Treillis de concepts et classification supervisée

Engelbert MEPHU NGUIFO — Patrick NJIWOUA

Centre de Recherche en Informatique de Lens - CNRS FRE 2499
IUT de Lens – Université d'Artois
Rue de l'université SP 16, 62307 Lens cedex
{mephu,njiwoua}@cril.univ-artois.fr

RÉSUMÉ. *La classification supervisée est une tâche de fouille de données qui consiste à construire un classifieur à partir d'un ensemble d'exemples étiquetés par leur classe (phase d'apprentissage), et ensuite à prédire la classe de nouveaux exemples avec le classifieur (phase de classement). Cet article présente plusieurs méthodes de classification supervisée basées sur la structure de treillis de concepts. Nous décrivons les principes et algorithmes d'apprentissage et de classement de chacune des méthodes, ainsi que leur complexité. Nous reportons les résultats expérimentaux publiés dans la littérature qui comparent ces méthodes avec d'autres méthodes de classification, et discutons des avantages et limitations de cette structure pour la classification supervisée.*

ABSTRACT. *Supervised classification is a two-step process. The first step (learning step) consists in building a model (or classifier) describing a predetermined set of data classes. In the second step (classification step), the model is used to predict the class label of previously unseen objects. In this paper we present different concept lattices-based supervised classification methods. We describe the learning and classification principle of each method, their algorithm and complexity. We also describe experimental comparison results obtained with other classification methods, as reported in the literature. We discuss advantages and limitations of concept lattices for supervised classification.*

MOTS-CLÉS : *Classification, Treillis de Galois, Treillis de concepts, Apprentissage, Ensemble de fermés, Votes, Plus proches voisins, Classifieur bayésien naïf*

KEYWORDS: *Classification, Galois lattice, Concept lattice, Machine learning, Closed sets, Votes, Nearest neighbour, Naive bayes classifier*

1. Introduction

La classification est une des tâches centrales de l'étape de fouille de données dans le processus d'extraction de connaissances dans les bases de données. Le problème de la classification est traité dans plusieurs communautés de recherche qui se découvrent et s'enrichissent mutuellement : statistiques, reconnaissances de formes, apprentissage automatique, réseaux de neurones et raisonnement à partir de cas. Le terme *classifi cation* en français désigne à la fois les termes anglais *classifi cation* (classification supervisée) et *clustering* (classification non supervisée). Nous nous focalisons sur la classification supervisée qui est un processus comprenant deux phases : apprentissage et classement. La phase d'apprentissage consiste à construire un modèle (ou classifieur) qui décrit un ensemble prédéterminé de classes d'exemples. La phase de classement consiste à utiliser le modèle pour affecter une classe à un nouvel exemple.

Il existe de nombreux domaines d'application de ce problème : l'attribution de crédit bancaire, la reconnaissance de gènes, la prédiction de sites archéologiques, le diagnostic médical, etc. Plusieurs méthodes de classification supervisée publiées dans la littérature s'appuient sur des techniques différentes [COR 02, SEB 02] : inférence bayésienne, plus proches voisins, arbres de décision, réseaux de neurones ou treillis de concepts. Nous nous intéressons dans cet article aux méthodes basées sur les treillis de concepts [1]. Parler de classification basée sur les treillis est une façon abusive de désigner le fait que des hypothèses résultant d'une phase d'abstraction des données sont organisées sous la forme d'un treillis. Tout système de classification utilisant cette approche recherche dans cet espace les hypothèses les plus pertinentes pour le concept à apprendre. La structure du treillis permet de restituer le concept décrit par les données dans toute sa complexité et sa diversité.

Les treillis de concepts formels (ou treillis de Galois) sont une structure mathématique permettant de représenter les classes non disjointes sous-jacentes à un ensemble d'objets (*exemples, instances, tuples ou observations*) décrits à partir d'un ensemble d'attributs (*propriétés, descripteurs ou items*). Ces classes non disjointes sont aussi appelées concepts formels, hyper-rectangles ou ensembles fermés. Une classe matérialise un concept (à savoir une idée générale que l'on a d'un objet). Ce concept peut être défini formellement par une extension (exemples du concept) ou par une intension (abstraction du concept).

Les premiers travaux formels sur les treillis de concepts se situent en algèbre universelle, à travers les travaux de Birkhoff [BIR 40, BIR 67], et plus tard de Barbut et Monjardet [BAR 70]. Les travaux de Wille [WIL 82, WIL 92], de Duquenne [GUI 86, DUQ 99], de Ganter [GAN 99] ont permis de montrer l'utilité de cette structure pour l'analyse de données. En classification supervisée, l'apprentissage se ramène dans le cas général à un problème de recherche dans l'espace des hypothèses. A ce titre l'espace de versions développé par Mitchell [MIT 82] présente des liens étroits avec

1. Le terme concept peut être employé dans les sens <unité de connaissances à apprendre> et <structure mathématique>. Dans le cas où cela peut prêter à ambiguïté, nous utiliserons *concept formel* pour désigner concept au sens "mathématique".

le treillis de concepts, tant d'un point de vue structurel que du point de vue de la complexité du modèle. Malgré la complexité exponentielle sous-jacente à la construction de la structure du treillis de concepts, plusieurs méthodes de classification ont été développées et ont produit des résultats comparables à ceux de méthodes standards, aussi bien en ce qui concerne la précision que l'efficacité en temps. En outre cette structure présente des propriétés particulières qui favorisent son usage pour la fouille de données, notamment pour la recherche de règles d'association [PAS 99] et pour l'apprentissage de concepts [KOU 98]. La problématique de mise en œuvre des algorithmes par niveaux, largement étudiée en fouille de données, et dont une formalisation algébrique est proposée par Ganascia [GAN 93], est bien présente dans les modélisations réalisées avec cette structure.

Certains travaux de recherche se sont inspirés du modèle des espaces des versions pour construire des modèles d'apprentissage à partir des treillis de concepts. CHARADE (Cubes de Hilbert Appliqués à la Représentation et à l'Apprentissage de Descriptions à partir d'Exemples) [GAN 87] est la première méthode à utiliser la correspondance de Galois, pour générer un ensemble de règles d'association non redondantes dont le but peut être restreint à une classe prédéfinie. Pour limiter l'espace de recherche, plusieurs heuristiques sont définies comme des paramètres que l'utilisateur peut sélectionner en fonction des propriétés des règles qu'il souhaite obtenir ou de la nature du problème qu'il traite. Le principe de classement à partir de cet ensemble de règles n'est pas explicitement formulé par l'auteur. L'utilisateur peut ensuite les employer pour construire sa méthode de classement. Ganascia [GAN 00] mentionne un certain nombre d'extensions de CHARADE ayant permis de traiter des applications de grande taille, notamment pour la catégorisation de textes, la découverte médicale, l'apprentissage de caractères chinois et l'extraction de connaissances dans une base de données épidémiologiques.

Plusieurs méthodes de classification basées sur le treillis de concepts ont été développées et comparées à des méthodes standards couramment utilisées pour ce type de problème. Dans cet article nous décrivons suivant un même schéma le principe des systèmes qui sont documentés dans la littérature et qui nous paraissent représentatifs de systèmes de classification à deux phases (apprentissage et classement) : GRAND [OOS 88], LEGAL [LIQ 90], GALOIS [CAR 93], RULEARNER [SAH 95], CIBLe [NJI 99] et CLNN & CLNB [XIE 02]. Nous présentons ensuite une synthèse bibliographique des résultats expérimentaux obtenus par les différents systèmes, sur la base des articles publiés dans la littérature.

Nous commençons en section 2 par une introduction formelle du treillis de concepts. La section 3 décrit le problème de la classification supervisée et les critères de comparaison couramment utilisés. Dans les sections suivantes (4 à 9), le principe de classification de chacune des méthodes énumérées ci-dessus est présenté, ainsi que les résultats expérimentaux reportés dans la littérature. Enfin nous discutons dans la section 10 les résultats de comparaison.

2. Treillis de Galois d'une relation binaire

Cette section présente la définition du treillis et une synthèse sur les algorithmes de construction. Le lecteur ayant une connaissance sur les notions de base des treillis de Galois peut passer à la section suivante.

2.1. *Définition formelle*

Définition 1 *Soit R une relation d'ordre défi nie sur un ensemble E :*

– $\forall x, y \in E$, si xRy, alors on dira que y majore x ou que x minore y.

– E est un ensemble ordonné s'il est muni d'une relation d'ordre R.

– Soit $B \subseteq E$, a est un majorant (resp. minorant) de B, s'il majore (resp. minore) tous les éléments de B.

– Soit $B \subseteq E$, le plus petit majorant (resp. plus grand minorant) de B s'il existe est appelé borne supérieure (resp. borne inférieure) de B.

