Information :

Samples arrive periodically as Dr. Wolberg reports his clinical cases. The database therefore reflects this chronological grouping of the data.

– Total : 699 points (as of the donated datbase on 15 July 1992)

Note that the results summarized above in Past Usage refer to a dataset of size 369.

Number of Instances :

699 (as of 15 July 1992)

Number of Attributes :

10 plus the class attribute

Attribute Information :

Missing Attribute Values :

There are 16 instances in Groups 1 to 6 that contain a single missing (*i.e.*, unavailable) attribute value, now denoted by "?". Replaced by 0.

#	Attribute	Domain
1.	Sample code number	id number
2.	Clump Thickness	1 - 10
3.	Uniformity of Cell Size	1 - 10
4.	Uniformity of Cell Shape	1 - 10
5.	Marginal Adhesion	1 - 10
6.	Single Epithelial Cell Size	1 - 10
7.	Bare Nuclei	1 - 10
8.	Bland Chromatin	1 - 10
9.	Normal Nucleoli	1 - 10
10.	Mitoses	1 - 10
11.	Class	2 for benign, 4 for malignant

TABLE D.1: Attribute Informations - Breast Cancer database

Class distribution :
Benign : 458 (65.5%) Malignant : 241 (34.5%)

D.2 Image Segmentation Database

Description compléte ici.

Title :
Image Segmentation data

Source Information
- Creators : Vision Group, University of Massachusetts
- Donor : Vision Group (Carla Brodley, brodley@cs.umass.edu)
- Date : November, 1990 3. Past Usage : None yet published

Relevant Information :
The instances were drawn randomly from a database of 7 outdoor images. The images were handsegmented to create a classification for every pixel. Each instance is a 3x3 region.

Number of Instances :
Training data : 210
Test data : 2100

Number of Attributes :
19 continuous attributes

#	Attribute	Domain
1.	region-centroid-col	$\mathbb{N}$ - the column of the center pixel of the region
2.	region-centroid-row	$\mathbb{N}$ - the row of the center pixel of the region
3.	region-pixel-count	9
4.	short-line-density-5	$\mathbb{R}$ - the results of a line extraction algorithm that counts how many lines of length 5 (any orientation) with low contrast, less than or equal to 5, go through the region
5.	short-line-density-2	$\mathbb{R}$ - same as short-line-density-5 but counts lines of high contrast, greater than 5.
6.	vedge-mean	$\mathbb{R}$ - measure the contrast of horizontally adjacent pixels in the region. There are 6, the mean and standard deviation are given. This attribute is used as a vertical edge detector.
7.	vegde-sd	$\mathbb{R}$ - see 6
8.	hedge-mean	$\mathbb{R}$ - measures the contrast of vertically adjacent pixels. Used for horizontal line detection.
9.	hedge-sd	$\mathbb{R}$ - see 8
10.	intensity-mean	$\mathbb{R}$ - the average over the region of (R + G + B)/3
11.	rawred-mean	$\mathbb{R}$ - the average over the region of the R value
12.	rawblue-mean	$\mathbb{R}$ - the average over the region of the B value
13.	rawgreen-mean	$\mathbb{R}$ - the average over the region of the G value
14.	exred-mean	$\mathbb{R}$ - measure the excess red : (2R - (G + B))
15.	exblue-mean	$\mathbb{R}$ - measure the excess blue : (2B - (G + R))
16.	exgreen-mean	$\mathbb{R}$ - measure the excess green : (2G - (R + B))
17.	value-mean	$\mathbb{R}$ - 3-d nonlinear transformation of RGB. (Algorithm can be found in Foley and VanDam, Fundamentals of Interactive Computer Graphics)
18.	saturatoin-mean	$\mathbb{R}$ - see 17
19.	hue-mean	$\mathbb{R}$ - see 17

Table D.2: Attribute Informations - Image segmentation database

Attribute Information : (see table D.2)

Missing Attribute Values :
None

Class Distribution :
Classes : brickface, sky, foliage, cement, window, path, grass.
– 30 instances per class for training data.
– 300 instances per class for test data.

D.3 Glass Identification Database

Description compléte ici.

Title :
Glass Identification Database

Sources :
1. Creator : B. German
   – Central Research Establishment Home Office Forensic Science Service Aldermaston, Reading, Berkshire RG7 4PN
2. Donor : Vina Spiehler, Ph.D., DABFT Diagnostic Products Corporation (213) 776-0180 (ext 3014)
3. Date : September, 1987

Past Usage :
– Rule Induction in Forensic Science
  – Ian W. Evett and Ernest J. Spiehler (Central Research Establishment Home Office Forensic Science Service Aldermaston, Reading, Berkshire RG7 4PN)
  – Unknown technical note number
  – General Results : nearest neighbor held its own with respect to the rule-based system

Relevant Information :
Vina conducted a comparison test of her rule-based system, BEAGLE, the nearest-neighbor algorithm, and discriminant analysis. In determining whether the glass was a type of "float" glass or not, the following results were obtained (# incorrect answers) :

The study of classification of types of glass was motivated by criminological investigation. At the scene of the crime, the glass left can be used as evidence, if it is correctly identified !

Number of Instances :
214

Type of Sample	Beagle	NN	DA
Windows that were float processed (87)	10	12	21
Windows that were not (76)	19	16	22

TABLE D.3: Past Classification Results - Glass data base

Number of Attributes :
10 (including an Id#) plus the class attribute, all attributes are continuously valued.

Attribute Information : (see table D.4)

#	Attribute	Domain
1.	Id number	$\mathbb{N}$ - 1 to 214
2.	RI	$\mathbb{R}$ - refractive index
3.	Na	$\mathbb{R}$ - Sodium (unit measurement : weight percent in corresponding oxide, as are attributes 4-10)
4.	Mg	$\mathbb{R}$ - Magnesium
5.	Al	$\mathbb{R}$ - Aluminum
6.	Si	$\mathbb{R}$ - Silicon
7.	K	$\mathbb{R}$ - Potassium
8.	Ca	$\mathbb{R}$ - Calcium
9.	Ba	$\mathbb{R}$ - Barium
10.	Fe	$\mathbb{R}$ - Iron
11.	Type of glass : (class attribute)	{1-building_windows_float_processed, 2-building_windows_non_float_processed, 3-vehicle_windows_float_processed, 5-containers, 6-tableware, 7-headlamps}

TABLE D.4: Attribute Informations - Glass database

Missing Attribute Values :
None

Class Distribution : (see table D.6)

D.4 Iris Database

Title :
Iris Plants Database Updated Sept 21 by C.Blake - Added discrepency information

Sources :
1. Creator : R.A. Fisher

	Min	Max	Mean	SD	Correlation with class
RI	1.5112	1.5339	1.5184	0.0030	-0.1642
Na	10.73	17.38	13.4079	0.8166	0.5030
Mg	0	4.49	2.6845	1.4424	-0.7447
Al	0.29	3.5	1.4449	0.4993	0.5988
Si	69.81	75.41	72.6509	0.7745	0.1515
K	0	6.21	0.4971	0.6522	-0.0100
Ca	5.43	16.19	8.9570	1.4232	0.0007
Ba	0	3.15	0.1750	0.4972	0.5751
Fe	0	0.51	0.0570	0.0974	-0.1879

TABLE D.5: Summary Statistics - Glass database

<table>
<tr><td rowspan="4">163 Window glass</td><td rowspan="2">87 float processed</td><td>70 building windows (class 1)</td></tr>
<tr><td>17 vehicle windows (class 3)</td></tr>
<tr><td rowspan="2">76 non-float processed</td><td>76 building windows (class 2)</td></tr>
<tr><td>0 vehicle windows (class 4)</td></tr>
<tr><td rowspan="3">51 Non-window glass</td><td colspan="2">13 containers (class 5)</td></tr>
<tr><td colspan="2">9 tableware (class 6)</td></tr>
<tr><td colspan="2">29 headlamps (class 7)</td></tr>
</table>

TABLE D.6: Class Distribution - Glass database

2. Donor : Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
3. Date : July, 1988

Past Usage :

1. Dasarathy, B.V. (1980) "Nosing Around the Neighborhood : A New System Structure and Classification Rule for Recognition in Partially Exposed Environments". IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No. 1, 67-71.

   a) Results : very low misclassification rates (0% for the setosa class)

2. Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule". IEEE Transactions on Information Theory, May 1972, 431-433.

   a) Results : very low misclassification rates again

3. See also : 1988 MLC Proceedings, 54-64. Cheeseman *et al*'s AUTOCLASS II conceptual clustering system finds 3 classes in the data.

Relevant Information :

- This is perhaps the best known database to be found in the pattern recognition literature. Fisher's paper is a classic in the field and is referenced frequently to this day. (See Duda & Hart, for example.) The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2 ; the latter are NOT linearly separable from each other.
- Predicted attribute : class of iris plant.
- This is an exceedingly simple domain.
- This data differs from the data presented in Fishers article (identified by Steve Chadwick, spchadwick@espeedaz.net). The 35th sample should be : 4.9,3.1,1.5,0.2,"Iris-setosa" where the error is in the fourth feature. The 38th sample : 4.9,3.6,1.4,0.1,"Iris-setosa" where the errors are in the second and third features.

Number of Instances :

150

Number of Attributes :

4 numeric, predictive attributes and the class

Attribute Information : (see table D.7)

Missing Attribute Values :

None

Class Distribution :

33.3% for each of 3 classes.

#	Attribute	Domain
1.	sepal length	$\mathbb{R}$ - in cm
2.	sepal width	$\mathbb{R}$ - in cm
3.	petal length	$\mathbb{R}$ - in cm
4.	petal width	$\mathbb{R}$ - in cm
5.	class attribute	{Setosa, Versicolour, Virginica}

Table D.7: Attribute Informations - Glass database

	Min	Max	Mean	SD	Class Correlation
sepal length	4.3	7.9	5.84	0.83	0.7826
sepal width	2.0	4.4	3.05	0.43	-0.4194
petal length	1.0	6.9	3.76	1.76	0.9490 (high!)
petal width	0.1	2.5	1.20	0.76	0.9565 (high!)

Table D.8: Summary Statistics - Iris database

D.5 GREC 2003 database

[EXTRAIT DE LA THÈSE DE S. GUILLAS [Guillas 07]]

La base GREC 2003 [GREC 03] (Graphics RECognition) est composée d'images de symboles segmentés, c'est-à-dire isolés. Elle a été créée pour un concours de reconnaissance de symboles organisé au cours de l'atelier Graphics Recognition qui a lieu tous les 2 ans. Il y a trois catégories de bruits proposées : les dégradations binaires (bruit impulsif), les distorsions vectorielles et les images réelles numérisées et photocopiées.

Concernant les dégradations binaires, la base contient 39 classes de symboles (voir Figure D.5.1) et 3510 symboles bruités. Chaque classe contient 1 symbole modèle (non bruité) et 90 symboles bruités. Les bruits appliqués sur les images reproduisent les déformations qui peuvent survenir lors de l'utilisation d'un photocopieur, d'un scanner ou encore d'une imprimante. Ils sont générés par la méthode Kanungo [Kanungo 94] et sont composés de 9 types de dégradation (voir Figure D.5.2). Pour chaque type de dégradation, la base contient 10 symboles bruités par classe. Ces symboles sont également proposés en ayant subi une rotation ou un changement d'échelle.

Pour les distorsions vectorielles, la base contient 15 classes de symboles architecturaux ou électriques (voir Figure D.5.3) et 600 symboles distordus. Les distorsions sont de trois types (voir Figure D.5.4). Ces symboles sont également proposés en ayant subi une rotation, un changement d'échelle, ou en étant doublement bruités par dégradations binaires et par distorsions vectorielles.

Enfin, les images réelles sont composées des 15 symboles architecturaux et électriques

présentés précédemment et de 75 symboles bruités par numérisation et 75 symboles bruités par numérisation et photocopie.