Définition 2 Contexte. *Un contexte*[2] *C est un triplet (O, A, I), où O, A sont des ensembles et $I \subseteq O \times A$ est une relation (voir fi gure 1).*

O\A	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	Classe
o_1	1	1	1	1	1	1	1		+
o_2	1	1	1	1	1	1		1	+
o_3	1	1	1	1	1		1	1	+
o_4	1	1	1	1		1			+
o_5	1	1		1	1		1		-
o_6	1	1	1		1			1	-
o_7	1		1			1			-

Figure 1. *La matrice binaire décrivant la relation I du contexte $C = (O, A, I)$*

La figure 1 montre un exemple de contexte représenté par $C = (O, A, I)$ avec $O = \{o_1, o_2, o_3, o_4, o_5, o_6, o_7\}$ et $A = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8\}$. Ce contexte nous servira d'exemple de référence tout au long de cet article. Nous avons ajouté la colonne *Classe* qui servira à illustrer le problème de classification supervisée.

Définition 3 Contexte binaire. *Un contexte (O, A, I) est dit binaire si les éléments de A ne peuvent prendre que deux valeurs (*0 *ou* 1*) qui indiquent respectivement l'absence ou la présence de l'attribut concerné dans la description d'un objet.*

2. Il s'agit d'un contexte formel au sens de R. Wille [WIL 82].

Définition 4 Correspondance de Galois entre ensembles ordonnés. *Soient E_1 et E_2 deux ensembles ordonnés. Soit p (respectivement q) une application monotone décroissante définie de E_1 dans E_2 (respectivement de E_2 dans E_1) telle que soit $x \in E_1$ et $y \in E_2$ $y \leq p(x)$ si et seulement si $x \leq q(y)$. Le couple d'applications (p, q) est appelé correspondance de Galois entre E_1 et E_2.*

Définition 5 Applications f et g d'un contexte binaire. *Soient f et g deux applications définies comme suit :*

- *A tout élément o de O, on associe $f(o) = \{a \in A; (o, a) \in I\}$;*
- *A tout élément a de A, on associe $g(a) = \{o \in O; (o, a) \in I\}$.*

Proposition 1 *f et g sont des applications monotones décroissantes.*

Remarque 1 *f et g sont étendues respectivement aux parties de O et de A comme suit : soient $O_1 \subseteq O$ et $A_1 \subseteq A$,*

$$f(O_1) = \bigcap_{o \in O_1} f(o) \qquad g(A_1) = \bigcap_{a \in A_1} g(a)$$

Proposition 2 Fermeture. *$h = g \circ f$ et $h' = f \circ g$ sont des applications monotones croissantes, extensives et idempotentes. h et h' sont appelés opérateurs de fermeture sur les ensembles de parties $\mathcal{P}(O)$ et $\mathcal{P}(A)$.*

Proposition 3 *Le couple (f, g) est une correspondance de Galois entre les ensembles ordonnés $\mathcal{P}(O)$ et $\mathcal{P}(A)$ du contexte binaire (O, A, I).*

Définition 6 Concept : Intension/Extension. *Soient $O_1 \subseteq O$ et $A_1 \subseteq A$. La paire $\{A_1, O_1\}$ est un concept formel si $f(O_1) = A_1$ et $g(A_1) = O_1$. O_1 (resp. A_1) est appelé l'extension (resp. l'intension) du concept $\{A_1, O_1\}$.*

Exemple 1 *D'après le contexte de la figure 1, $\{\{a_1, a_2, a_3, a_4\}, \{o_1, o_2, o_3, o_4\}\}$ est un concept mais $\{\{a_6\}, \{o_1, o_2\}\}$ n'en est pas un, puisqu'on a :*
$f(\{o_1, o_2\}) = \{a_1, a_2, a_3, a_4, a_5, a_6\} \neq \{a_6\}$.

Remarque 2 *Si $\{A_1, O_1\}$ est un concept alors A_1 est un fermé de h' ($h'(A_1) = f \circ g(A_1) = f(O_1) = A_1$) et O_1 est un fermé de h ($h(O_1) = g \circ f(O_1) = g(A_1) = O_1$).*

Définition 7 L'ordre sur les concepts. Sous-concept et sur-concept *La relation de sous-concept/sur-concept (dénotée $\leq$[3]) est la relation d'ordre définie entre les concepts de la façon suivante : pour deux concepts $\{A_1, O_1\}$ et $\{A_2, O_2\}$, $\{A_1, O_1\} \leq \{A_2, O_2\} \iff O_1 \subseteq O_2$ ($\iff A_2 \subseteq A_1$).*

$\{A_1, O_1\}$ (respectivement $\{A_2, O_2\}$) est un sous-concept (resp. sur-concept) de $\{A_2, O_2\}$ (resp. $\{A_1, O_1\}$).

3. lire "moins général que".

Proposition 4 *L'ensemble des fermés de h (resp. h') ordonné par l'inclusion, avec les opérations d'union et d'intersection ensemblistes, a une structure de treillis complet. Ces deux treillis, basés respectivement sur O et sur A, sont isomorphes.*

Définition 8 Treillis de Galois. *L'ensemble L de tous les concepts, muni de la relation d'ordre $\leq$, est un treillis complet, appelé treillis de Galois $L(C)$ de C.*

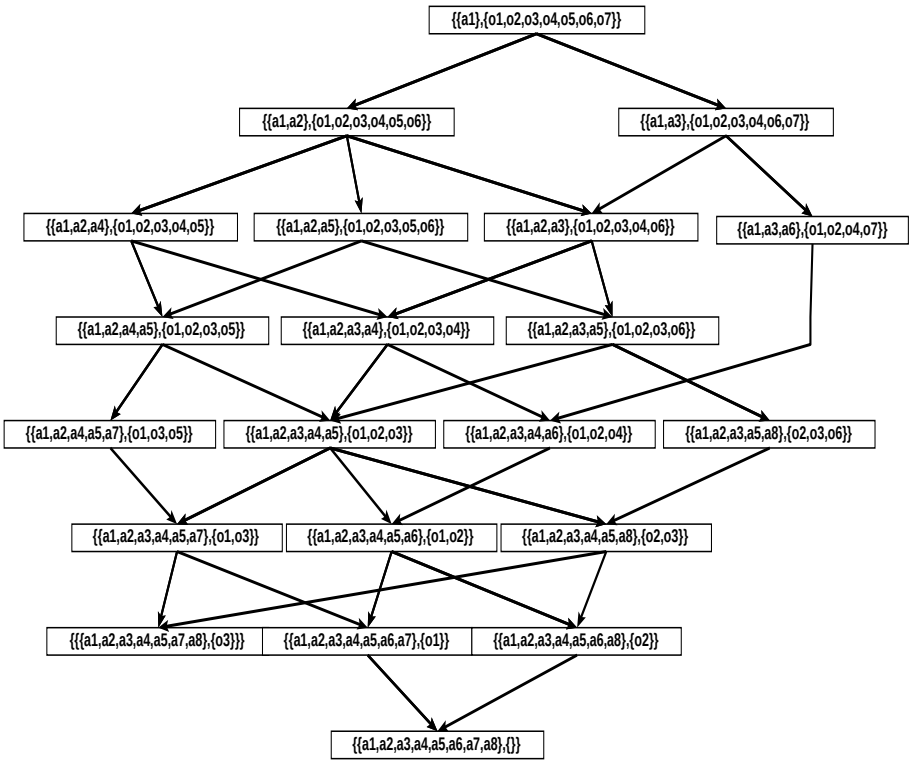


Figure 2. *Le treillis de Galois du contexte précédent (voir fi gure 1)*

La figure 2 montre le diagramme de Hasse du treillis de Galois du contexte précédent. Un rectangle représente un concept et les arcs entre les rectangles matérialisent la relation d'ordre du plus général (en haut) vers le plus spécifique (en bas). Plusieurs logiciels de visualisation du graphe de Hasse du treillis ont été développés, parmi lesquels TOSCANA et ANACONDA par l'équipe de Wille [VOG 95], GLAD par Duquenne [DUQ 96] et CERNATO par la société Navicon GmbH [BEC 01].

2.2. *Algorithmes de construction du treillis de Galois*

Plusieurs algorithmes de construction du treillis de Galois ont été proposés et ont fait l'objet d'études comparatives [KOU 98, NJI 00, KUZ 01] ou de synthèse [GUé 91]. Etendant l'étude faite par Guénoche [GUé 91], Njiwoua [NJI 00] a présenté six algorithmes de construction du treillis suivant une typologie relative à l'approche (incrémentale ou non) choisie par leurs auteurs et en se limitant au cas du contexte décrit par une matrice binaire. Ces algorithmes sont caractérisés suivant la méthode de construction choisie (ascendante ou descendante).

Parmi les algorithmes analysés dont certains sont plus ou moins détaillés, on trouve ceux de Chein [CHE 69], de Ganter [GAN 84], de Bordat [BOR 86], de Godin et al. [GOD 91] (assez proche de l'algorithme de Norris [NOR 78]), d'Oosthuizen [OOS 91], de Carpineto et Romano [CAR 93]. Ils sont comparés dans [NJI 00] suivant différents critères parmi lesquels : la complexité théorique en temps d'exécution dans le pire des cas, la construction ou non du diagramme de Hasse, la technique de mise à jour utilisée (incrémentale ou non) et le type de parcours effectué (descendant ou ascendant).