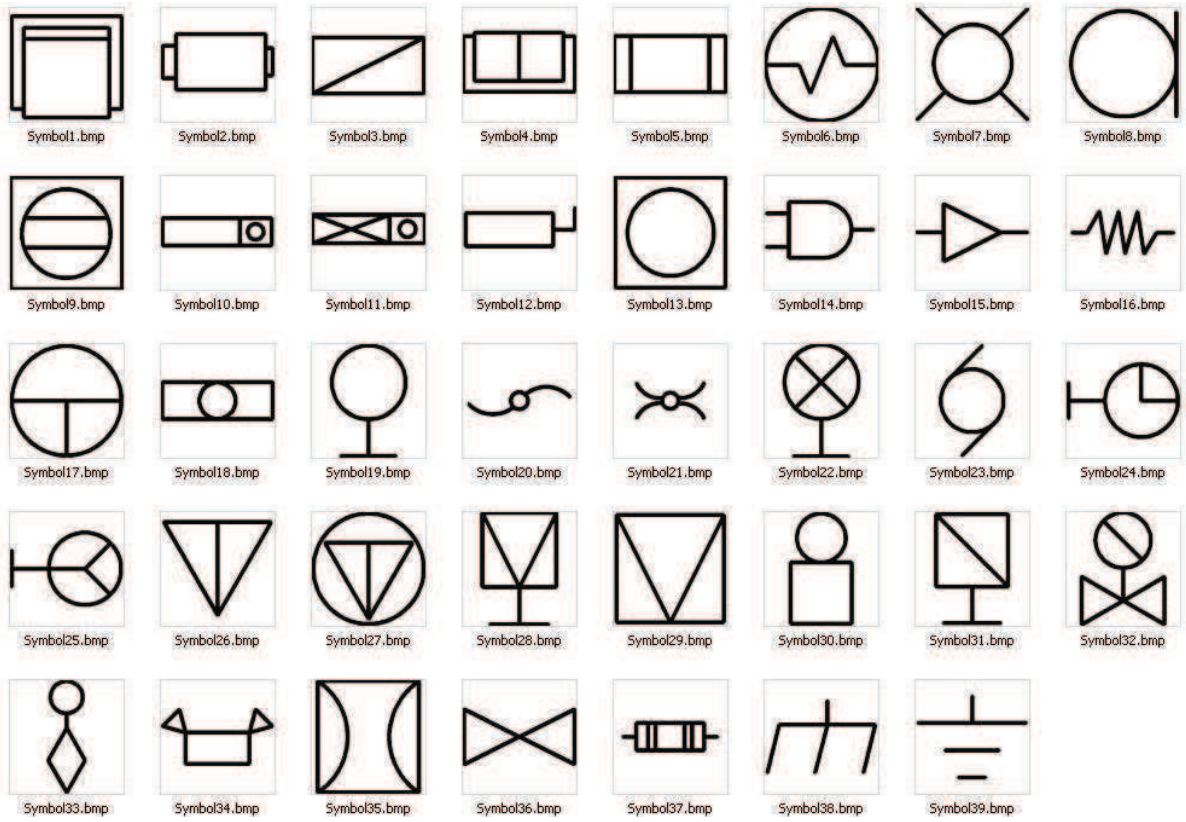


FIGURE D.5.1: Symboles modèles de la base GREC 2003 (dégradations binaires)

D.5.1 GREC Radon

La base GREC Radon utilisée pour nos expérimentations se limitent à 910 objets parmi l'ensemble des images proposées. Seules les premières 10 classes d'images sont utilisées (*cf.* « Exp 2 », p 169 de [Guillas 07]). Pour chaque classe le symbole original, *i.e.* non dégradé est utilisé ainsi que 90 symboles dégradés. La signature extraite de ces 910 symboles est la signature de Radon telle que définie dans la thèse de S. Guillas [Guillas 07] (p 8 et 76).

D.5.2 GREC Structurelle

La base GREC Structurelle utilisée pour les expérimentations présentées dans ce manuscrit se limitent quant à elle à 1 900 objets parmi l'ensemble des images proposées. Seules 19 classes sont utilisées, pour chaque classe le symbole original, *i.e.* non dégradé

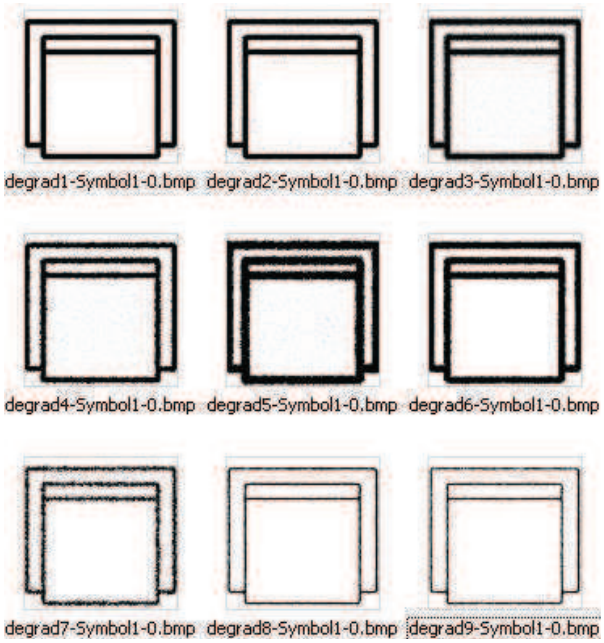


FIGURE D.5.2: Symboles bruités de la base GREC 2003 (dégradations binaires)

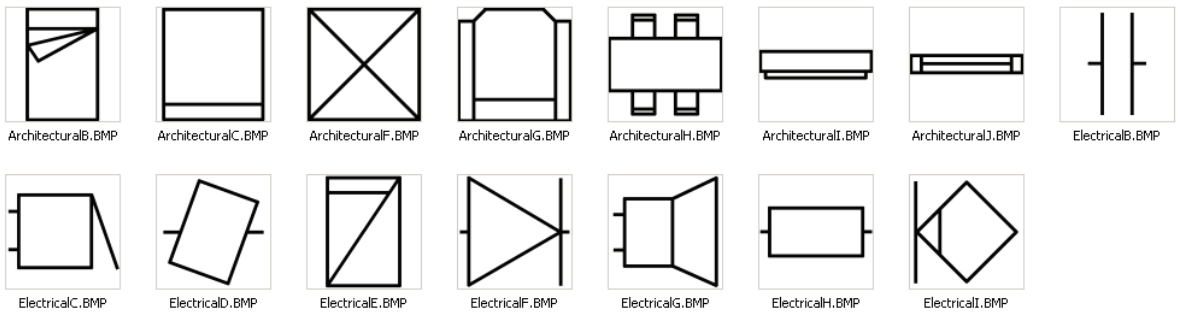


FIGURE D.5.3: Symboles modèles de la base GREC 2003 (distorsions vectorielles)

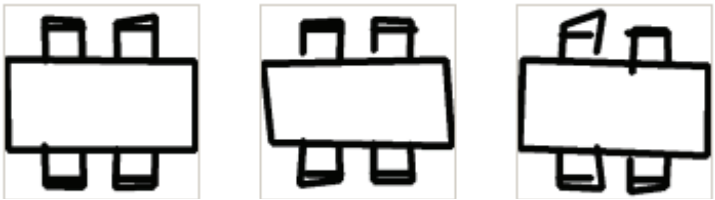


FIGURE D.5.4: Symboles bruités de la base GREC 2003 (distorsions vectorielles)

est utilisé ainsi que 99 symboles dégradés. La signature extraite de ces 1 900 symboles est une signature structurelle composée de 15 descripteurs quantitatifs, telle que définie dans [Guillas 07, Coustaty 11b]. Cette signature structurelle représente l'organisation spatiale des primitives graphiques (lignes, arcs de cercle, ...) contenues dans l'image. Ces primitives graphiques sont extraites à partir de la transformée de Hough [Hough 62]. Puis un graphe de vecteur intégrant les relations spatiales entre paire de primitives est construit. Différents types de relations spatiales peuvent être considérés comme illustrés sur la Figure D.5.5. Enfin la signature quantitative est extraite de la matrice adjacence de ce graphe de vecteurs. Les valeurs représentent la longueur des chemins dans le graphe de vecteurs [Guillas 07] (p 77 à 91). Pour nos tests, nous avons utilisées les chemins de longueur 2.

FIGURE D.5.5: Types de relations spatiales entre deux segments s et s' [Guillas 07]

Annexe E

Développement logiciel

Les travaux et algorithmes présentés dans ce manuscrit ont été développés autour de la bibliothèque JAVA *lattice [Bertet 11].*

La bibliothèque *lattice* contenant l'ensemble des algorithmes de création de treillis à partir d'un contexte binaire, les parties processus de discrétisation, classification d'objets et simplification de la structure ont été développées comme surcouche de la bibliothèque *lattice.*

Par exemple, nous avons défini les structures de données adaptées à la gestion des tables de données quantitatives et définies les méthodes de discrétisation évaluées dans ce manuscrit. Nous avons défini des treillis étiquetés héritant des treillis définis dans la bibliothèque *lattice,* permettant ainsi leur utilisation en classification supervisée. La méthode de navigation mise en œuvre dans Navigala [Guillas 07], a été adaptée selon les propriétés des attributs construits *via* la discrétisation locale que nous avons définie. De plus, pour permettre une comparaison directe avec les arbres de classification (*i.e.* sans mortification du format des données), un package *DecisionTree* a été développé.

L'ensemble de ces fonctionnalités a été rendu accessible *via* une interface graphique. Finalement, nous avons une application complète qui permet, à partir d'un ensemble de données quantitatives étiquetées, de construire un treillis de Galois en utilisant différents types de discrétisation présentées dans le chapitre 5 ; puis de simplifier la structure du treillis construit *via* les méthodes présentées dans le chapitre 6 ; et enfin d'utiliser et d'évaluer les performances du modèle construit en classification, par navigation (*cf.* sections 3.4.2 et 5.2.6.2).

L'application développée étant fortement liée à la méthode Navigala ; elle a été nommée **NAVIGALA 2012** (*cf.* logo Figure E.0.1).

Pour leurs contributions à ces différents développements, nous remercions Najib Laaribi (développement d'une partie des méthodes de discrétisation - stage M2), Gaël Le

FIGURE E.0.1: Application Navigala 2012 - Logo

Baccon (développement du package *DecisionTree* - stage M1), ainsi que Élodie Le Goff (développement de l'interface graphique et du logo - stage L3) et Joffrey Leblay (développement de l'interface graphique - stage L3).

Les Figures E.0.2, E.0.3, E.0.4, E.0.5 présentent les différents onglets de l'application et la Figure E.0.6 présente l'évolution de l'affichage des résultats au cours de l'exécution du programme. Chaque onglet propose le paramétrage des différentes étapes d'apprentissage et de classification.

Remarque. Notons que certains paramètres sont proposés bien qu'ils n'aient pas été utilisés dans nos expérimentations, mais en tant que paramètres ajustables nous avons préféré les proposer à l'utilisateur, nous préconisons cependant l'utilisation des paramètres par défaut.

Les onglets sont ordonnés selon l'ordre des traitements effectués, *i.e.* chargement des données, paramétrage de l'apprentissage, paramétrage de la classification, résultats obtenus en classification. Chaque onglet est décomposé comme suit :

1. Chargement des données à analyser (Figure E.0.2) :
    a) L'utilisateur doit choisir le type d'analyse souhaitée :
        i. Simple : le répertoire d'apprentissage devra contenir un fichier d'apprentissage et un fichier de test ;

ii. Validation croisée : le répertoire d'apprentissage devra contenir n fichiers d'apprentissage et n fichiers de test ; l'utilisateur doit alors saisir la valeur souhaitée pour n ;

iii. Général : le répertoire d'apprentissage devra contenir un unique fichier qui sera divisé en n paquets pour permettre une validation croisée à n paquets ; l'utilisateur doit alors saisir la valeur souhaitée pour n.

b) L'utilisateur doit saisir dans le champ *Données d'apprentissage*, le répertoire contenant le ou les fichiers d'apprentissages et de tests. Et dans le champ *Répertoire de sortie*, le répertoire (vide) dans lequel les résultats devront être enregistrés.

2. Paramétrage de l'apprentissage (Figure E.0.3) :

   a) L'utilisateur doit choisir la discrétisation souhaitée et le critère de coupe associé. Il doit indiquer s'il souhaite que le contexte construit par discrétisation soit enregistré ;

   b) L'utilisateur doit choisir s'il souhaite un élagage du modèle construit ou non et les paramètres associés (Base de validation entre autre).

3. Paramétrage de la classification (Figure E.0.4), cet onglet contient les paramètres ajustables pour lesquels nous conseillons l'utilisation des valeurs par défaut :

   a) Taux d'homogénéité des concepts terminaux : l'utilisateur peut modifier le critère d'étiquetage des concepts terminaux, *i.e.* selon le taux choisi $x\%$; tous les concepts contenant au moins $x\%$ d'objets de la même classe, seront étiquetés avec cette dernière et considérés comme terminaux. Par défaut, seuls les concepts contenant des objets appartenant tous à la même classe et les coatomes sont considérés comme terminaux.

   b) Degré de flou : comme expliqué précédemment (cf. section 5.2.6.2), lorsque Navigala a été défini, les intervalles construit par discrétisation globale étaient élargis *via* des bornes floues. Bien que la discrétisation proposée dans notre modèle ne permette pas de définir des bornes floues sur les intervalles compris entre deux autres intervalles (*cf.* section 5.2.6.2), il est tout de même possible de définir des bornes floues pour les intervalles extérieurs. Par héritage de la méthode Navigala, nous avons donc choisi de proposer l'ajustement de ce paramètre qui n'a que très peu d'influence sur les résultats de classification. Le degré par défaut est *0*.

   c) Validation normalisée : l'utilisateur peut choisir, au cours de la navigation et lorsque deux concepts sont validés, de privilégier le concept pour lequel le taux d'attributs validés est le plus important et non pas uniquement le nombre d'attributs. Ce paramètre a été défini pour normaliser les étapes de navigation, mais comme le précédent il n'a que très peu d'influence sur les

résultats de classification. Le paramétrage par défaut est la *maximisation du nombre d'intervalles validés.*

4. Paramétrage de l'enregistrement des résultats (Figure E.0.5) :
   a) L'utilisateur peut choisir le format d'enregistrement des résultats de la classification (exemple de fichier de résultats au format texte *cf.* Figure E.0.7) ;
   b) Dans la fenêtre d'affichage, les paramètres choisis par l'utilisateur *via* les onglets précédents sont rappelés ;
   c) Lorsque le programme est lancé, l'affichage évolue en fonction des étapes franchies et, les résultats obtenus sont détaillés tels que présentés sur la Figure E.0.6.

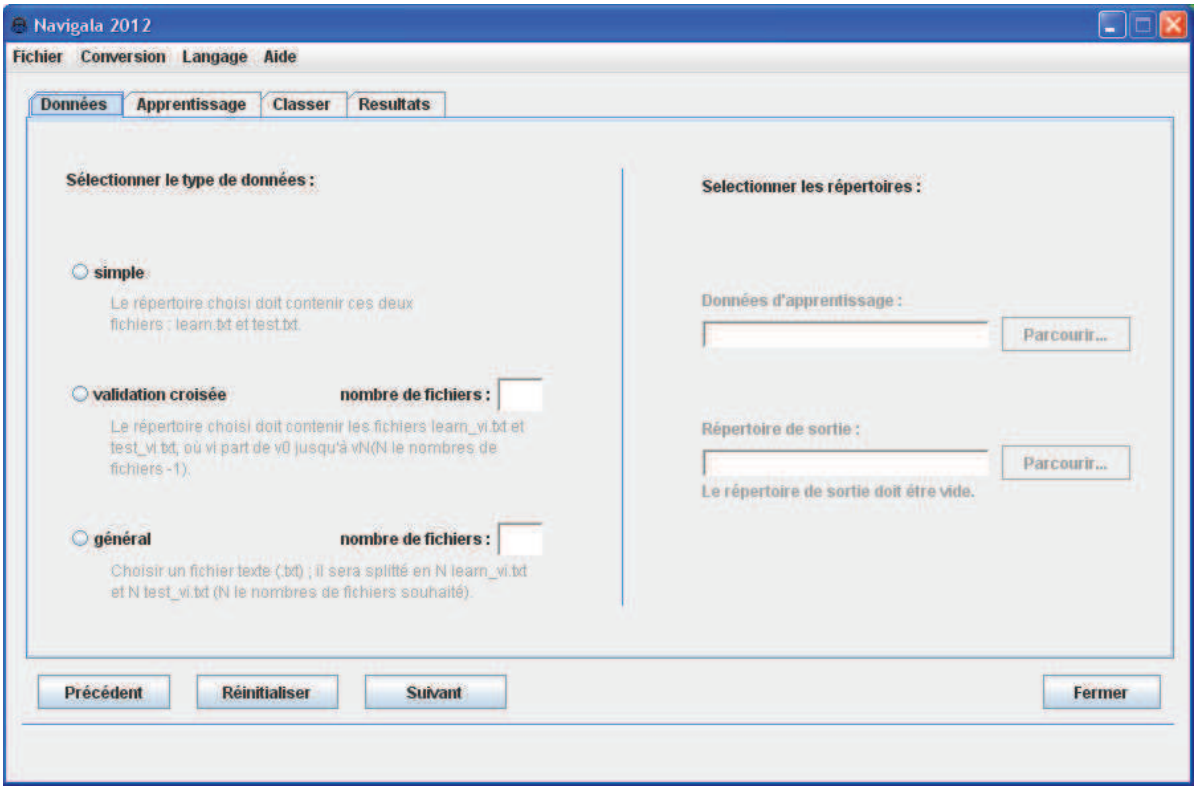


FIGURE E.0.2: Application Navigala2012 - Onglet Données

La Figure E.0.8 présente le format de fichier pris en charge par l'application. Notons que des convertisseurs, à partir des formats issus de logiciels d'analyse de données connus Tanagra [Rakotomalala 05b, Rakotomalala 05a] et Weka [Machine Learning Group , Hall 09] (Waikato Environment for Knowledge Analysis), sont proposés et accessibles par le menu *Conversion.* Ce menu permet d'accéder à la fenêtre de choix des fichiers à convertir (*cf.* Figure E.0.9).