Le treillis de Galois construit dépend non seulement de la taille des éléments (nombre d'exemples et d'attributs), mais aussi pour une taille fixée, de la nature de la relation entre exemples et attributs. Par conséquent, la seule comparaison théorique en terme de complexité des différents algorithmes est faite dans le pire des cas. Cependant les bornes de complexité présentées sont généralement de loin supérieures aux résultats observés en pratique par les différents auteurs, ainsi que dans une étude récente faite par Kuznetsov et Obiedkov [KUZ 01].

Dans leur article, Kuznetsov et Obiedkov [KUZ 01] analysent dix algorithmes de construction du treillis de Galois. Ils présentent une étude de leurs complexités théoriques et une comparaison expérimentale sur des jeux de données artificiels. Dans cette analyse, on constate que l'algorithme de Nourine et Raynaud [NOU 99] qui a la meilleure complexité théorique actuelle dans le pire des cas, n'est pas le plus rapide dans les expérimentations. Les auteurs font des recommandations en fonction de la nature du contexte.

Afin de mesurer l'efficacité de ces algorithmes sur les données réelles, Fu et Mephu [FU 03] ont comparé une implémentation de quatre algorithmes sur les données couramment utilisées pour tester les méthodes d'apprentissage [BLA 98]. Il en résulte que l'algorithme de Ganter (NextClosure) est en général le plus efficace sur ces données, ainsi que dans le pire des cas, lorsqu'il s'agit de générer simplement les concepts du treillis. Fu et Mephu discutent aussi de l'efficacité des algorithmes en fonction du nombre d'attributs par rapport au nombre d'objets.

3. Classification supervisée – éléments de comparaison

La classification supervisée est un processus composé d'une phase d'apprentissage suivie d'une phase de classement. Avant de définir les deux phases, nous allons rappeler quelques définitions.

3.1. *Définitions : Cohérence, Complétude, Pertinence d'une hypothèse*

Définition 9 *Soit O_1 un ensemble d'exemples.*

– O_1 est dit **cohérent** *si tous ses éléments appartiennent à une même classe.*

– O_1 est dit **complet** *s'il contient tous les éléments d'une classe donnée.*

– O_1 est dit **significatif** *s'il contient "suffi samment" d'exemples (en fonction d'un seuil donné par l'utilisateur).*

– O_1 est dit **pertinent** *s'il est à la fois signifi catif et cohérent.*

Définition 10 *Une hypothèse (ou un ensemble de propriétés A_1) couvre un ensemble d'exemples O_1, si chaque élément de O_1 vérifi e toutes les propriétés appartenant à A_1. On dit encore que O_1 est couvert par A_1.*

Définition 11 *Une hypothèse A_1 rejette O_1 s'il existe un élément de O_1 qui ne vérifi e pas l'une des propriétés appartenant à A_1.*

Définition 12 *Une hypothèse est dite cohérente (resp. complète ou signifi cative ou pertinente) si l'ensemble des exemples qu'elle couvre est cohérent (resp. complet ou signifi catif ou pertinent).*

Définition 13 *Soient A_1 et A_2 deux ensembles de propriétés. A_2 est dit plus général que A_1 (on note $A_1 \leq A_2$) si l'ensemble des exemples couverts par A_1 est inclus dans l'ensemble des exemples couverts par A_2.*

3.2. *Phase d'apprentissage*

Le problème de l'apprentissage supervisé est exprimé par la donnée d'un *ensemble d'apprentissage* qui contient des exemples o_i (encore appelés tuples, objets, instances ou observations) décrits par un ensemble fini de propriétés a_j (encore appelés attributs ou descripteurs), et appartenant à une ou plusieurs classes y_i. L'ensemble d'apprentissage caractérise une certaine fonction f à déterminer.

L'ensemble d'apprentissage de taille $n \in \mathbb{N}$ est une suite de couples (o_1, y_1), ..., (o_n, y_n), avec $y_i = f(o_i)$ pour $1 \leq i \leq n$ (voir colonne $classe$ de la figure 1). Chaque exemple o_i est représenté par un vecteur $(o_{i_1}, \ldots, o_{i_m})$ où o_{i_j} est la valeur de o_i pour la propriété a_j de l'ensemble des attributs A. Cette valeur peut être *numérique*

(c'est-à-dire soit discrète et non bornée, soit continue) ou *symbolique* (c'est-à-dire soit booléenne, soit nominale). Les y_i indiquent la classe des exemples o_i, correspondent aux valeurs d'une propriété particulière a_j, et prennent un nombre fini de valeurs symboliques (d'où un nombre fini de classes).

Dans l'exemple de la figure 1, les valeurs de y_i sont '$+$' pour les exemples positifs (ou exemple du concept à apprendre) et '$-$' pour les exemples négatifs (ou contre-exemple du concept à apprendre). Les objets $\{o_1, \ldots, o_4\}$ sont les exemples de la classe $+$, et les objets $\{o_5, \ldots, o_7\}$ sont des exemples de la classe $-$.

La phase d'apprentissage consiste à construire un modèle (ou classifieur) $\hat{f}$, qui approxime au mieux la fonction f à partir d'un ensemble d'exemples sélectionnés de manière aléatoire dans l'ensemble d'apprentissage. Dans le cas où l'on a plusieurs classes, on parle d'apprentissage multi-classes. Si le système de classification ne peut traiter qu'une seule classe, le mode opératoire dans le cas de plusieurs classes, consistera à faire autant d'apprentissages qu'il y a de classes, où les instances positives sont celles de la classe apprise et les instances négatives sont les exemples des autres classes.

3.3. *Phase de classement*

Dans la phase de classement le modèle appris $\hat{f}$ est utilisé pour affecter une classe à chaque nouvel exemple. Etant donné o_k, il s'agit de déterminer $\hat{y}_k = \hat{f}(o_k)$. Pour valider le modèle appris, un ensemble test contenant des exemples dont on connaît la classe y_i est utilisé. Le modèle commet une erreur lorsque $\hat{y}_k$ est différent de $y_k = f(o_k)$. Le modèle appris $\hat{f}$ approxime au mieux la fonction f lorsque le nombre d'erreurs calculé tend vers zéro. Ce qui revient à déterminer le taux d'erreur du modèle (taux d'erreur = 1 – taux de précision) qui exprime le pourcentage du nombre d'exemples mal classés par rapport au nombre total d'exemples.

Si le modèle est validé sur l'ensemble test alors il est utilisé pour classer les exemples dont on ne connaît pas la classe.

Remarque 3 *Le treillis de concepts peut être utilisé pour faire de la classifi cation non supervisée (caractériser des exemples sans aucune information sur les classes représentées) ou de la classifi cation supervisée.*

3.4. *Critères de comparaison de méthodes de classification*

Nous allons caractériser les méthodes de classification basées sur les treillis de concepts en fonction des critères suivants : principe de génération des hypothèses apprises, validation du modèle, complexité théorique. D'autres critères comme l'utilisation des connaissances du domaine peuvent aussi être utilisés. Dans les systèmes que nous allons étudier, aucun n'utilise des connaissances du domaine.

Le principe de génération détermine l'espace de recherche considéré pour engendrer les hypothèses. La manière par laquelle les hypothèses apprises sont obtenues et évaluées est décrite. Le critère d'arrêt permettant de limiter l'exploration de l'espace de recherche est présenté. Dans les cas qui nous intéressent, l'apprentissage est dirigé par les données avec deux approches (duales) possibles : l'approche descendante ou par spécialisations successives qui consiste à détecter à chaque étape les sous-ensembles spécifiques maximaux d'un ensemble donné d'objets ; et l'approche ascendante ou par généralisations successives qui consiste à construire à chaque étape les propriétés communes les plus spécifiques à un sous-ensemble d'objets.

Le critère de validation est basé sur le calcul du taux de précision qui mesure la capacité du modèle appris à prédire correctement la classe des exemples de l'ensemble test. Ce taux est égal au pourcentage des exemples de l'ensemble test qui sont correctement classés par le modèle. La plupart des techniques de calcul du taux de précision viennent du domaine des statistiques et sont caractérisées par la façon selon laquelle l'ensemble initial (apprentissage et test) est généré :

• la validation croisée (d'ordre k). Les exemples sont partitionnés en k sous-ensembles de taille identique. Chaque sous-ensemble sert une fois comme ensemble test, l'apprentissage s'effectuant sur les $k-1$ autres sous-ensembles.

• la resubstitution. Le même ensemble est utilisé pour l'apprentissage et le test de la pertinence des hypothèses apprises.

• le "leave-one-out". Chaque exemple de l'ensemble est utilisé une fois et à tour de rôle pour le test de la pertinence des hypothèses obtenues à partir des autres exemples de l'ensemble.