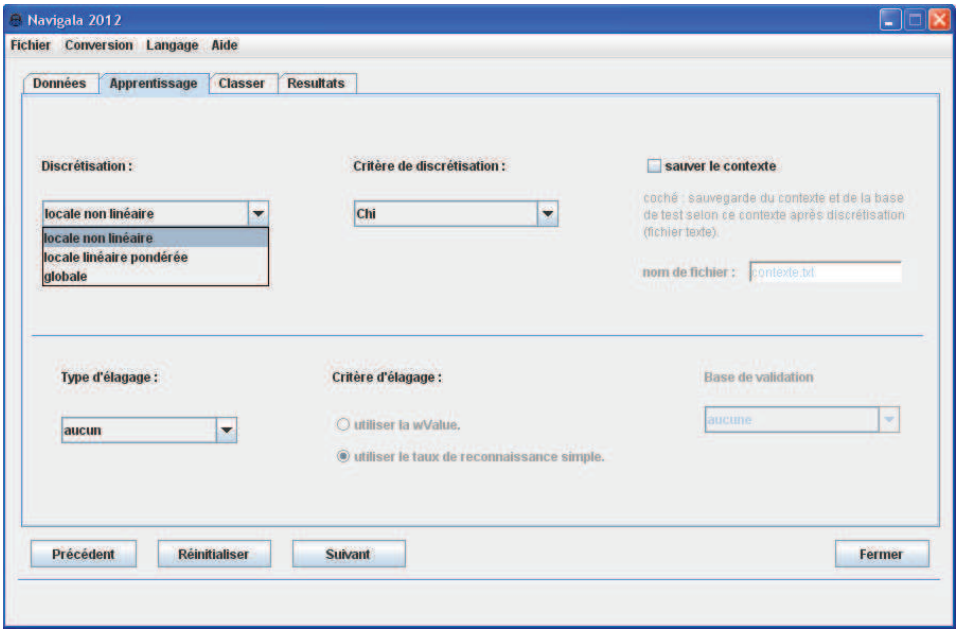


FIGURE E.0.3: Application Navigala2012 - Onglet paramètres d'apprentissage

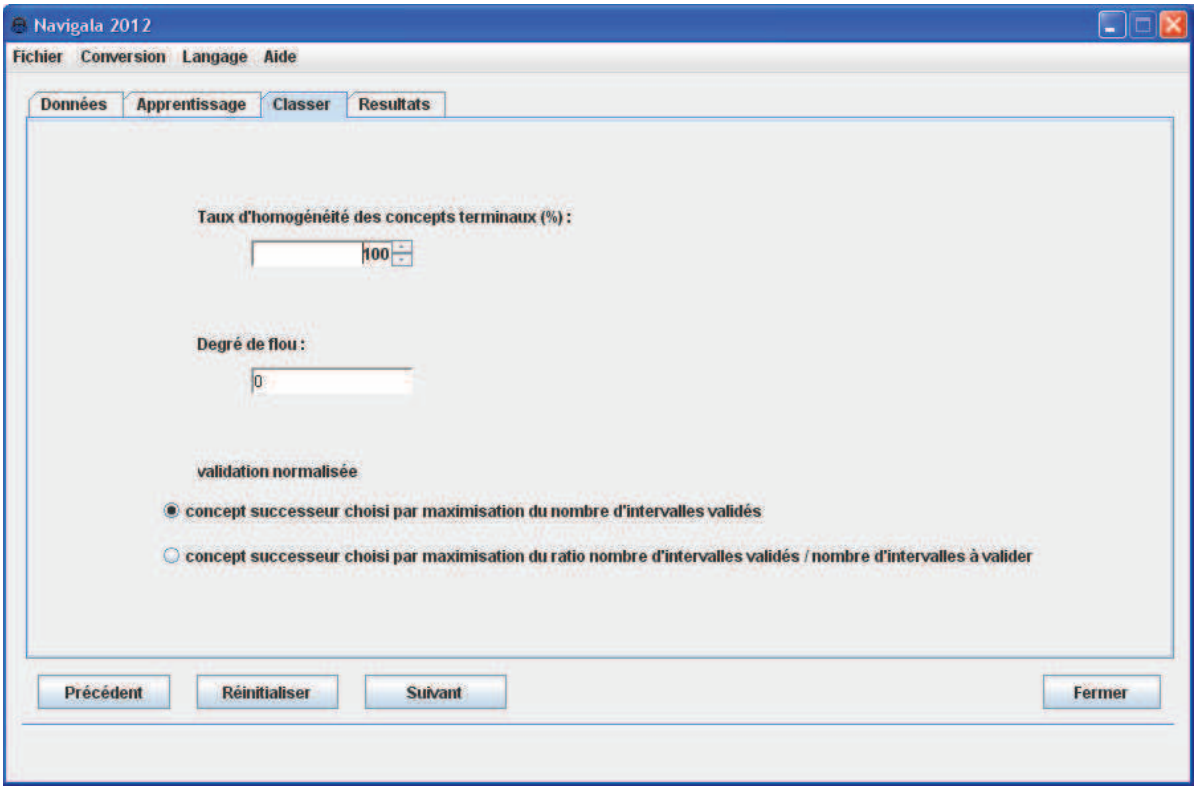


FIGURE E.0.4: Application Navigala2012 - Onglet paramètres de classification

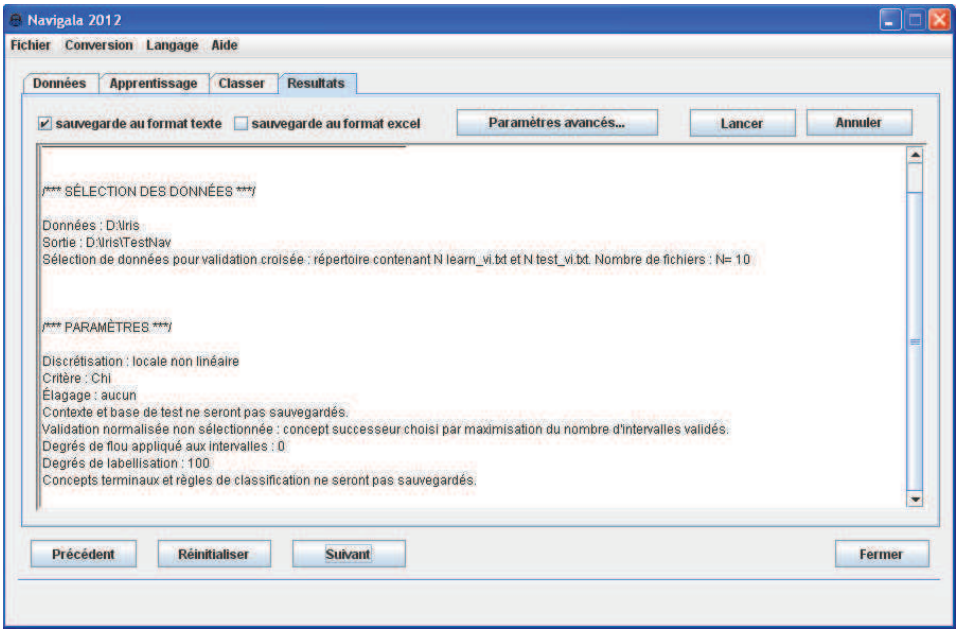


FIGURE E.0.5: Application Navigala2012 - Onglet résultats

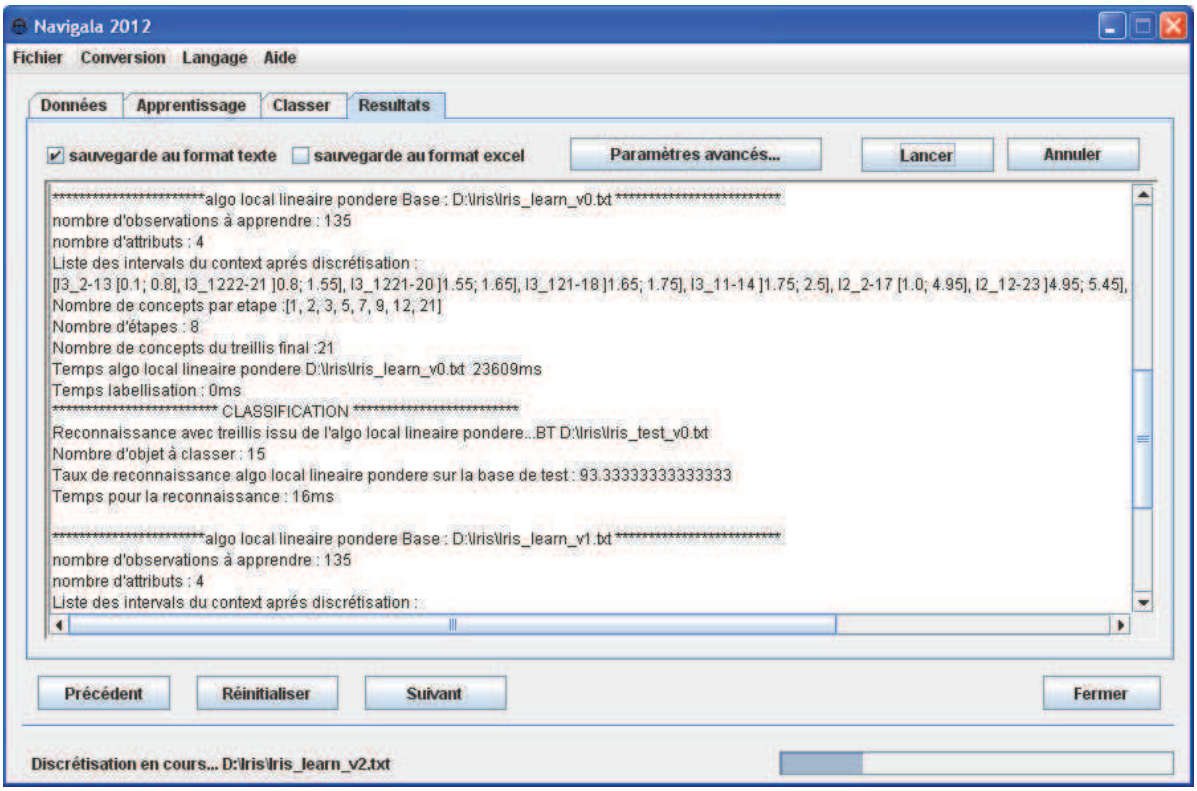


FIGURE E.0.6: Application Navigala2012 - Évolution de l'affiche des résultats

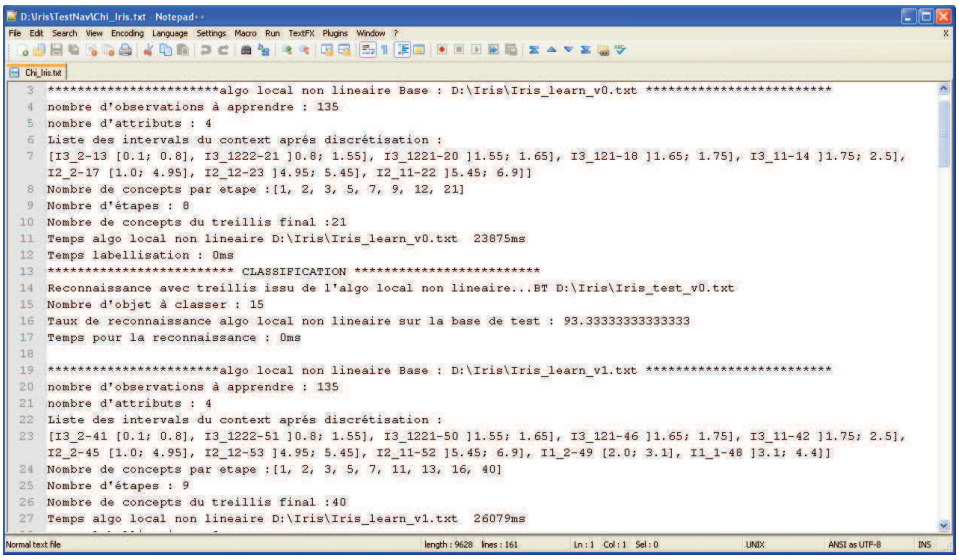


FIGURE E.0.7: Application Navigala2012 - Fichier de résultats - format texte

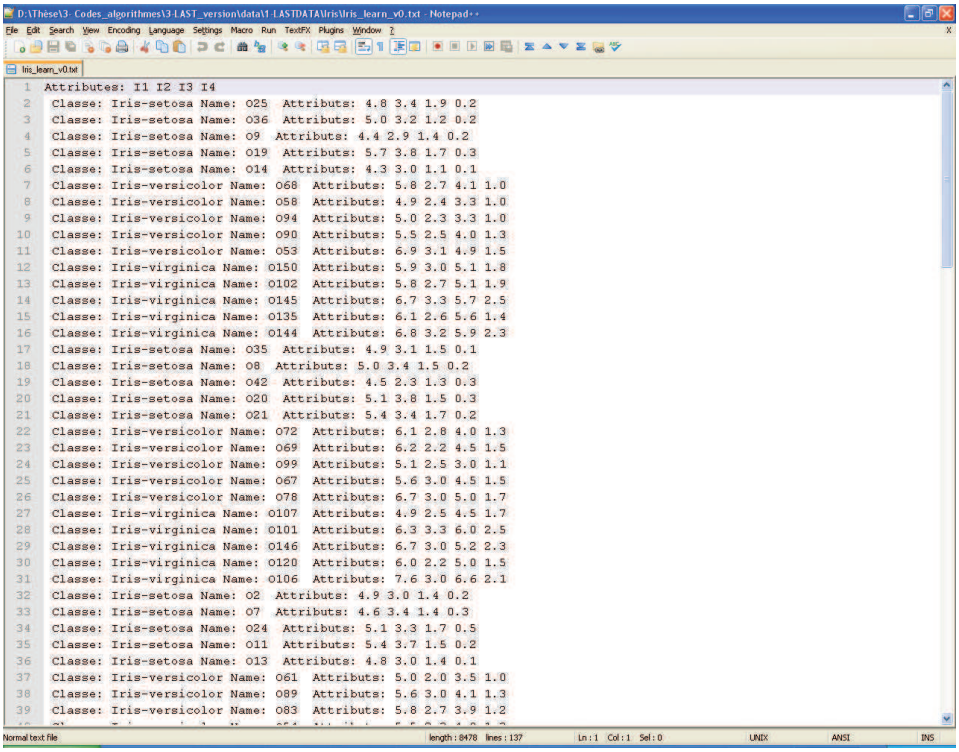


FIGURE E.0.8: Application Navigala2012 - Fichier d'entrées

FIGURE E.0.9: Application Navigala2012 - Fenêtre de conversion

Publications

Article dans des revues avec comité de lecture

N. Girard, K. Bertet & M. Visani. *Dichotomic Lattices & Local Discretization for Lattices*. International Journal of Computer Science and Artificial Intelligence (IJCSAI) accepté mar-2013.[Girard 13]

K. Bertet, M. Visani & **N. Girard**. *Treillis dichotomiques et arbres de décision*. Traitement du Signal, vol. 26, no. 5, pages 407–416, 2009.[Bertet 09]

Chapitres de livres avec comité de lecture

N. Girard, J.-M. Ogier & E. Baudrier. *A New Image Quality Measure Considering Perceptual Information and Local Spatial Feature*. In J.-M. Ogier, W. Liu & J. Lladós, editeurs, Graphics Recognition. Achievements, Challenges, and Evolution, volume 6020 of *Lecture Notes in Computer Science*, pages 242–250. Springer Berlin Heidelberg, 2010.[Girard 10]

E. Baudrier, **N. Girard** & J.-M. Ogier. *A Non-symmetrical Method of Image Local-Difference Comparison for Ancient Impressions Dating*. In W. Liu, J. Lladós & J.-M. Ogier, editeurs, Graphics Recognition. Recent Advances and New Opportunities, pages 257–265. Springer-Verlag, Berlin, Heidelberg, 2008. [Baudrier 08]

Conférences internationales avec comité de lecture

N. Girard, K. Bertet & M. Visani. *Local Discretization of Numerical Data for Galois Lattices*. In IEEE 23rd International Conference on Tools with Artificial Intelligence, ICTAI 2011, pages 902–903, Nov. 2011 - *Best Poster Award*. [Girard 11b]

N. Girard, K. Bertet & M. Visani. *A local discretization of continuous data for lattices : Technical aspects*. In A. Napoli & V. Vychodil, editeurs, Proceedings of the 8th

International Conference on Concept Lattices and Their Applications, pages 409 – 412, Nancy, France, Nov. 2011 (Acceptance rate : 0.57 (27/47)).[Girard 11a]

N. Girard, J.-M. Ogier & E. Baudrier. *A perceptual image quality evaluation based on local spatial information*. In 8th International Workshop on Graphics Recognition - GREC 2009, Lecture notes in computer science, pages 353–358, La Rochelle, France, juillet 2009. IAPR, Springer. [Girard 09b]

S. Guillas, K. Bertet, M. Visani, J.M. Ogier & **N. Girard**. *Some Links Between Decision Tree and Dichotomic Lattice*. In Proc. of the Sixth International Conference on Concept Lattices and Their Applications, pages 193–205. CLA 2008, Oct. 2008 (Acceptance rate : 0.63 (19/30)). [Guillas 08]

Conférences nationales avec comité de lecture

N. Girard, K. Bertet & M. Visani. *Vers une discrétisation locale pour les treillis dichotomiques*. In Actes XVIèmes Rencontres de la Société Francophone de Classification, pages 113–116, Grenoble, France, 2009. [Girard 09a]

Bibliographie

[Agresti 02] A. Agresti. Categorical data analysis. Wiley Series in Probability and Statistics. Wiley-Interscience, 710 p., 2002. www

[Alaoui 93] H. Alaoui. *Algorithmes de construction du treillis de Galois d'une relation binaire*, mars 1993.

[Allen 83] J.F. Allen. *Maintaining knowledge about temporal intervals.* Commun. ACM, vol. 26, no. 11, pages 832–843, November 1983. www

[Anquetil 94] N. Anquetil & J. Vaucher. *Extracting Hierarchical Graphs Of Concepts From An Objects Set : Comparison Of Two Methods.* In Proceedings of the International Conference on Conceptual Structures, pages 26–45, 1994. www

[Antonie 02a] M. Antonie & O. Zaiane. *Classifying Text Documents by Associating Terms with Text Categories.* In Proc. of the Thirteenth Austral-Asian Database Conference (ADC'02), Melbourne, Australia, 2002. www

[Antonie 02b] M. Antonie & O. Zaiane. *Text Document Categorization by Term Association.* In Proc. of IEEE International Conference on Data Mining (ICDM'02), pages 19–26, Maebashi City, Japan, 2002. IEEE Computer Society. www

[Barbut 70] M. Barbut & B. Monjardet. Ordres et classifications : Algèbre et combinatoire. Hachette, Paris, 1970. 2 tomes.

[Baudrier 08] E. Baudrier, N. Girard & J.-M. Ogier. *A Non-symmetrical Method of Image Local-Difference Comparison for Ancient Impressions Dating.* In W. Liu, J. Lladós & J.-M. Ogier, editeurs, Graphics Recognition. Recent Advances and New Opportunities, pages 257–265. Springer-Verlag, Berlin, Heidelberg, 2008. www

[Bayes 63] T. Bayes. *An Essay towards solving a Problem in the Doctrine of Chances.* Philosophical Transactions of the Royal Society of London, vol. 53, pages 370–418, jan. 1763. www

[Belohlávek 07] R. Belohlávek, B. De Baets, J. Outrata & V. Vychodil. *Inducing Decision Trees via Concept Lattices.* In Peter W. Eklund, Jean Diatta & Michel Liquiere, editeurs, CLA, volume 331 of *CEUR Workshop Proceedings*, pages 270–281. CEUR-WS.org, 2007. www

[Belongie 02] S. Belongie, J. Malik & J. Puzicha. *Shape Matching and Object Recognition Using Shape Contexts.* IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 24, no. 4, pages 509–522, April 2002. www

[Ben Ayed Mezghani 10] D. Ben Ayed Mezghani, S. Zribi Boujelbene & N. Ellouze. *Evaluation of SVM Kernels and Conventional Machine Learning Algorithms for Speaker Identification.* International Journal of Hybrid Information Technology, vol. 3, no. 3, July 2010. www

[Ben Tekaya 05] S. Ben Tekaya, S. Ben Yahia & Y. Slimani. *Algorithme de construction d'un treillis des concepts formels et de détermination des générateurs minimaux.* ARIMA Journal, no. Numéro spécial CARI 2004, pages 171–193, Nov. 2005. www

[Bertet 07] K. Bertet, S. Guillas & J.M. Ogier. *Extensions of Bordat's algorithm for attributes.* In Fifth International Conference on Concept Lattices and their Applications (CLA'2007), pages 38–49, Montpellier, France, October 2007. www

[Bertet 09] K. Bertet, M. Visani & N. Girard. *Treillis dichotomiques et arbres de décision.* Traitement du Signal, vol. 26, no. 5, pages 407–416, 2009. www

[Bertet 11] K. Bertet. *Structure de treillis. Contributions structurelles et algorithmiques. Quelques usages pour des données image.* Hdr, Université de La Rochelle, 2011. www

[Birkhoff 34a] G. Birkhoff. *Applications of lattice algebra.* In Mathematical Proceedings of the Cambridge Philosophical Society, volume 30, pages 115–122, 1934. www

[Birkhoff 34b] G. Birkhoff. *On the lattice theory of ideals.* In Bulletin of the Americal Mathematical Society, volume 40, pages 613–619, 1934. www

[Birkhoff 40] G. Birkhoff. Lattice theory. American Mathematical Society, first edition, 1940.

[Birkhoff 48] G. Birkhoff. Lattice theory. American Mathematical Society, second edition, 1948.

[Birkhoff 49] G. Birkhoff. *Théorie et applications des treillis.* vol. 11, no. 5, pages 227–240, 1949. www

[Birkhoff 67] G. Birkhoff. Lattice theory, volume 25. American Mathematical Society, 418 pages, third edition, 1967. www

[Bishop 95] C.M. Bishop. Neural networks for pattern recognition. Oxford University Press, Inc., New York, NY, USA, 1995.

[Boland 07] D.Jr. Boland. Data discretization simplified : Randomized binary search trees for data preprocessing. West Virginia University, 2007. www

[Bordat 86] J.P. Bordat. *Calcul pratique du treillis de Galois d'une correspondance.* Math. Sci. Hum., vol. 96, pages 31–47, 1986. www

[Boser 92] B.E. Boser, I.M. Guyon & V.N. Vapnik. *A training algorithm for optimal margin classifiers.* In Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory, pages 144–152. ACM Press, 1992. www

[Bourbaki 56] N. Bourbaki. Éléments de mathématique. Fascicule XX. Livre I : Théorie des ensembles. Chap. 3, Ensembles ordonnés ; Cardinaux, nombres entiers. (1968, 2ème édition revue et augmentée), Hermann, Paris, 124 p., 1956.