• le "holdout". Il consiste à partitionner aléatoirement l'ensemble initial en deux sous-ensembles distincts ; l'un servira pour l'apprentissage et l'autre pour le test.

Une comparaison de plusieurs techniques d'estimation du taux de précision est faite par Dietterich [DIE 97]. Il en ressort qu'aucune technique n'est meilleure que les autres dans toutes les situations et par conséquent, il faut utiliser le même critère de validation pour comparer la précision des méthodes de classification.

Le troisième critère de comparaison consiste à calculer un ordre de complexité en temps et en espace, en fonction de la taille des données et des paramètres de classification. Le critère de complexité permet de juger de l'efficacité des systèmes de classification. Lorsque les performances de précision et de compréhensibilité entre systèmes de classification sont égales, le critère de complexité en temps et en espace peut permettre de préférer le système ayant la plus petite complexité.

D'autres critères, tels que le langage de description ou le langage de représentation des hypothèses apprises, peuvent aussi être utilisés. Ils permettent de juger la compréhension par l'utilisateur des connaissances apprises et des décisions produites par le système. Dans notre cas, le langage de description utilisé est en général limité à la logique propositionnelle (présentée sous la forme attribut/valeur).

Nous allons dans les sections suivantes présenter chronologiquement par rapport aux dates de publication, les méthodes de classification basées sur les treillis de concepts, suivant leur principe, la manière de construire le treillis, la phase d'apprentissage, la phase de classement, la complexité et les comparaisons expérimentales effectuées.

4. GRAND

GRAND (GRAph-based iNDuction) est le système développé par Oosthuizen [OOS 88], qui s'adapte à l'apprentissage supervisé et non supervisé.

4.1. *Principe et Construction du pseudo-treillis*

Il construit un pseudo-treillis d'exemples-attributs qui contient l'ensemble des généralisations les plus spécifiques de l'espace des versions. Tous les concepts du treillis complet sont présents dans le pseudo-treillis à l'exception du supremum ou de l'infimum si chacun d'eux correspond à l'ensemble vide. Certains nœuds sont rajoutés tels que les nœuds-attributs et les nœuds-exemples. Ceux-ci ne correspondent pas toujours à des concepts formels. Dans le cas d'attribut universel (attribut commun à tous les exemples) le supremum est bien présent dans le pseudo-treillis. De même que dans le cas où au moins un exemple possède l'ensemble des attributs (exemple universel), l'infinimum est representé. La classe des exemples est un attribut comme les autres (voir la figure 3 pour une illustration sur l'exemple courant). Oosthuizen [OOS 91] représente un nœud uniquement par son intension. Il considère alors trois types de nœuds :

1) Les *nœuds-attributs*, il s'agit des éléments de A. Ce sont les éléments maximaux du treillis. Le nœud correspondant est le plus général contenant l'attribut considéré.

2) Les *nœuds-exemples*, il s'agit des éléments de O, représentés par leurs propriétés. Ce sont les éléments minimaux du treillis. Le nœud correspondant est le plus spécifique contenant l'exemple considéré.

3) Les *nœuds-intermédiaires* ou concepts. Un nœud intermédiaire hérite des attributs de tous ses majorants (ascendants) et des exemples de tous ses minorants (descendants).

Il est possible grâce à un paramètre, de prendre en compte des exemples qui vérifient partiellement les hypothèses construites. Il en découle une plus grande souplesse dans la description des concepts. Il est aussi possible de préciser le nombre minimum d'exemples que doit couvrir un nœud pour être utilisé dans la phase de classement.

Chaque nouvel exemple est comparé explicitement à l'ensemble des exemples déjà insérés dans le pseudo-treillis. Pour chaque nouvel exemple o_{k+1} considéré, des connexions sont établies ou mises à jour. Pour des raisons de simplicité, nous ne matérialisons pas ces connexions dans l'algorithme de la figure 4 (algorithme de mise à jour) qui procède de la façon suivante :

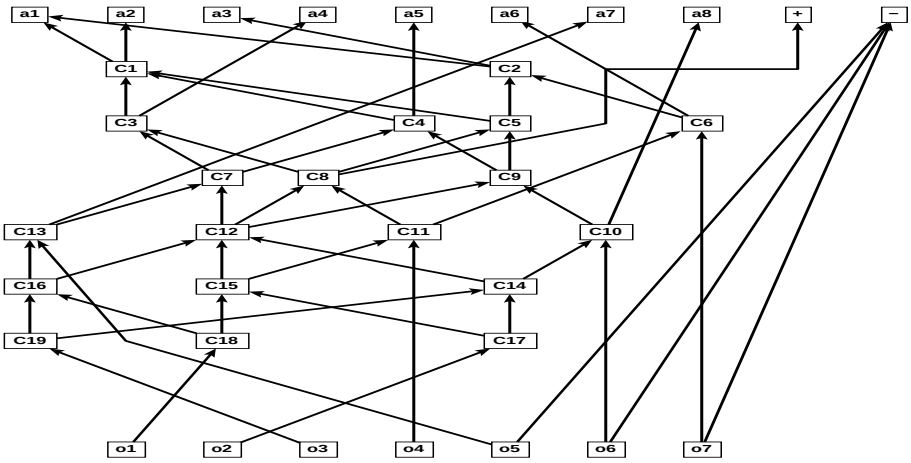


Figure 3. *Le pseudo-treillis obtenu par GRAND sur le contexte initial de la fi gure 1*

1) Un nœud-exemple correspondant à o_{k+1} est créé (*ligne* 1). Ce nœud est relié à l'ensemble des attributs de o_{k+1} (*ligne* 2). Prenons l'exemple de la figure 5 avec l'insertion de l'exemple o_3. o_3 est relié aux nœuds-attributs $a_1, a_2, a_3, a_4, a_5, a_7, a_8, +$;

2) Pour chaque nœud déjà présent dans L et qui a au moins un attribut en commun avec o_{k+1}, insérer un nœud correspondant aux attributs communs, et créer les liens entre nœuds (*lignes* 5 *à* 11, *on procède par ordre décroissant sur la taille de l'ensemble des attributs communs entre* o_{k+1} *et les nœuds déjà présents*). Sur l'exemple, o_3 a 6 attributs communs avec $C1$, ce qui va engendrer la création du nœud $C2$. On crée un lien entre o_3 et $C2$, et entre $C1$ et $C2$. Partant de a_7 (resp. a_8) on trouve un autre attribut commun entre o_1 (resp. o_2) et o_3 qui permet de créer le nœud $C3$ (resp. $C4$).

3) Eliminer les connexions redondantes (*ligne* 12). Sur l'exemple, on supprime les liens entre o_3 et les nœuds attributs $a_1, a_2, a_3, a_4, a_5, a_7, a_8, +$. On supprime les liens entre $C1$ et $a_1, a_2, a_3, a_4, a_5, +$, entre o_1 et a_7, entre o_2 et a_8.

La figure 5 illustre la construction du pseudo-treillis avec l'algorithme de mise à jour sur le contexte initial de la figure 1. A l'étape 2, $C1$ correspond aux attributs en commun entre o_1 et o_2.

4.2. *Apprentissage et Classement*

Le système induit les règles les plus générales. Pour ce faire, GRAND recherche pour chaque nœud-attribut représentant une classe x du concept, le nœud-intermédiaire $\{A_i, O_i\}$ le plus général à partir duquel on peut inférer la classe x par déduction ; c'est-à-dire $x \in A_i$ et $x \in (f \circ g)(A_i - \{x\})$ et A_i minimal. La règle obtenue a pour antécédent $A_i - \{x\}$ et pour conséquent la classe x.

Mise-à-jour(o_{k+1}, L^k)
DEBUT
1. $L^{k+1} \longleftarrow L^k \bigcup \{o_{k+1}\}$
2. Relier o_{k+1} aux nœuds-attributs qu'il possède
3. $A_{k+1} \longleftarrow \{a_i$ communs à o_{k+1} et aux éléments de $L^k\}$
4. $C_{k+1} \longleftarrow L^k$ moins les noeuds-attributs
5. TANTQUE $A_{k+1} \neq \{\}$ FAIRE
6. Choisir O_i dans C_{k+1} tel que
   $|f_{A_{k+1}}(o_{k+1}) \bigcap f_{A_{k+1}}(O_i)|$ soit le plus élevé
7. créer dans L^{k+1} le noeud c dont l'intension est
   $\{f_{A_{k+1}}(o_{k+1}) \bigcap f_{A_{k+1}}(O_i)\}$
8. $L^{k+1} \longleftarrow L^{k+1} \bigcup c$
9. Relier le noeud c aux autres noeuds de L^{k+1}
10. Retirer $f_{A_{k+1}}(O_i)$ dans A_{k+1}
11. FTANTQUE
12. Eliminer les connexions redondantes
13. Retourner(L^{k+1})
FIN

Figure 4. *Algorithme de Oosthuizen*

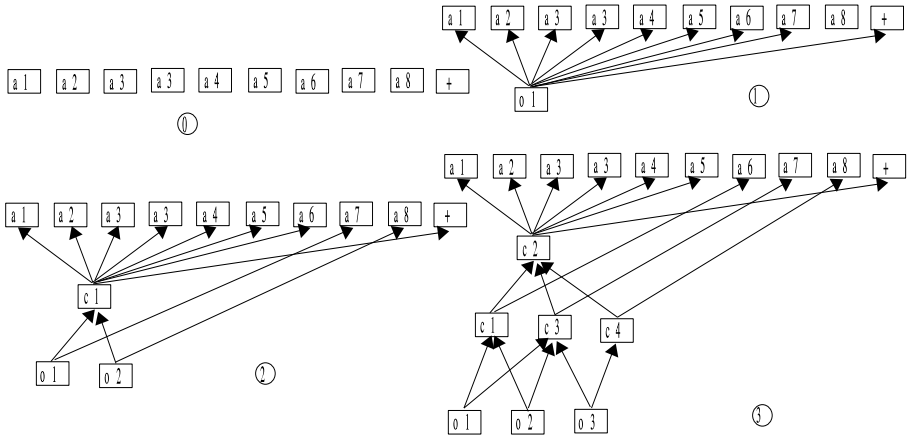


Figure 5. *Etapes de construction du pseudo-treillis avec l'algorithme de Oosthuizen. Ici nous n'avons fait fi gurer que la classe + (classe de ces 3 exemples).*

Exemple 2 *Pour l'exemple courant (fi gure 3), partant du nœud-attribut qui matérialise la classe +, on arrive au concept $C8$ qui est un des fi ls (dans ce cas c'est le seul). On recherche les nœuds-attributs dérivant de $C8$, et on obtient a_1, a_2, a_3, a_4 et +.*

Puisque $C8$ est la borne inférieure des nœuds-attributs a_1, a_2, a_3 et a_4, la règle **IF** $a_1 \wedge a_2 \wedge a_3 \wedge a_4$ **THEN** $+$ *sera générée. Ici les exemples négatifs ne forment pas un concept.*

Pour chaque exemple à classer, GRAND choisit d'appliquer les règles les plus spécifiques pour cet exemple. Un vote majoritaire peut être effectué sur l'ensemble des règles vérifiées par l'objet à classer, mais l'auteur de GRAND conseille (de façon empirique sur la base des expérimentations effectuées) de restreindre ce vote aux trois meilleures règles (les plus spécifiques pour l'objet).

4.3. *Complexité et Comparaison expérimentale*

GRAND construit entièrement le pseudo-treillis avant de mettre en œuvre son principe de classification. L'ordre de complexité en temps est en $O(2^k \cdot k^4)$ où k = min (n,m) [OOS 88].

La complexité en temps de l'opération de mise à jour du treillis est liée au nombre de concepts déjà générés et majorant l'objet concerné. L'ordre de cette complexité est en $O(m^3)$ dans le meilleur des cas [OOS 88]. Dans les cas pratiques étudiés, la complexité en espace du treillis est en $O(nm^3)$ [OOS 88].

GRAND implémenté en Prolog puis en C, a été comparé [OOS 94] sur 2 jeux de données de l'UC-irvine [BLA 98] (Breast Cancer et Lymphography) à plusieurs méthodes de classification [MIC 86] : Assistant, AQ15, AQR, Bayes et CN2. L'auteur de GRAND a reproduit les procédures expérimentales des autres systèmes et reporté les résultats publiés. La méthode de validation utilisée consiste à faire une sorte de "holdout" amélioré. 70% d'exemples sont sélectionnés pour l'ensemble d'apprentissage et 30% pour l'ensemble test. L'auteur répète cette opération quatre fois et le résultat de GRAND est la moyenne des quatre taux de précision obtenus.

GRAND obtient des taux de précision comparables aux autres méthodes. La comparaison est donnée à titre indicatif pour montrer la faisabilité de l'usage des treillis de concepts pour la classification. Les temps d'exécution de ces expériences ne sont pas mentionnés. L'auteur indique seulement un temps approximatif de 15 mn permettant de construire un treillis d'environ 1500 nœuds avec 200 exemples et 10 attributs sur une machine DEC 5000/240, puis un temps mis pour maintenir le treillis avec un nouvel exemple de moins d'une demi-seconde, d'une minute ou d'environ 20mn lorsqu'il contient respectivement 100, 6000 et 18000 nœuds.

5. LEGAL

LEGAL (LEarning with GAlois Lattice) [LIQ 90] s'appuie sur ses paramètres d'apprentissage pour construire seulement un sup-demi treillis, ce qui permet de réduire l'espace de recherche des hypothèses. Parmi les variantes de ce système [MEP 94],

nous présentons celle qui construit le treillis des concepts uniquement sur les exemples positifs (LEGAL-E).

5.1. *Principe et Construction du treillis*

Le processus d'apprentissage de LEGAL consiste à construire un ensemble ordonné d'hypothèses. LEGAL utilise une méthode de généralisation descendante basée sur le treillis de Galois. Le concept à apprendre est représenté par le contexte (O, A, I) où O regroupe l'ensemble des exemples positifs (O^+) et l'ensemble des exemples négatifs ou contre-exemples (O^-). Le système utilise deux paramètres α et β qui sont respectivement les critères de *validité* et de *quasi-cohérence*. Les notions de validité (ou complétude) et de quasi-cohérence (ou correction) pour les hypothèses et les nœuds sont introduites comme suit :

• Une hypothèse est dite *valide* (ou suffisamment complète) si elle est vérifiée par "assez" d'exemples positifs. Soit V l'ensemble des hypothèses valides : $V = \{Y \subseteq A$ t.q. $\exists\{A_1, O_1\} \in L,\ Y = A_1$ et $\frac{|O_1 \bigcap O^+|}{|O^+|} \geq \alpha\}$. Ce critère correspond à la notion de support utilisé pour générer les motifs fréquents pour les règles d'association [PAS 99].

• Une hypothèse est dite *quasi-cohérente* (ou suffisamment correcte) si elle est vérifiée par "peu" de contre-exemples. Soit C l'ensemble des hypothèses quasi-cohérentes : $C = \{Y \subseteq A$ t.q. $\exists\{A_1, O_1\} \in L,\ Y = A_1$ et $\frac{|O_1 \bigcap O^-|}{|O^-|} \leq \beta\}$.

• Une hypothèse est dite *pertinente* si elle est à la fois valide et quasi-cohérente. Soit P l'ensemble des hypothèses pertinentes : $P = C \bigcap V$.

• Un nœud est dit valide (resp. quasi-cohérent ou pertinent) si son intension constitue une hypothèse valide (resp. quasi-cohérente ou pertinente).

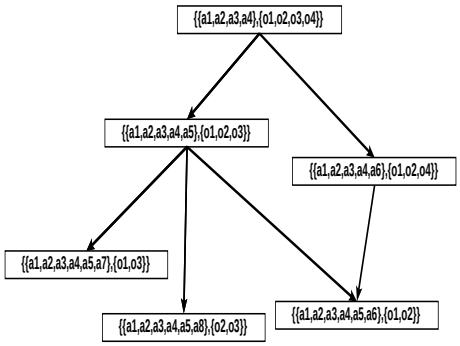
Proposition 5 *Si* $\{A_1, O_1\}$ *est quasi-cohérent alors tous ses minorants le sont aussi. Si* $\{A_1, O_1\}$ *est valide alors tous ses majorants le sont aussi.*

LEGAL [MEP 94] utilise une version modifiée de l'algorithme de Bordat [BOR 86] dans laquelle, à chaque étape, seuls les nœuds valides sont conservés dans le treillis et font l'objet d'une spécialisation à l'étape suivante. Le programme s'arrête lorsqu'aucun des nœuds générés à l'étape précédente n'est valide. La construction du treillis se fait par niveaux.

5.2. *Apprendre et Classer avec LEGAL*

LEGAL construit uniquement le sup-demi treillis des nœuds valides. Cette restriction se justifie par le fait que les nœuds situés plus bas dans le treillis sont trop spécifiques. Ils sont donc moins adaptés à la généralisation et peuvent conduire le système à faire du sur-apprentissage. Dans sa version actuelle, LEGAL se limite à un

apprentissage d'une classe constituée d'instances positives et éventuellement d'instances négatives. La connaissance obtenue est matérialisée par l'ensemble des hypothèses pertinentes.



O	Nb. pert. vérifi ées	Classe
1	5	+
2	5	+
3	4	+
4	2	+
5	0	-
6	0	-
7	0	-

Figure 6. *Le sup-demi treillis et le nombre d'hypothèses pertinentes vérifi ées par chaque objet*

La figure 6 montre le treillis construit par LEGAL sur les exemples positifs et le nombre d'hypothèses pertinentes vérifiées par chaque objet. Pour $\alpha = 0.5$ et $\beta = 0.33$, on a comme ensemble d'hypothèses pertinentes :

$P = \{\{a_1, a_2, a_3, a_4\}, \{a_1, a_2, a_3, a_4, a_5\}, \{a_1, a_2, a_3, a_4, a_6\}, \{a_1, a_2, a_3, a_4, a_5, a_7\}, \{a_1, a_2, a_3, a_4, a_5, a_8\}, \{a_1, a_2, a_3, a_4, a_5, a_6\}\}$.