[Bouroche 80] J.M. Bouroche & G. Saporta. L'analyse de données. Que sais-Je ? Presses Universitaires de France, 127 pages, 1980. www

[Bouzouita 06] I. Bouzouita, S. Elloumi & S. Ben-Yahia. *GARCm : Generic Association Rules based Classifier Multi-parameterizable.* In S. Ben-Yahia & E. Mephu-Nguifo, editeurs, Proc. of the Fourth International Conference on Concept Lattices and their Applications (CLA'06), pages 115–125, Yasmine Hammamet, Tunisia, 2006. www

[Breiman 84] L. Breiman, J.H. Friedman, R.A. Olshen & C.J. Stone. Classification and regression trees. Wadsworth Inc., 358 pages, Belmont,California, 1984.

[Breiman 96a] L. Breiman. *Bagging Predictors.* Machine Learning, vol. 24, no. 2, pages 123–140, Aug. 1996. www

[Breiman 96b] L. Breiman. *Technical Note : Some Properties of Splitting Criteria.* Machine Learning, vol. 24, no. 1, pages 41–47, 1996. www

[Breiman 01] L. Breiman. *Random Forests.* Machine Learning, vol. 45, pages 5–32, 2001. 10.1023/A :1010933404324. www

[Breslow 97] L.A. Breslow & D.W. Aha. *Simplifying decision trees : A survey*. Knowl. Eng. Rev., vol. 12, pages 1–40, Jan. 1997. www

[Brin 97] S. Brin, R. Motwani, J.D. Ullman & S. Tsur. *Dynamic itemset counting and implication rules for market basket data*. In Proceedings of the 1997 ACM SIGMOD international conference on Management of data, SIGMOD '97, pages 255–264, New York, NY, USA, 1997. ACM. www

[Buhot 99] A. Buhot. *Etude de propriétés d'apprentissage supervisé et non supervisé par des méthodes de Physique Statistique*. Thèse, Université de Grenoble, 1999. www

[Buntine 92] W. Buntine & T. Niblett. *A Further Comparison of Splitting Rules for Decision-Tree Induction*. Mach. Learn., vol. 8, no. 1, pages 75–85, January 1992. www

[Burges 98] C.J.C. Burges. *A Tutorial on Support Vector Machines for Pattern Recognition*. Data Min. Knowl. Discov., vol. 2, no. 2, pages 121–167, June 1998. www

[Carpineto 93] C. Carpineto & G. Romano. *GALOIS : An Order-Theoretic Approach to Conceptual Clustering*. In ICML'93, pages 33–40, 1993. www

[Carpineto 04] C. Carpineto & G. Romano. Concept data analysis. John Wiley & Sons,Wiley Online Library, 220 p, 2004. www

[Caspard 07] N. Caspard, B. Leclerc & B. Monjardet. Les ensembles ordonnés finis : concepts, résultats et usages. Springer, 2007. 340 pages. www

[Catlett 91] J. Catlett. *On changing continuous attributes into ordered discrete attributes*. In Yves Kodratoff, editeur, Machine Learning — EWSL-91, volume 482 of *Lecture Notes in Computer Science*, pages 164–178. Springer Berlin / Heidelberg, 1991. 10.1007/BFb0017012. www

[Celeux 93] G. Celeux & C. Robert. *Une histoire de discrétisation*. Revue Modulad, vol. 11, no. 40, pages 7–20, 1993. www

[Chandon 81] J.L. Chandon & S. Pinson. Analyse typologique - theories et applications. Masson, 251 pages, 1981. 1er prix de l'Académie des Sciences Commerciales.

[Chein 69] M. Chein. *Algorithme de recherche des sous-matrices premières d'une matrice*. Bull. Math. Soc. Sc. Math. de Roumanie, vol. 1, no. 13, pages 21–25, 1969.

[Chen 09] C.S. Chen, C.W. Yeh & P.Y. Yin. *A novel Fourier descriptor based image alignment algorithm for automatic optical inspection.* Journal of Visual Communication and Image Representation, vol. 20, no. 3, page 178–189, 2009. www

[Choi 06] V. Choi. *Faster Algorithms for Constructing a Concept (Galois) Lattice.* CoRR, vol. abs/cs/0602069, 2006. www

[Codd 93] E.F. Codd, S.B. Codd & C.T. Salley. *Providing OLAP (On-Line Analytical Processing) to User-Analysis : An IT Mandate.* Rapport technique, 1993. www

[Cornuéjols 10] A. Cornuéjols & L. Miclet. Apprentissage artificiel. Numeéro 2. Eyrolles, 803p, 2010. www

[Cornuéjols 02] A. Cornuéjols. *Une nouvelle méthode d'apprentissage : Les SVM. Séparateurs à vaste marge.* Rapport technique Bulletin 51 de l'AFIA (Association Française d'Intelligence Artificielle), juin 2002. www

[Cortes 95] C. Cortes & V. Vapnik. *Support-Vector Networks.* In Machine Learning, pages 273–297, 1995. www

[Coustaty 11a] M. Coustaty. *Contribution à l'analyse complexe de documents anciens Application aux lettrines.* Thèse, Université de La Rochelle, France, 2011. www

[Coustaty 11b] M. Coustaty, K. Bertet, M. Visani & J.M. Ogier. *A New Adaptive Structural Signature for Symbol Recognition by Using a Galois Lattice as a Classifier.* IEEE Transactions on Systems, Man, and Cybernetics, Part B : Cybernetics, vol. 41, no. 4, pages 1136 –1148, aug. 2011. www

[Crawley 73] P. Crawley & R.P. Dilworth. Algebraic theory of lattices. Prentice-Hall, 201 pages, 1973. www

[Dash 11] Raj. Dash, R.L. Paramguru & Ras. Dash. *Comparative Analysis of Supervised and Unsupervised Discretization Techniques.* International Journal of Advances in Science and Technology, vol. 2, no. 3, pages 29 –37, mar. 2011. www

[Dedekind 00] R. Dedekind. *Ueber die von drei Moduln erzeugte Dualgruppe.* Mathematische Annalen, vol. 53, no. 3, pages 371–403, 1900. www

[Derrode 99] S. Derrode, M. Daoudi & F. Ghorbel. *Invariant content-based image retrieval using a complete set of Fourier-Mellin descriptors.* In Multimedia Computing and Systems, 1999. IEEE International Conference on, volume 2, pages 877 – 881 vol.2, jul 1999.

[Do 10] T.N. Do, P. Lenca, S. Lallich & N.K. Pham. *Classifying Very-High-Dimensional Data with Random Forests of Oblique Decision Trees.* In Fabrice Guillet, Gilbert Ritschard, Djamel Zighed & Henri Briand, editeurs, Advances in Knowledge Discovery and Management, volume 292 of *Studies in Computational Intelligence*, pages 39–55. Springer Berlin / Heidelberg, 2010. www

[Domenach 12] F. Domenach, D. Ignatov & J. (eds.) Poelmans. Proceedings of the 10th international conference on formal concept analysis, ICFCA 2012. Lecture Notes in Computer Science, Vol. 7278, 710 p., 75 illus, 2012. www

[Dong 01] M. Dong & R. Kothari. *Classifiability based pruning of decision trees.* In Neural Networks, 2001. Proceedings. IJCNN '01. International Joint Conference on, volume 3, pages 1739 –1743, 2001.

[Dougherty 95] J. Dougherty, R. Kohavi & M. Sahami. *Supervised and Unsupervised Discretization of Continuous Features.* In Machine Learning : Proc. of the Twelfth International Conference, pages 194–202. Morgan Kaufmann, 1995. www

[Droesbeke 05] J.J. Droesbeke, M. Lejeune & G. Saporta. Modèles statistiques pour données qualitatives. Technip, 291p, 2005. www

[Duda 73] R.O. Duda & P.E. Hart. Pattern classification and scene analysis. Wiley-Interscience publication, 482p. Wiley, 1973.

[Esposito 97] F. Esposito, D. Malerba & G. Semeraro. *A Comparative Analysis of Methods for Pruning Decision Trees.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, pages 476–491, 1997.

[Falk 11] I. Falk & C. Gardent. *Combining Formal Concept Analysis and Translation to Assign Frames and Thematic Role to French Verbs.* In Amedeo Napoli & Vilem Vychodil, editeurs, Proceedings of the 8th International Conference on Concept Lattices and Their Applications, pages 223 – 238, Nancy, France, 2011. www

[Fayyad 92] Usama M. Fayyad & Keki B. Irani. *On the Handling of Continuous-Valued Attributes in Decision Tree Generation.* Machine Learning, vol. 8, pages 87–102, 1992. 10.1023/A :1022638503176. www

[Fayyad 93] U.M. Fayyad & K.B. Irani. *Multi-interval discretization of continuous-valued attributes for classification learning.* In

Proc. of the International Joint Conference on Uncertainty in AI, pages 1022–1027. Morgan Kaufman, 1993. www

[Fayyad 96] U.M. Fayyad, G. Piatetsky-Shapiro & P. Smyth. *From Data Mining to Knowledge Discovery : An Overview*. In Advances in Knowledge Discovery and Data Mining, pages 1–34. 1996. www

[Ferré 07] S. Ferré. *CAMELIS : Organizing and Browsing a Personal Photo Collection with a Logical Information System*. In J. Diatta, P. Eklund & M. Liquière, editeurs, Int. Conf. Concept Lattices and Their Applications, volume 331, pages 112–123, Montpellier, France, 2007. www

[Ferré 09] S. Ferré. *Camelis : a logical information system to organize and browse a collection of documents*. International Journal of General Systems - Special Issue : Concept-lattice applications, vol. 38, no. 4, pages 379–403, 2009.

[Fisher 35] R.A. Fisher. The design of experiments. Oliver and Boyd, 260 p., 1935.

[Frank 99] E. Frank & I.H. Witten. *Making better use of global discretization*. In Proc. of 16th International Conference on Machine Learning, Bled, Slovenia, pages 115–123, 1999. www

[Frank 10] A. Frank & A. Asuncion. *UCI Machine Learning Repository*, 2010. `$http://archive.ics.uci.edu/ml$`

[Freedman 81] D. Freedman & P. Diaconis. *On the histogram as a density estimator : L2 theory*. Probability Theory and Related Fields, vol. 57, pages 453–476, 1981. 10.1007/BF01025868. www

[Fu 04a] H. Fu, H. Fu, P. Njiwoua & E.M. Mephu-Nguifo. *A Comparative Study of FCA-Based Supervised Classification Algorithms*. In Concept Lattices, volume LNCS 2961 of *Lecture Notes in Computer Science*, pages 219–220. Springer Berlin / Heidelberg, 2004. www

[Fu 04b] H. Fu & E. Mephu-Nguifo. *Etude et conception d'algorithmes de génération de concepts formels*. Ingénierie des Systèmes d'Information(Revue ISI), vol. 9, no. 3-4, pages 109–132, sep 2004.

[Gansner 00] E.R. Gansner & S.C. North. *An open graph visualization system and its applications to software engineering*. SOFTWARE - PRACTICE AND EXPERIENCE, vol. 30, no. 11, pages 1203–1233, September 2000. www

[Ganter 84] B. Ganter. *Two basic algorithms in concept analysis.* Technische Hochschule Darmstadt (Preprint 831), 1984. www

[Ganter 97] B. Ganter & R. Wille. *Applied Lattice Theory : Formal Concept Analysis.* In Birkhäuser G. Grätzer, editeur, In General Lattice Theory. Preprints, 1997. www

[Ganter 99] B. Ganter & R. Wille. Formal concept analysis, mathematical foundations. Springer Verlag, Berlin, 284 pages, 1999. www

[Ganter 01] B. Ganter & S. Kuznetsov. *Pattern Structures and Their Projections.* In Harry Delugach & Gerd Stumme, editeurs, Conceptual Structures : Broadening the Base, volume 2120 of *Lecture Notes in Computer Science*, pages 129–142. Springer Berlin / Heidelberg, 2001. www

[Ganter 13] B. Ganter, P. Cellier & F. (eds.) Distel. Proceedings of the 11th international conference on formal concept analysis, ICFCA 2013. Lecture Notes in Computer Science, 2013.

[Girard 09a] N. Girard, K. Bertet & M. Visani. *Vers une discrétisation locale pour les treillis dichotomiques.* In Actes XVIèmes Rencontres de la Société Francophone de Classification, pages 113–116, Grenoble, France, 2009. www

[Girard 09b] N. Girard, J.-M. Ogier & E. Baudrier. *A perceptual image quality evaluation based on local spatial information.* In 8th International Workshop on Graphics Recognition - GREC 2009, Lecture notes in computer science, pages 353–358, La Rochelle, France, juillet 2009. IAPR, Springer. www

[Girard 10] N. Girard, J.-M. Ogier & E. Baudrier. *A New Image Quality Measure Considering Perceptual Information and Local Spatial Feature.* In J.-M. Ogier, W. Liu & J. Lladós, editeurs, Graphics Recognition. Achievements, Challenges, and Evolution, volume 6020 of *Lecture Notes in Computer Science*, pages 242–250. Springer Berlin Heidelberg, 2010. www

[Girard 11a] N. Girard, K. Bertet & M. Visani. *A local discretization of continuous data for lattices : Technical aspects.* In A. Napoli & V. Vychodil, editeurs, Proceedings of the 8th International Conference on Concept Lattices and Their Applications, pages 409 – 412, Nancy, France, 2011. www

[Girard 11b] N. Girard, K. Bertet & M. Visani. *Local Discretization of Numerical Data for Galois Lattices.* In IEEE 23rd International Conference on Tools with Artificial Intelligence, ICTAI 2011, pages 902–903, Nov. 2011. www

[Girard 13] N. Girard, K. Bertet & M. Visani. *Dichotomic Lattices & Local Discretization for Lattices*. International Journal of Computer Science and Artificial Intelligence, 2013. accepted.

[Gély 04] A. Gély. *Méthodes de génération du treillis des fermés d'une relation binaire*. In Atelier Evaluation des Algorithmes de Génération de Concepts et de Règles - EGC'04, Clermont-Ferrand, France, 2004. www

[Godin 86a] R. Godin. *L'utilisation de treillis pour l'accès aux systèmes d'information*. Thèse, Université de Montréal,474p, 1986.

[Godin 86b] R. Godin, E. Saunders & J. Gecsei. *Lattice Model of Browsable Data Spaces*. Information Sciences, vol. 40, no. 2, pages 89–116, December 1986. www

[Godin 91] R. Godin, R. Missaoui & H. Alaoui. *Learning algorithms using a Galois lattice structure*. In Tools for Artificial Intelligence, 1991. TAI'91., Third International Conference on, pages 22 –29, nov 1991.

[Godin 93] R. Godin, R. Missaoui & A. April. *Experimental comparison of navigation in a Galois lattice with conventional information retrieval methods*. International Journal of Manmachine Studies, vol. 38, no. 5, pages 747–767, May 1993. www

[Godin 94] R. Godin & R. Missaoui. *An incremental concept formation approach for learning from databases*. Theor. Comput. Sci., vol. 133, no. 2, pages 387–419, October 1994. www

[Godin 95a] R. Godin, G. Mineau, R. Missaoui & H. Mili. *Méthodes de classification conceptuelle basées sur les treillis de Galois et applications*. Revue d'Intelligence Artificielle, vol. 9, no. 2, pages 105–137, 1995. www

[Godin 95b] R. Godin, R. Missaoui & H. Alaoui. *Incremental Concept Formation Algorithms Based on Galois (Concept) Lattices*. Computational Intelligence, vol. 11, no. 2, pages 246–267, 1995. www

[Goralčíková 79] A. Goralčíková & V. Koubek. *A reduct-and-closure algorithm for graphs*. In J. Becvár, editeur, Mathematical Foundations of Computer Science 1979, volume 74 of *Lecture Notes in Computer Science*, pages 301–307. Springer Berlin / Heidelberg, 1979. www

[GREC 03] GREC. *www.cvc.uab.es/grec2003/SymRecContest/index.htm*, 2003. Images base GREC 2003 (Graphics RECognition), revisited : 08-01-2008.

[Guermeur 99] Y. Guermeur & H. Paugam-Moisy. *Théorie de l'apprentissage de Vapnik et SVM, Support Vector Machines*. In Apprentissage Automatique, pages 109–138. Hermès science publications, 1999. www

[Guermeur 07] Y. Guermeur. *SVM Multiclasses, Théorie et Applications*. Hdr, Université Henri Poincaré - Nancy I, November 2007. www

[Guillas 05] S. Guillas, K. Bertet & J.M. Ogier. *Les treillis de Galois : un outil pour la sélection de primitives ?* Traitement du Signal, vol. 22, no. 3, pages 273–291, 2005. www

[Guillas 06a] S. Guillas, K. Bertet & J.M. Ogier. *Comment utiliser le treillis de Galois en reconnaissance d'images ?* In Atelier Extraction de COnnaissance à partir d'Images (ECOI) - Conférence EGC 2006, pages 31–36, 2006. www

[Guillas 06b] S. Guillas, K. Bertet & J.M. Ogier. *A Generic Description of the Concept Lattices' Classifier : Application to Symbol Recognition*. In Liu Wenyin & Josep Lladós, editeurs, Graphics Recognition : Ten Years Review and Future Perspectives - Selected papers from GREC'05, volume 3926, pages 47–60. Lecture Notes in Computer Science, 2006. Revised and extended version of paper first presented at Sixth IAPR International Workshop on Graphics Recognition (GREC'05). www

[Guillas 07] S. Guillas. *Reconnaissance d'objets graphiques détériorés : approche fondée sur un treillis de Galois*. Thèse, Université de La Rochelle, 2007.

[Guillas 08] S. Guillas, K. Bertet, M. Visani, J.M. Ogier & N. Girard. *Some Links Between Decision Tree and Dichotomic Lattice*. In Proc. of the Sixth International Conference on Concept Lattices and Their Applications, pages 193–205. CLA 2008, October 2008. www

[Guénoche 90] A. Guénoche. *Construction du treillis de Galois d'une relation binaire*. Mathématiques et Sciences Humaines, no. 109, pages 41–53, 1990. www

[Hall 09] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann & I.H. Witten. *The WEKA Data Mining Software : An Update*. SIGKDD Explorations, vol. 11, pages 10–19, 2009.

[Hartley 28] R.V. Hartley. *Transmission of information.* Bell System Technological Journal, no. 7, page 535–563, 1928. www

[Hausdorff 78] F. Hausdorff. Grundzüge der mengenlehre. Chelsea Publishing Series. (reprinted and completed 1949,1965,1978) Chelsea Publishing Company, 1978. www

[Ho 97] K.M. Ho & P.D. Scott. *Zeta : A Global Method for Discretization of Continuous Variables.* In David Heckerman, Heikki Mannila, Daryl Pregibon & Ramasamy Uthurusamy, editeurs, KDD-97 Proceedings, page 191. AAAI Press, 1997.

[Hotelling 33a] H. Hotelling. *Analysis of a complex of statistical variables into principal components.* Journal of Educational Psychology, vol. 24, pages 417–441, sep. 1933.

[Hotelling 33b] H. Hotelling. *Analysis of a complex of statistical variables into principal components (continued from September issue).* Journal of Educational Psychology, vol. 24, pages 498–520, oct. 1933.

[Hough 62] P. Hough. *Method and Means for Recognizing Complex Patterns.* U.S. Patent 3.069.654, December 1962.

[Hu 62] M.K. Hu. *Visual pattern recognition by moment invariants.* Information Theory, IRE Transactions on, vol. 8, no. 2, pages 179 –187, february 1962.

[Hwang 02] G. Hwang & F. Li. *A Dynamic Method for Discretization of Continuous Attributes.* In Hujun Yin, Nigel Allinson, Richard Freeman, John Keane & Simon Hubbard, editeurs, Intelligent Data Engineering and Automated Learning — IDEAL 2002, volume 2412 of *Lecture Notes in Computer Science*, pages 181–193. Springer Berlin / Heidelberg, 2002. www

[Johnson 88] D.S. Johnson, M. Yannakakis & C.H. Papadimitriou. *On generating all maximal independent sets.* Information Processing Letters, vol. 27, no. 3, pages 119 – 123, 1988. www

[Kanungo 94] T. Kanungo, R.M. Haralick, H.S. Baird, W. Stuetzle & D. Madigan. *Document Degradation Models : Parameter Estimation and Model Validation.* In IAPR Workshop on machine vision applications, pages 552–557, 1994.

[Kass 80] G.V. Kass. *An Exploratory Technique for Investigating Large Quantities of Categorical Data.* Journal of the Royal Statistical Society. Series C (Applied Statistics), vol. 29, no. 2, pages 119–127, 1980. www

[Kaytoue 09a] M. Kaytoue, S. Duplessis, S. Kuznetsov & A. Napoli. *Two FCA-Based Methods for Mining Gene Expression Data.* In Sébastien Ferré & Sebastian Rudolph, editeurs, Formal Concept Analysis, pages 251–266. LNCS 5548, 2009. $10.1007/978-3-642-01815-2_19$. www

[Kaytoue 09b] M. Kaytoue & A. Napoli. *Classification de données numériques par treillis de concepts et structures de patrons.* In Journées Nationales de l'IA Fondamentale, 2009. www

[Kaytoue 11] M. Kaytoue. *Traitement de données numériques par analyse formelle de concepts et structures de patrons.* Thèse, Université Henri Poincaré - Nancy 1, 2011. www

[Kent Martin 97] J. Kent Martin. *An Exact Probability Metric for Decision Tree Splitting and Stopping.* Machine Learning, vol. 28, no. 2-3, pages 257–291, 1997. www

[Klimushkin 10] M. Klimushkin, S. Obiedkov & C. Roth. *Approaches to the selection of relevant concepts in the case of noisy data.* In Léonard Kwuida & Baris Sertkaya, editeurs, Proc. 8th Intl. Conf. Formal Concept Analysis, volume 5986 of *LNCS/LNAI*, pages 255–266. Springer, 2010. www

[Kohonen 82] T. Kohonen. *Self-Organized Formation of Topologically Correct Feature Maps.* Biological Cybernetics, vol. 43, pages 59–69, 1982. www

[Kohonen 01] T. Kohonen, M. R. Schroeder & T. S. Huang, editeurs. Self-organizing maps. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 3rd edition, 2001.

[Kotsiantis 06] S. Kotsiantis & D. Kanellopoulos. *Discretization Techniques : A recent survey.* GESTS International Transactions on Computer Science and Engineering, vol. 32, no. 1, pages 47–58, 2006. www

[Kotsiantis 07] S.B. Kotsiantis. *Supervised Machine Learning : A Review of Classification Techniques.* Informatica (Slovenia), vol. 31, no. 3, pages 249–268, 2007. www

[Kuznetsov 90] S.O. Kuznetsov. *Stability as an estimate of the degree of substantiation of hypotheses derived on the basis of operational similarity.* Nauchn. Tekh. Inf., no. 12, pages 21–29, 1990. in Russian.

[Kuznetsov 93] S. Kuznetsov. *A fast algorithm for computing all intersections of objects in a Finite semi-lattice.* Automatic Documentation and Mathematic Linguistic, vol. 27, no. 5, pages 11–21, 1993.

[Kuznetsov 01a] S. Kuznetsov. *Machine Learning on the Basis of Formal Concept Analysis*. Autom. Remote Control, vol. 62, no. 10, pages 1543–1564, 2001. www

[Kuznetsov 01b] S. Kuznetsov & S. Obiedkov. *Algorithms for the Construction of Concept Lattices and Their Diagram Graphs*. In Luc De Raedt & Arno Siebes, editeurs, Principles of Data Mining and Knowledge Discovery, volume 2168 of *Lecture Notes in Computer Science*, pages 289–300. Springer Berlin / Heidelberg, 2001. www

[Kuznetsov 02] S. Kuznetsov & S. Obiedkov. *Comparing performance of algorithms for generating concept lattices*. Journal of Experimental and Theoritical Artificial Intelligence, vol. 14, no. 2-3, pages 189–216, 2002. www

[Kuznetsov 03] S. Kuznetsov. *On stability of a formal concept*. In Eric San Juan Alain Sigayret Mohamed Nadif Amedeo Napoli, editeur, Fourth International Conference on Knowledge Discovery and Discrete Mathematics - Journées de l'informatique Messine - JIM'2003, pages 295–305, Metz, France, September 2003. INRIA.

[Kuznetsov 04] S. Kuznetsov. *Machine Learning and Formal Concept Analysis*. In ICFCA'04, pages 287–312, 2004. www

[Kuznetsov 07a] S. Kuznetsov. *On stability of a formal concept*. Annals of Mathematics and Artificial Intelligence, vol. 49, pages 101–115, 2007. 10.1007/s10472-007-9053-6. www

[Kuznetsov 07b] S. Kuznetsov, S. Obiedkov & C. Roth. *Reducing the Representation Complexity of Lattice-Based Taxonomies*. In U. Priss, S. Polovina & R. Hill, editeurs, Proc. of ICCS 15th Intl Conf Conceptual Structures, volume 4604 of *LNCS/LNAI*, pages 241–254. Springer, 2007. www

[Lawler 80] E.L. Lawler, J.K. Lenstra & A.H.G. Rinnooy Kan. *Generating all Maximal Independent Sets : NP-Hardness and Polynomial-Time Algorithms*. SIAM Journal on Computing, vol. 9, pages 558–565, 1980.

[Lim 00] T.S. Lim, W.Y. Loh & Y.S. Shih. *A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-Three Old and New Classification Algorithms*. Mach. Learn., vol. 40, no. 3, pages 203–228, September 2000. www

[Liquière 90] M. Liquière & E. Mephu-Nguifo. *LEGAL : LEarning with GAlois Lattice*. In Actes des Journées Françaises sur l'Ap-

prentissage (JFA), pages 93–113, Lannion, France, April 1990.