LEGAL effectue un *vote majoritaire* parmi l'ensemble P des hypothèses pertinentes vérifiées par l'exemple à classer. Soit o_i un exemple, P_i est l'ensemble des hypothèses pertinentes que vérifie o_i. $P_i = \{r \in P \mid o_i \text{ vérifie } r\}$. Soient λ et γ, respectivement les seuils de justification et de réfutation proposés par LEGAL.

• o_i est un exemple positif si $|P_i|/|P| \geq \lambda$ c'est-à-dire si o_i vérifie *suffi samment* d'hypothèses pertinentes : o_i est *justifi é*.

• o_i est un exemple négatif si $|P_i|/|P| < \gamma$ c'est-à-dire si o_i vérifie *peu* d'hypothèses pertinentes : o_i est *réfuté*.

• Sinon ($\gamma \leq |P_i|/|P| < \lambda$), o_i est ambigu et le système est silencieux. Pour éviter que le système soit silencieux, il suffit de donner une valeur identique à γ et λ.

Si on se réfère à la figure 6, les exemples d'apprentissage peuvent être partitionnés en fonction du nombre d'hypothèses pertinentes vérifiées. Pour des valeurs de λ et γ

respectivement égales à $\frac{1}{3}$ et $\frac{1}{6}$, on obtient deux classes qui coïncident avec les classes connues des exemples.

5.3. *Complexité et Comparaison expérimentale*

Une analyse de la complexité du système LEGAL qui intègre les paramètres du système a été proposée [NJI 97a].

Taille du sup-demi treillis : chaque nœud du treillis pour être valide doit contenir au moins $\alpha \cdot n$ exemples positifs. Par conséquent, un majorant pour la taille du sup-demi treillis L est obtenue : $|L| \leq 2^k - 2^{\inf(k,\alpha \cdot n)-1}$, avec $k = \min(n, m)$.

Complexité en temps : une étape de la construction du treillis consiste à générer tous les sous-nœuds d'un nœud donné et à insérer les nœuds valides dans L, ce qui donne : $O\left((|L| \cdot n \cdot (1-\alpha)) \cdot (|I| + min\left(n \cdot (1-\alpha), m\right)) + n.m\right)$.

Dans l'article original [LIQ 90], aucune comparaison expérimentale n'est faite. Dans des publications ultérieures [MEP 94, MEP 93], LEGAL a été appliqué à plusieurs jeux de données de l'UCI [BLA 98]. Les comparaisons avec d'autres systèmes de classification sont présentées à titre indicatif, car les conditions d'expérimentation ne sont pas identiques, les auteurs se limitant à reporter des résultats publiés dans la littérature pour les autres systèmes. LEGAL a été appliqué sur plusieurs problèmes réels de classement venant de domaines différents comme la biologie [VIG 96] ou l'archéologie [MEP 98].

6. GALOIS

Le système GALOIS, développé par Carpineto et Romano [CAR 93], fait du regroupement conceptuel à l'aide du treillis. Il s'applique autant à la classification non supervisée (par exemple l'interrogation des bases documentaires [CAR 96]), qu'à la classification supervisée. Dans ce dernier cas, l'apprentissage s'effectue de façon non supervisée et le treillis est ensuite utilisé pour classer les nouveaux exemples.

6.1. *Principe et Construction du treillis*

Carpineto et Romano [CAR 93] caractérisent les nœuds du treillis uniquement par leur intension. Par conséquent dans ce qui suit, on parlera du concept A_i pour désigner $\{A_i, O_i\}$. Pour la procédure de mise à jour, ils font la même remarque que Godin [GOD 95], selon laquelle les concepts déjà présents dans le treillis peuvent être modifiés mais ne sont jamais supprimés.

Ils proposent alors, en plus de maintenir le diagramme de Hasse du treillis, un principe simple à respecter à chaque étape pour garantir le résultat voulu. Ce principe est un résultat du théorème fondamental de l'analyse formelle de concepts [WIL 82] :

Proposition 6 *Si* $\{A_i, O_i\}$ *est la borne supérieure des deux concepts* $\{A_j, O_j\}$ *et*$\{A_l, O_l\}$, *alors* $A_i = A_j \bigcap A_l$.

Pour les concepts modifiés, seule l'extension est concernée (ajout du nouvel objet dans son extension). Cette opération n'a aucune conséquence pour GALOIS, puisque l'extension n'est pas représentée. Par contre, un nouveau concept est créé en se basant sur un concept existant dit concept générateur. L'algorithme de mise à jour du treillis de concepts se base sur la caractérisation du concept générateur.

Proposition 7 *Soit* o_{k+1} *le nouvel objet à insérer, le concept* $A_i \in L^k$ *est dit générateur du concept* $Z \in L^{k+1}$ *si et seulement si* A_i *n'est pas inclus dans* $f(o_{k+1})$ *et* $\forall A_j \in L^k$ *sur-concept de* A_i, $A_i \bigcap f(o_{k+1})$ *n'est pas inclus dans* A_j.

La construction du treillis se fait de façon incrémentale en utilisant l'approche ascendante (voir figure 7). A l'étape k, soient o_{k+1} le nouvel objet à insérer dans le treillis L^k, A_i un concept, A_j un sur-concept de A_i et $Z = f(o_{k+1}) \bigcap A_i$.Lorsque Z est non vide, on considère les quatre cas exclusifs suivants :

1) $\exists A_j \supset Z$; par conséquent A_i n'est pas un concept générateur (*lignes* 9 *à* 10).

2) $\exists A_j = Z$; le nouveau concept Z appartient déjà au treillis (*lignes* 9 *à* 10).

3) $\exists A_j \subset Z$; le nouveau concept Z est inséré dans le treillis comme borne supérieure de A_i et de $f(o_{k+1})$ (*lignes* 11 *à* 12, *à la sortie de la boucle POUR Inserer=Vrai*).

4) $\forall A_j$, A_j est incomparable à Z ; le nouveau concept Z est inséré dans le treillis car c'est la borne supérieure de A_i et de $f(o_{k+1})$ (*lignes* 11 *à* 13, *à la sortie de la boucle POUR Inserer=Vrai*).

Dans le cas où Z est vide $\forall A_j$, il faut insérer $f(o_{k+1})$ et Z dans le treillis (*lignes* 20 *à* 22). Z devient le concept supremum du treillis L^{k+1}. L'infimum n'est pas modifié puisque les auteurs de GALOIS initialisent la construction du treillis L^0 en faisant l'union des propriétés de chaque objet.

Remarque 4 *Dans les deux cas où il y a création d'un nouveau concept,* A_i *est un concept générateur [GOD 91] pour le concept* $\{A_i \bigcap f(o_{k+1}), g(A_i \bigcap f(o_{k+1}))\}$.

L'algorithme de mise à jour (voir figure 7) consiste à déterminer le cas correspondant à chaque concept du treillis lorsqu'on ajoute un nouvel objet. Une version améliorée de cet algorithme de mise à jour a été publiée par les auteurs du système GALOIS dans [CAR 96]. Cette version diffère de l'algorithme de génération incrémentale de treillis développé par Godin, appelé aussi GALOIS [GOD 95] dont une présentation synthétique est faite dans [VAL 02]. Le lecteur intéressé par l'implémentation du système de classification supervisée GALOIS [CAR 93] peut choisir d'implémenter l'algorithme de Godin [GOD 95]. La figure 8 présente les quatre premières étapes de la construction du treillis du contexte courant (voir figure 1).

```
Mise-à-jour(o_{k+1}, L^k)
DEBUT
1.  L^{k+1} ⟵ L^k
2.  EmptyCase ⟵ Vrai
3.  POUR A_i ∈ L^k FAIRE
4.    Z ⟵ f(o_{k+1}) ⋂ A_i
5.    SI Z ≠ {} ALORS
6.      EmptyCase ⟵ Faux
7.      Inserer ⟵ Vrai
8.      POUR TOUT A_j sur-concept de A_i FAIRE
9.        SI A_j ⊇ Z ALORS
10.         Inserer ⟵ Faux
11.       SINON SI A_j ⊂ Z ALORS
12.         STOP. Aller à la ligne 15.
13.       FSI
14.     FPOUR
15.     SI Inserer = Vrai ALORS
16.       L^{k+1} ⟵ L^{k+1} ⋃ Z
17.     FSI
18.   FSI
19. FPOUR
20. SI EmptyCase = Vrai ALORS
21.   L^{k+1} ⟵ L^{k+1} ⋃ {} ⋃ f(o_{k+1})
22. FSI
23. Retourner(L^{k+1})
FIN
```

Figure 7. *Algorithme de Carpineto et Romano*

6.2. *Apprentissage et Classement*

La phase d'apprentissage consiste à sélectionner les concepts cohérents générés qui satisfont la distribution existante. Le treillis des fermés de h' obtenu avec GALOIS sur l'exemple courant est identique à celui de la figure 2 lorsqu'on masque l'extension de chaque concept.