[Liu 02] H. Liu, F. Hussain, C.L. Tan & M. Dash. *Discretization : An Enabling Technique*. Data Min. Knowl. Discov., vol. 6, pages 393–423, October 2002. www

[Lladós 02] J. Lladós, E. Valveny, G. Sánchez & E. Martí. *Symbol Recognition : Current Advances and Perspectives*. In Selected Papers from the Fourth International Workshop on Graphics Recognition Algorithms and Applications, GREC '01, pages 104–127, London, UK, UK, 2002. Springer-Verlag. www

[Loh 09] W.Y. Loh. *Improving the precision of classification trees*. The Annals of Applied Statistics, vol. 3, pages 1710–1737, 2009. www

[Loncaric 98] S. Loncaric. *A Survey of Shape Analysis Techniques*. Pattern Recognition, vol. 31, pages 983–1001, 1998. www

[Lowe 99] D.G. Lowe. *Object recognition from local scale-invariant features*. In Proceeding of the International Conference on Computer Vision, page 1150–1157, sep. 1999. www

[Machine Learning Group] Machine Learning Group. *Weka*. Rapport technique. URL http ://www.cs.waikato.ac.nz/~ml/weka. www

[Malerba 96] D. Malerba, F. Esposito & G. Semeraro. *A Further Comparison of Simplification Methods for Decision-Tree Induction*. In In D. Fisher and H. Lenz (Eds.), Learning, pages 365–374. Springer-Verlag, 1996. www

[Mansour 97] Y. Mansour. *Pessimistic decision tree pruning based Continuous-time*. In Proceedings of the Fourteenth International Conference on Machine Learning, ICML '97, pages 202–210, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc. www

[Markowetz 03] F. Markowetz. *Practical DNA Microarray Analysis*. Rapport technique, 2003. www

[Mas 10] J. Mas, J. Llados, G. Sanchez & J.A.P. Jorge. *A syntactic approach based on distortion-tolerant Adjacency Grammars and a spatial-directed parser to interpret sketched diagrams*. Pattern Recogn., vol. 43, no. 12, pages 4148–4164, December 2010. www

[McCulloch 88] W.S. McCulloch & W. Pitts. *Neurocomputing : foundations of research*. chapitre A logical calculus of the ideas immanent in nervous activity, pages 15–27. MIT Press, Cambridge, MA, USA, 1988. www

[Mephu-Nguifo 00] E. Mephu-Nguifo & P. Njiwoua. *GLUE : A Lattice-based Constructive Induction System.* Intl. Journ. of Intelligent Data Analysis (IDA), vol. 4, no. 4, pages 1–49, 2000.

[Mephu-Nguifo 05] E. Mephu-Nguifo & P. Njiwoua. *Treillis de concepts et classification supervisée.* Technique et Science Informatiques(TSI), vol. 24, no. 4, pages 449–488, 2005. www

[Mineau 00] G.W. Mineau, A. Bissoon & R. Godin. *Simple pre- and post-pruning techniques for large conceptual clustering structures.* vol. 4, pages 1–20, 2000. www

[Mingers 87] J. Mingers. *Expert Systems - Rule Induction with Statistical Data.* Journal of the Operations Research Society, vol. 38, no. 1, pages 39–47, 1987. www

[Mingers 89a] J. Mingers. *An Empirical Comparison of Pruning Methods for Decision Tree Induction.* Machine Learning, vol. 4, pages 227–243, 1989. 10.1023/A :1022604100933. www

[Mingers 89b] J. Mingers. *An Empirical Comparison of Selection Measures for Decision-Tree Induction.* Mach. Learn., vol. 3, no. 4, pages 319–342, March 1989. www

[Moore 09] E.H. Moore. *On a form of general analysis with applications to linear differential and integral equations.* In Atti del IV Congr. Int. dei Mat. II, pages 98–114, 1909.

[Morgan 63] J. N. Morgan & J. A. Sonquist. *Problems in the Analysis of Survey Data, and a Proposal.* Journal of the American Statistical Association, vol. 58, no. 302, Jun. 1963. article consists of 20 pages. www

[Muhlenbach 02] F. Muhlenbach & R. Rakotomalala. *Multivariate supervised discretization, a neighborhood graph approach.* In IEEE International Conference on Data Mining, Proceedings., pages 314–321. IEEE Comput. Soc, 2002.

[Muhlenbach 05] F. Muhlenbach & R. Rakotomalala. *Discretization of Continuous Attributes.* In Idea Group Reference, editeur, Encyclopedia of Data Warehousing and Mining, pages 397–402. John Wang, 2005. www

[Nakache 03] J.P. Nakache & J. Confais. Statistique explicative appliquée : analyse discriminante, modèle logistique, segmentation par arbre. Technip, 282p, 2003.

[Nguyen 12] V.A. Nguyen & A. Yamamoto. *Learning from graph data by putting graphs on the lattice.* Expert Systems with Applications, vol. 39, no. 12, pages 11172 – 11182, 2012. www

[Niblett 87] T. Niblett & I Bratko. *Learning decision rules in noisy domains*. In Proceedings of Expert Systems '86, The 6Th Annual Technical Conference on Research and development in expert systems III, pages 25–34, New York, NY, USA, 1987. Cambridge University Press. www

[Nijssen 07] S. Nijssen & E. Fromont. *Mining optimal decision trees from itemset lattices*. In Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '07, pages 530–539, New York, NY, USA, 2007. ACM. www

[Nijssen 10] Siegfried Nijssen & Elisa Fromont. *Optimal constraint-based decision tree induction from itemset lattices*. Data Mining and Knowledge Discovery, vol. 21, no. 1, pages 9–51, April 2010.

[Njiwoua 97] P. Njiwoua & E. Mephu-Nguifo. *GLUE : un Système d'Apprentissage à base de Treillis qui utilise la fonction d'Entropie*. In Actes des Cinquièmes rencontres de la Société Francophone de Classification (SFC-97), pages 397–400, Lyon, France, sep 1997. Université Lumière.

[Njiwoua 99] P. Njiwoua & E. Mephu-Nguifo. *Améliorer l'apprentissage à partir d'instances grâce à l'induction de concepts : le système CIBLe*. Revue d'intelligence Artificielle (RIA), vol. 13, no. 2, pages 413–440, 1999. www

[Norris 78] E.M. Norris. *An Algorithm for Computing the Maximal Rectangles in a Binary Relation*. Revue Roumaine de Mathématiques Pures et Appliquées, vol. 23, no. 2, pages 243–250, 1978.

[Nourine 99] L. Nourine & O. Raynaud. *A fast algorithm for building lattices*. Information Processing Letters, vol. 71, no. 5–6, pages 199 – 204, 1999. www

[Oosthuizen 88] G.D. Oosthuizen. *The use of a Lattice in Knowledge Processing*. Thèse, University of Strathclyde, Glasgow, 1988.

[Ore 44] O. Ore. *Galois Connexions*. Transactions of the American Mathematical Society, vol. 55, no. 3, pages 493–513, May 1944. 21 pages. www

[Patil 10] D.D. Patil, V.M. Wadhai & J.A. Gokhale. *Evaluation of Decision Tree Pruning Algorithms for Complexity and Classification Accuracy*. International Journal of Computer Applications, vol. 11, no. 2, pages 23–30, December 2010. Published By Foundation of Computer Science. www

[Pearson 01] K. Pearson. *On lines and planes of closest fit to systems of points in space.* Philosophical Magazine, vol. 2, pages 559–572, 1901. www

[Pearson 04] K. Pearson. On the theory of contingency and its relation to association and normal correlation. Drapers' company research memoirs : Biometric series. Cambridge University Press, 35p, 1904. www

[Peirce 01] C.S. Peirce. *On lines and planes of closest fit to systems of points in space.* Philosophical Magazine, vol. 2, pages 559–572, 1901. www

[Pfaltz 07] J. Pfaltz. *Representing numeric values in concept lattices.* In J. Diatta, P. Eklund & M. Liquiere, editeurs, Proc. of the 5th International Conference on Concept Lattices and Their Applications, pages 260–269, 2007. www

[Poelmans 10] J. Poelmans, P. Elzinga, S. Viaene & G. Dedene. *Formal concept analysis in knowledge discovery : a survey.* In Proceedings of the 18th international conference on Conceptual structures : from information to intelligence, ICCS'10, pages 139–153, Berlin, Heidelberg, 2010. Springer-Verlag. www

[Polaillon 98] G. Polaillon. *Organisation et interprétation par les treillis de Galois de données de type multivalué, intervalle ou histogramme.* Thèse, Université Paris IX-Dauphine, 1998. www

[Priss 08] U. Priss. *FCA Software Interoperability.* In Proceedings of the Sixth International Conference on Concept Lattices and Their Applications (CLA'08), pages 133–144, 2008. www

[Prum 10] S. Prum, M. Visani & J.M. Ogier. *On-line Handwriting word recognition using a bi-character model.* In 20th International Conference on Pattern Recognition, pages 2700 – 2703. ICPR 2010, August 2010. www

[Quinlan 79] J.R. Quinlan. *Discovering rules by induction from large collections of examples.* Expert systems in the micro electronic age, pages 168–201, 1979.

[Quinlan 86a] J.R. Quinlan. *Induction of Decision Trees.* Machine Learning, vol. 1, 1986. www

[Quinlan 86b] J.R. Quinlan. *Simplifying Decision Trees.* 1986. www

[Quinlan 87a] J.R. Quinlan. *Generating production rules from decision trees.* In Proceedings of the 10th international joint conference on Artificial intelligence - Volume 1, IJCAI'87, pages 304–307, San Francisco, CA, USA, 1987. Morgan Kaufmann Publishers Inc. www

[Quinlan 87b] J.R. Quinlan. *Simplifying Decision Trees.* International Journal of Man-Machine Studies - Special Issue : Knowledge Acquisition for Knowledge-based Systems., vol. 27, pages 221–234, sep 1987.

[Quinlan 90] J.R. Quinlan. *Decision trees and decisionmaking.* IEEE Transactions on Systems, Man and Cybernetics, vol. 20, no. 2, pages 339–346, 1990. www

[Quinlan 93] J.R. Quinlan. C4.5 : Programs for machine learning. Morgan Kaufman, Los Altos, California, 302 pages, 1993. www

[Quinlan 96a] J.R. Quinlan. *Bagging, boosting and C4.5.* In Proc. of the 13th American Association National Conference on Artificial Intelligence, pages 725–730. AAAI Press, Menlo Park, CA, 1996. www

[Quinlan 96b] J.R. Quinlan. *Improved Use of Continuous Attributes in C4.5.* Journal of Artificial Intelligence Research, vol. 4, pages 77–90, 1996. www

[Quinlan 99] J.R. Quinlan. *Simplifying decision trees.* Int. J. Hum.-Comput. Stud., pages 497–510, 1999.

[Rabaséda 96] S. Rabaséda, R. Rakotomalala & M. Sebban. *A comparison of some contextual discretization methods.* Information Sciences, vol. 92, no. 1–4, pages 137 – 157, 1996. www

[Radon 17] J. Radon. *Über die Bestimmung von Funktionen durch ihre Integralwerte längs gewisser Mannigfaltigkeiten.* Berichte Saechsische Akademie der Wissenschaften, vol. 69, pages 262–277, 1917. www

[Rakotomalala 96] R. Rakotomalala & E. Chettouh. *Extraction de règles par validation dans les graphes d'induction.* In Actes des Quatrièmes Rencontres de la Société Francophone de Classification, Sep. 1996.

[Rakotomalala 97] R. Rakotomalala. *Graphe d'induction.* Thèse, Université Claude Bernard - Lyon I, 1997. www

[Rakotomalala 05a] R. Rakotomalala. *Arbres de décision.* Revue Modulad, vol. 33, pages 163–187, 2005. www

[Rakotomalala 05b] R. Rakotomalala. *TANAGRA : un logiciel gratuit pour l'enseignement et la recherche.* In Actes de EGC'2005, RNTI-E-3, volume 2, pages 697–702, 2005. www

[Rish 01] I. Rish. *An empirical study of the naive Bayes classifier.* IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence, vol. 3, no. 22, page 41–46, 2001. www

[Rokach 05] L. Rokach & O. Maimon. *Top-down induction of decision trees classifiers - a survey*. Systems, Man, and Cybernetics, Part C : Applications and Reviews, IEEE Transactions on, vol. 35, no. 4, pages 476 – 487, nov. 2005. www

[Rokach 08] L. Rokach & O.Z. Maimon. Data mining with decision trees : Theory and applications, volume 69 of *Series in Machine Perception and Artificial Intelligence*. World Scientific, 244p, 2008. www

[Roth 06] C. Roth, S.A. Obiedkov & D.G. Kourie. *Towards Concise Representation for Taxonomies of Epistemic Communities*. In CLA, pages 240–255, 2006. www

[Roth 08] C. Roth, S.A. Obiedkov & D.G. Kourie. *On succinct representation of knowledge community taxonomies with formal concept analysis*. International Journal of Foundations of Computer Science (IJFCS), vol. 19, pages 383–404, 2008. www

[Rusiñol 10] Marçal Rusiñol & Josep Lladós. *Efficient logo retrieval through hashing shape context descriptors*. In Proceedings of the 9th IAPR International Workshop on Document Analysis Systems, DAS '10, pages 215–222, New York, NY, USA, 2010. ACM. www

[Rusiñol 06] M. Rusiñol & J. Lladós. *Symbol Spotting in Technical Drawings Using Vectorial Signatures*. In W. Liu & J. Lladós, editeurs, Graphics Recognition. Ten Years Review and Future Perspectives, volume 3926 of *Lecture Notes in Computer Science*, pages 35–46. Springer Berlin / Heidelberg, 2006. www

[Sahak 09] R. Sahak, W. Mansor, L.Y. Khuan, A. Yassin, A. Zabidi & F.Y.A. Rahman. *Choice for a support vector machine kernel function for recognizing asphyxia from infant cries*. In Industrial Electronics Applications, 2009. ISIEA 2009. IEEE Symposium on, volume 2, pages 675 –678, oct. 2009.