GALOIS utilise deux méthodes différentes pour affecter une classe à un objet. Au choix, le système effectue un *calcul de la similarité* entre le nouvel objet et chaque concept cohérent et maximal (pour l'inclusion). La similarité entre un objet et un concept est le nombre de propriétés du concept vérifiées par l'objet. Le système attribue ensuite à l'objet la classe du concept le plus similaire. GALOIS propose aussi un *processus empirique* qui attribue à chaque objet la classe majoritaire parmi les concepts pertinents qu'il vérifie.

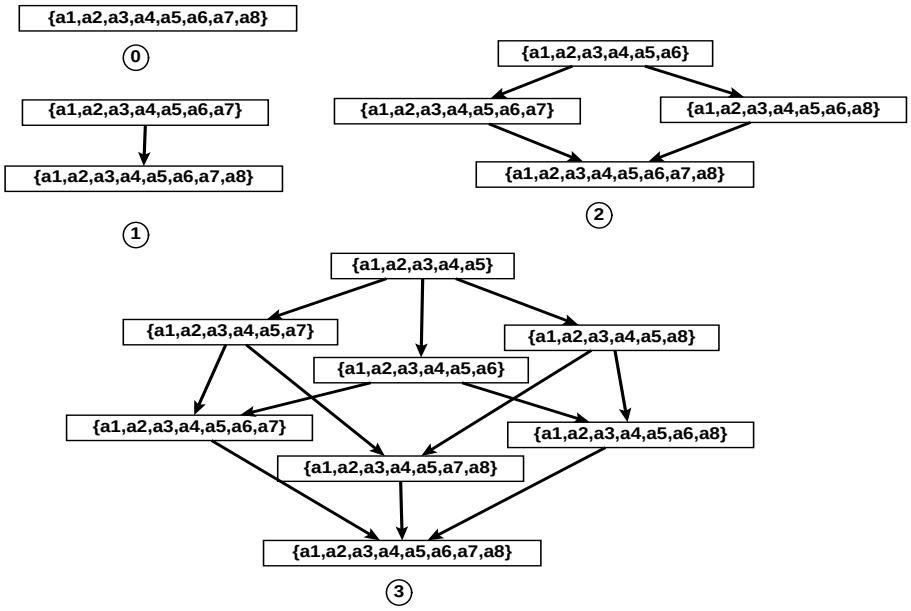


Figure 8. *Etapes de construction du treillis avec l'algorithme de Carpineto et Romano*

6.3. *Complexité et Comparaison expérimentale*

La complexité de la mise à jour des nœuds par un algorithme incrémental croît au fur et à mesure que de nouveaux objets sont considérés. Cette opération est effectuée en temps quadratique en fonction du nombre d'objets dans le treillis et exponentiel pour le nombre d'attributs.

Dans le pire des cas, la complexité en temps de GALOIS est égale à $O((v+1)^m \cdot v^m \cdot n) < O((v+1)^{2m} \cdot n)$, où v est le nombre maximal de valeurs que peut prendre un attribut. La complexité en temps de la construction incrémentale du treillis en fonction d'un nouvel exemple est comprise entre $O(l)$ et $O(l^2)$ où l est le nombre de concepts dans le treillis à mettre à jour.

Une implémentation de GALOIS avec le langage Common Lisp, a été evaluée sur quatre ensembles de données de la banque de données UCI [BLA 98] : Small Soybean, Congressional Voting, Breast Cancer, Iris.

La méthode d'évaluation n'est pas identique pour ces quatre ensembles. En outre GALOIS n'est pas comparé avec d'autres méthodes de classification. Les auteurs se limitent à fournir les résultats (intervalle de valeur du taux de précision et écart-type) obtenus en prédiction sur ces quatre ensembles, en ayant de manière générale repété dix exécutions du programme sur un ensemble de cinquante exemples d'apprentissage et cinquante exemples de test choisis aléatoirement dans l'ensemble initial. Les

résultats sont comparables aux meilleurs résultats reportés dans la littérature. Le choix du nombre 50 peut rendre les résultats non significatifs par rapport à l'ensemble initial. Ceci est dû principalement à un problème d'espace mémoire, lié à la génération entière du treillis de Galois.

Les temps d'exécution du programme sont commentés par les auteurs, mais ne sont pas fournis, ni comparés à ceux d'autres systèmes de classification utilisés dans la comparaison.

7. RULEARNER

Sahami a proposé le système RULEARNER [SAH 95] qui utilise le treillis de concepts sur les données pour organiser et orienter la recherche de règles de classification.

Notations : $SousCon(C)$ (resp. $SurCon(C)$) désigne l'ensemble des sous-concepts (resp. sur-concepts) du concept C. $Couverture(C)$ désigne l'ensemble des nœuds-exemples présents dans SousCon(C). Nous employons ici le terme concept pour désigner à la fois les concepts formels du treillis de Galois, et les nœuds-exemples et nœuds-attributs qui sont rajoutés dans le représentation du pseudo-treillis construit par l'algorithme de Oosthuisen.

7.1. *Principe et Construction du pseudo-treillis*

Ce système construit dans sa phase d'apprentissage un ensemble minimal de règles (ordonnées ou non) qui recouvrent les nœuds-exemples du treillis. Un paramètre de généralisation N, qui indique le pourcentage d'erreurs autorisées pour chaque règle, est fixé par l'utilisateur.

Définition 14 *Un concept est dit positif (resp. négatif) si le pourcentage des exemples négatifs (resp. positifs) couverts par le concept est en deçà de N. Sinon le concept est dit mixte, puisqu'il ne peut être étiqueté.*

RULEARNER utilise le pseudo-treillis de Galois généré par l'algorithme d'Oosthuisen. Le graphe construit par RULEARNER sur l'exemple courant est le même que celui obtenu avec le système GRAND en omettant les classes ($+$ et $-$) (voir section 4 sur le système GRAND).

7.2. *Apprentissage et Classement*

Tous les concepts sont considérés comme *actifs* au départ (*ligne* 1). RULEARNER associe à chaque concept du treillis un label représentant sa classe ou la valeur "mixte" en fonction du paramètre de bruit N (*ligne* 2). A chaque étape, une règle est générée

à partir du treillis. Le système sélectionne le concept actif non mixte qui recouvre le plus d'objets (*ligne* 4). S'il en existe plusieurs, il choisit celui ayant la plus petite intension. La règle ayant pour antécédent l'intension du concept sélectionné et pour conséquent la classe des exemples couverts est générée.

Les concepts dont tous les objets sont couverts par au moins une des règles générées sont marqués comme *inactifs* (*lignes* 8 *à* 16). La prochaine étape consiste à rechercher des règles pour l'ensemble des objets non couverts par la nouvelle règle. Le programme s'arrête lorsque tous les concepts du treillis sont inactifs. L'ensemble des règles obtenues forme une partition de l'ensemble d'apprentissage.

RULEARNER va générer un ensemble ordonné (liste de décision) ou non ordonné de règles de classification suivant le choix de l'utilisateur (*lignes* 17 *à* 20). Un inconvénient majeur de RULEARNER est sa propension à générer des règles trop spécifiques.

```
FindRules(L, C, N)
L : treillis de Galois
C : concepts du treillis
N : paramètre de bruit, nombre d'erreurs tolérées
DEBUT
1.  Initialiser tous les concepts du treillis L comme <actif>
2.  Etiqueter tous les concepts avec leur classe
3.  TANT QUE (il existe encore des concepts actifs)
4.    C_i ⟵ concept non mixte avec le plus d'exemples
5.    A_i ⟵ noeuds-attributs de SurCon(C_i)
6.    label ⟵ étiquette du concept C_i
7.    Produire la règle : <A_i ⟹ label>
8.    POUR TOUT (C_k ∈ SousCon(C_i) où C_k est un noeud-exemple)
9.      POUR TOUT (C_j ∈ SurCon(C_k))
10.       couverture(C_j) ⟵ couverture(C_j) − 1
11.       SI (couverture(C_j) ≤ 0)
12.         Marquer C_j comme <inactif>
13.       FSI
14.     FPOUR
15.     Marquer C_k comme <inactif>
16.   FPOUR
17.   SI (option <liste de décision> choisie)
18.     re-étiqueter les concepts actifs en fonction
19.     des noeuds-exemples encore actifs
20.   FSI
21. FTANTQUE
FIN
```

Figure 9. *Algorithme de génération de règles avec RULEARNER.*

Exemple 3 *Pour l'exemple courant, avec $N = 0$, le concept $C8$ de la fi gure 3 correspondant à $\{a_1, a_2, a_3, a_4\}$ sera sélectionné. C'est l'hypothèse de taille minimale qui recouvre tous les exemples positifs. La règle $a_1 \wedge a_2 \wedge a_3 \wedge a_4 \rightarrow +$ est générée.*

Pour classer un nouvel objet avec une liste ordonnée, RULEARNER applique les règles en respectant l'ordre sur les antécédents. La première règle vérifiée par l'objet est utilisée pour déterminer sa classe.