[Sahami 95] M. Sahami. *Learning Classification Rules Using Lattices*. In Nada Lavrac & Stefan Wrobel, editeurs, Proc. of European Conference on Machine Learning, ECML'95, pages 343–346, Heraclion, Crete, Greece, April 1995. www

[Saporta 90] G. Saporta. Probabilités, analyse des données et statistiques. Editions Technip, 493 pages, 1990.

[Saporta 06] G. Saporta. Probabilités, analyse des données et statistiques 2e édition. Editions Technip, 622 pages, 2006. www

[Schröder 90] E. Schröder, J. Lüroth & K.E. Müller. Vorlesungen über die algebra der logik : exakte logik. Vorlesungen über die algebra der logik : Exakte logik. B. G. Teubner, 1890.

[Schröder 95] E. Schröder, J. Lüroth & K.E. Müller. Vorlesungen über die algebra der logik : exakte logik. Numeéro vol. 3,ptie. 1 in Vorlesungen über die algebra der logik : exakte logik. B. G. Teubner, 1895.

[Scott 79] David W. Scott. *On Optimal and Data-Based Histograms*. Biometrika, vol. 66, no. 3, pages pp. 605–610, 1979. www

[Shannon 49] C.E. Shannon & W. Weaver. The mathematical theory of communication, volume 27 of *The Mathematical Theory of Communication*. University of Illinois Press, 117p, 1949. www

[Singhal 09] N. Singhal, Y.Y. Lee, C.S. Kim & S.U. Lee. *Robust image watermarking using local Zernike moments*. Journal of Visual Communication and Image Representation, vol. 20, no. 6, page 408–419, 2009. www

[Sonquist 71] J.A. Sonquist, E.L. Baker & J.N. Morgan. Searching for structure. Institute for Social Research, University of Michigan,287 p., 1971.

[Stumme 01] G. Stumme, R. Taouil, Y. Bastide & L. Lakhal. *Conceptual Clustering with Iceberg Concept Lattices*. In R. Klinkenberg, S. Rüping, A. Fick, N. Henze, C. Herzog, R. Molitor & O. Schröder, editeurs, Proc. GI-Fachgruppentreffen Maschinelles Lernen (FGML'01), Universität Dortmund 763, October 2001.

[Sturges 26] H.A. Sturges. *The Choice of a Class Interval*. Journal of the American Statistical Association, vol. 21, no. 153, pages 65–66, March 1926.

[Szathmary L. 12] Priss U. (Eds.) Szathmary L. Proceedings of the 9th international conference on concept lattices and their applications, CLA 2012. 2012.

[Tabbone 03a] S. Tabbone & L. Wendling. *Adaptation de la transformée de Radon pour la recherche d'objets à niveaux de gris et de couleurs*. Technique et Science Informatiques, pages 1139–1166, 2003.

[Tabbone 03b] S. Tabbone, L. Wendling & K. Tombre. *Matching of Graphical Symbols in Line-Drawing Images Using Angular Signature Information*. International Journal On Document

Analysis and Recognition, vol. 6, no. 2, pages 115–125, 2003. www

[Teague 80] M.R. Teague. *Image analysis via the general theory of moments*. Journal of the Optical Society of America, vol. 70, no. 8, pages 920–930, 1980. www

[Trier 96] O. Trier, A. Jain & T. Taxt. *Feature extraction methods for character recognition-A survey*. Pattern Recognition, vol. 29, no. 4, pages 641 – 662, 1996. www

[Tschuprow a] A.A. Tschuprow. *On the Mathematical Expectation of the Moments of Frequency Distributions in the Case of Correlated Obsevations*. volume II, pages 461–493. Metron.

[Tschuprow b] A.A. Tschuprow. *On the Mathematical Expectation of the Moments of Frequency Distributions in the Case of Correlated Obsevations*. volume II, pages 646–683. Metron.

[Tufféry 07] S. Tufféry. Data mining et statistique décisionnelle : l'intelligence des données. Éditions Technip, 533 pages, 2007.

[Tufféry 10] S. Tufféry. Data mining et statistique décisionnelle : l'intelligence dans les bases de données. Editions Technip, 705 pages, 2010. www

[Tzimiropoulos 09] G. Tzimiropoulos, N. Mitianoudis & T. Stathaki. *A unifying approach to moment-based shape orientation and symmetry classification*. Trans. Img. Proc., vol. 18, no. 1, pages 125–139, January 2009. www

[Valtchev 03] P. Valtchev, D. Grosser, C. Roume & M.R. Hacene. *Galicia : An Open Platform for Lattices*. In In Using Conceptual Structures : Contributions to the 11th Intl. Conference on Conceptual Structures (ICCS'03, pages 241–254. Shaker Verlag, 2003. www

[Van de Merckt 93] T. Van de Merckt. *Decision Trees in Numerical Attribute Spaces*. In Proceedings of the 13th International Joint Conference on Artificial Intelligence, pages 1016–1021, 1993.

[Vapnik 63] V.N. Vapnik & A. Lerner. *Pattern Recognition using Generalized Portrait Method*. Automation and Remote Control, vol. 24, no. 6, pages 774–780, 1963.

[Vapnik 71] V.N. Vapnik & A.Y. Chervonenkis. *On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities*. Theory of Probability and its Applications, vol. 16, no. 2, pages 264–280, 1971. www

[Vapnik 95] V.N. Vapnik. The nature of statistical learning theory. Springer-Verlag New York, Inc., New York, NY, USA, 1995. www

[Vapnik 98] V.N. Vapnik. Statistical learning theory. John Wiley and Sons, Inc., New York, 1 edition, 1998.

[Visani 11] M. Visani, K. Bertet & J.M. Ogier. *Navigala : an Original Symbol Classifier Based on Navigation through a Galois Lattice*. International Journal of Pattern Recognition and Artificial Intelligence, IJPRAI, vol. 25, no. 4, pages 449–473, 2011.

[Wang 05] J. Wang & G. Karypis. *HARMONY : Efficiently mining the best rules for classification*. In Proc. of the International Conference of Data Mining (ICDM'05), Newport Beach, CA, 2005. www

[Weiss 91] S. M. Weiss & C.A. Kulikowski. Computer systems that learn : classification and prediction methods from statistics, neural nets, machine learning, and expert systems. M. Kaufmann Publishers, San Mateo, Calif. :, 1991. www

[Wille 82] R. Wille. *Restructuring lattice theory : an approach based on hierarchies of concepts*. Ordered sets, pages 445–470, 1982. I. Rival (ed.), Dordrecht-Boston, Reidel.

[Wu 07] X. Wu, V. Kumar, J.R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J. McLachlan, A. Ng, B. Liu, P.S. Yu, Z.H. Zhou, M. Steinbach, D.J. Hand & D. Steinberg. *Top 10 algorithms in data mining*. Knowl. Inf. Syst., vol. 14, no. 1, pages 1–37, December 2007. www

[Yang 08] M. Yang, K. Kpalma & J. Ronsin. *A Survey of Shape Feature Extraction Techniques*. Pattern Recognition Techniques, Technology and Applications, 2008. www

[Yevtushenko 00] S.A. Yevtushenko. *System of data analysis "Concept Explorer"*. In Proceedings of the 7th national conference on Artificial Intelligence KII-2000, pages 127–134, 2000. In Russian.

[Yevtushenko 04] S.A. Yevtushenko. *Computing and Visualizing Concept Lattices*. Thèse, Fachbereich Informatik der Technischen Universität Darmstadt, 2004. www

[Zighed 96] D.A. Zighed, R. Rakotomalala & S. Rabaséda. *A discretization method of continuous attributes in induction graphs*. In Proceedings of the Thirteenth European Meeting on Cybernetics and Systems Research, pages 997–1002, 1996.

[Zighed 97] D.A. Zighed, R. Rakotomalala & F. Feschet. *Optimal multiple intervals discretization of continuous attributes for supervised learning*. In David Heckerman, Heikki Mannila, Daryl Pregibon & Ramasamy Uthurusamy, editeurs, Proceedings of the Third International Conference on Knowledge Discovery and Data Mining-KDD-97, pages 295–298. AAAI Press, 1997. www

[Zighed 00] D.A. Zighed & R. Rakotomalala. Graphes d'induction - apprentissage et data mining. Hermes, 2000.

[Zighed 02] D.A. Zighed & R. Rakotomalala. Extraction de connaissances à partir de données (ecd), volume H 3 744. Techniques de l'Ingénieur, 2002. www

[Zighed 07] D.A. Zighed, S. Marcellin & G. Ritschard. *Mesure d'entropie asymétrique et consistante*. In EGC, pages 81–86, 2007. www

[Zuwala 06] D. Zuwala. *Reconnaissance de symboles sans connaissance a priori*. Thèse, Institut National Polytechnique de Lorraine, France, 2006. www

Vers une approche hybride mêlant arbre de classification et treillis de Galois pour de l'indexation d'images

Résumé :

La classification d'images s'articule généralement autour des deux étapes que sont l'étape d'extraction de signatures suivie de l'étape d'analyse des données extraites, ces dernières étant généralement quantitatives. De nombreux modèles de classification ont été proposés dans la littérature, le choix du modèle le plus adapté est souvent guidé par les performances en classification ainsi que la lisibilité du modèle. L'arbre de classification et le treillis de Galois sont deux modèles symboliques connus pour leur lisibilité. Dans sa thèse [Guillas 07], Guillas a utilisé efficacement les treillis de Galois pour la classification d'images et des liens structurels forts avec les arbres de classification ont été mis en évidence. Les travaux présentés dans ce manuscrit font suite à ces résultats, et ont pour but de définir un modèle hybride entre ces deux modèles, qui réunisse leurs avantages (leur lisibilité respective, la robustesse du treillis et le faible espace mémoire de l'arbre). A ces fins, l'étude des liens existants entre les deux modèles, a permis de mettre en avant leurs différences. Tout d'abord, le type de discrétisation, les arbres utilisent généralement une discrétisation locale tandis que les treillis, initialement définis pour des données binaires, utilisent une discrétisation globale. A partir d'une étude des propriétés des treillis dichotomiques (treillis définis après une discrétisation), nous proposons une discrétisation locale pour les treillis permettant d'améliorer ses performances en classification et de diminuer sa complexité structurelle. Puis, le processus de post-élagage mis en œuvre dans la plupart des arbres a pour objectif de diminuer la complexité de ces derniers mais aussi d'augmenter leurs performances en généralisation. Les simplifications de la structure de treillis (exponentielle en la taille de données dans les pires cas), quant à elles, sont motivées uniquement par une diminution de la complexité structurelle. En combinant ces deux simplifications, nous proposons une simplification de la structure du treillis obtenu après notre discrétisation locale et aboutissant à un modèle de classification hybride qui profite de la lisibilité des deux modèles tout en étant moins complexe que le treillis mais aussi performant que celui-ci.

Mots clés : Modèle hybride de classification, treillis de Galois, arbres de décisions, classification d'images, discrétisation de données quantitatives, simplification de treillis

Towards an hybrid model between decision trees and Galois lattice for image indexing and classification

Summary :

Image classification is generally based on two steps namely the extraction of the image signature, followed by the extracted data analysis. Image signature is generally numerical. Many classification models have been proposed in the literature, among which most suitable choice is often guided by the classification performance and the model readability. Decision trees and Galois lattices are two symbolic models known for their readability. In [Guillas 07], Guillas efficiently used Galois lattices for image classification. Strong structural links between decision trees and Galois lattices have been highlighted. Accordingly, we are interested in comparing models in order to design a hybrid model between those two. The hybrid model will combine the advantages (robustness of the lattice, low memory space of the tree and readability of both). For this purpose, we study the links between the two models to highlight their differences. Firstly, the discretization type where decision trees generally use a local discretization while Galois lattices, originally defined for binary data, use a global discretization. From the study of the properties of dichotomic lattice (specific lattice defined after discretization), we propose a local discretization for lattice that allows us to improve its classification performances and reduces its structural complexity. Then, the process of post-pruning implemented in most of the decision trees aims to reduce the complexity of the latter, but also to improve their classification performances. Lattice filtering is solely motivated by a decrease in the structural complexity of the structures (exponential in the size of data in the worst case). By combining these two processes, we propose a simplification of the lattice structure constructed after our local discretization. This simplification leads to a hybrid classification model that takes advantage of both decision trees and Galois lattice. It is as readable as the last two, while being less complex than the lattice but also efficient.

Keywords : Hybrid model of classification, Galois lattices, decision trees, images classification, quantitative data discretization, lattice filtering

L3I

Laboratoire L3i, Pôle Sciences et Technologies, Université de La Rochelle, Avenue M. Crépeau

17042 LA ROCHELLE cedex 01