Dans le cas d'une liste non ordonnée de règles, RULEARNER utilise un système de vote pour déterminer la classe d'un nouvel objet.

7.3. *Complexité et Comparaison expérimentale*

Pour la construction du treillis, RULEARNER utilise l'algorithme de Oosthuizen à travers le programme GRAND. Il nécessite donc d'engendrer complètement le treillis de Galois de l'ensemble d'apprentissage pour débuter la génération de règles.

La discussion sur les temps de calcul porte sur des études antérieures effectuées par l'auteur montrant un ordre de complexité polynomial ($O(nm^3)$) pour la construction du treillis sur des jeux de données réelles [SAH 95].

RULEARNER a été comparé à $C4.5$ [QUI 93] et $CN2$ [CLA 89] sur les données Monks et Breast Cancer de la banque de données test d'apprentissage de l'UC-Irvine [BLA 98]. La technique de validation de type "holdout" est utilisée sur les données Monks. Dans le cas de Breast Cancer, cette technique est répétée trois fois et le résultat indique la moyenne et l'écart-type. Les résultats obtenus montrent qu'en moyenne sur ces jeux de données, RULEARNER a un taux de précision comparable, voire légèrement supérieur à celui des deux autres systèmes.

Aucune comparaison en termes de complexité en espace et en temps de calcul n'est présentée entre ces trois systèmes. En outre une comparaison avec le programme GRAND aurait pu être envisagée dans la mesure où l'auteur utilise le programme GRAND pour générer RULEARNER.

8. CIBLe

CIBLe (Concept Induction Based Learning) [NJI 99] est un système de classification supervisée qui opère en trois étapes successives : construction d'un sup-demi treillis de concepts, redescription numérique des données d'apprentissage et utilisation d'une mesure de similarité pour le classement de nouveaux exemples.

8.1. *Principe et Construction du treillis*

CIBLe est capable de traiter des problèmes contenant plusieurs classes et de prendre en compte des données décrites par des attributs symboliques et numériques. Les données sont préalablement séparées en fonction du type des attributs, puis le sup-demi treillis de concepts est construit en utilisant uniquement les attributs symboliques. Ensuite, CIBLe redécrit les données initiales en fonction de leurs interactions avec le sup-demi treillis. Finalement, le système utilise une technique de plus proches voisins dans sa phase de classement. Ces différentes étapes sont décrites ci-après.

Soient S le seuil de la fonction de sélection F, h la hauteur maximale du sup-demi treillis à construire, D l'ensemble des nouveaux descripteurs numériques, O l'ensemble des objets à classer, ϵ l'erreur de classement calculée sur les données d'apprentissage par la méthode k-PPV et suivant le protocole *leave one out* ; la stratégie globale de CIBLe est décrite par le pseudo-code de la figure 10.

La construction du sup-demi treillis est effectuée en appliquant une stratégie descendante et en largeur d'abord. L'algorithme d'induction utilisé est une version légèrement modifiée de celui proposé par Bordat [BOR 86] (comme avec LEGAL).

```
DEBUT
                    I-Construction du sup-demi treillis
1.  Construire le demi-treillis de hauteur h sur (O, A, I)
2.  Evaluer chaque nœud avec la fonction de sélection

                    II-Recherche de la valeur du seuil
3.  POUR T de 0 à  1 FAIRE
4.    Construire P = {{A_i, O_i} t.q. F({A_i, O_i}) ≤ T}
5.    Construire D = {a_j, t.q. ∃{A_t, O_t} ∈ P et a_j ∈ A_t}
6.    Redescription(O, D, P)
7.    Déterminer ε
8.  FINPOUR

                    III-Classement
9.  Donner à  T la valeur pour laquelle ε est minimale
10. Redescription(O, D, P)
11. POUR chaque exemple o_x FAIRE
12.   Redécrire o_x avec D et N
13.   Classer o_x avec la méthode PPV sur O
14. FINPOUR
FIN
```

Figure 10. *Stratégie globale d'induction de CIBLe*

CIBLe fait l'hypothèse selon laquelle seuls les concepts les plus généraux sont pertinents pour l'apprentissage. Ces derniers se trouvant a priori dans la partie supérieure du treillis, le système introduit un biais qui consiste à limiter artificiellement la hauteur du treillis à construire. Ce qui a pour effet de réduire sa complexité en espace et en temps. La construction du treillis s'arrête lorsque tous les nœuds du niveau h ont été obtenus ou lorsqu'il n'y a plus de nœuds à spécialiser (*ligne* 1). CIBLe construit le treillis sans tenir compte de la classe des exemples.

8.2. *Apprentissage et Classement*

Après la phase de construction, le sup-demi treillis de hauteur h forme l'espace de recherche pour la sélection des concepts pertinents. Cette sélection est effectuée en mesurant pour chaque concept la quantité d'information qu'il apporte au problème de la caractérisation des classes présentes dans les données d'apprentissage. Pour ce faire, CIBLe utilise une fonction de sélection basée sur une évaluation qualitative des concepts (*ligne* 2 *et* 4). Dans la pratique, l'utilisateur a le choix entre trois fonctions de sélection : l'entropie de Shannon utilisée dans $ID3$ [QUI 86], l'entropie améliorée [HUT 94] et la loi de succession de Laplace utilisée dans le système $CN2$ [CLA 89].

Définition 15 *Soient, F la fonction de sélection utilisée et S son seuil, un concept (A_i, O_i) est dit pertinent si $F(\{A_i, O_i\}) \leq S$.*

Définition 16 *Un attribut est dit pertinent s'il apparaît dans la description d'au moins un concept pertinent.*

CIBLe utilise l'ensemble d'apprentissage pour rechercher le meilleur seuil de sélection des concepts du treillis (*ligne* 3 *à* 10). Il itère en partant d'une valeur de seuil fixé (valeur par défaut : 0) avec un pas d'incrémentation de 0.01 pour choisir le seuil fournissant le taux d'erreur minimal en apprentissage.

CIBLe intègre aussi une méthode de redescription de l'espace de représentation (figure 11). A chaque attribut pertinent a_j, on fait correspondre un nouveau descripteur d_k. Chaque objet o_i prend alors comme valeur pour d_k la fréquence avec laquelle o_i est couvert par les concepts pertinents qui ont l'attribut a_j dans leur intension (*ligne* 5). On obtient de ce fait un nouvel espace de représentation à valeurs entièrement numériques, de taille réduite par rapport à l'espace de départ et basé uniquement sur des attributs pertinents. Il est à noter que la méthode de redescription proposée s'applique aussi lorsque la connaissance apprise est constituée par un ensemble de règles.

Dans l'exemple courant, pour $T = 0.66$ et $\epsilon = 0.0$, seuls trois concepts sont pertinents (représentés par des nœuds rectangulaires dans la figure 12) parmi les douze concepts du sup-demi treillis de hauteur $h = 3$. Ils sont totalement caractérisés par un sous-ensemble de cinq attributs $\{a_1, a_2, a_3, a_4, a_6\}$ pertinents parmi les huit attributs initiaux. Le tableau 13 présente le contexte initial redécrit avec CIBLe.

```
Redescription(O, D, P)
Renvoie O × D, la matrice redécrite
DEBUT
a.  POUR tout o_i ∈ O FAIRE
b.    POUR tout d_k ∈ D FAIRE d_ik ⟵ 0 FINPOUR
c.  FINPOUR
d.  POUR tout o_i ∈ O FAIRE
e.    P_i ⟵ {A_j t.q. {A_j, O_j} ∈ P, et o_i vérifie A_j}
f.    POUR tout A_j ∈ P_i FAIRE
g.      POUR tout a_k ∈ A_j FAIRE d_ik ⟵ d_ik + 1 FINPOUR
h.    FINPOUR
i.  FINPOUR
FIN
```

Figure 11. *Algorithme de redescription de CIBLe*

Chaque nouvel objet à classer est préalablement redécrit (*ligne* 12), puis sa classe est déterminée en utilisant un algorithme de type PPV (plus proche voisin) sur les exemples d'apprentissage (*ligne* 13). Cette méthode de classement fait l'hypothèse que les exemples les plus proches appartiennent forcément à la même classe.

Le fait que le nouvel espace de représentation soit entièrement numérique permet d'utiliser n'importe quelle "bonne" distance comme mesure de similarité. Dans la pratique, CIBLe propose trois mesures de similarité pour classer un objet à partir des éléments de l'ensemble d'apprentissage : la distance de Manhattan, la distance de Mahalanobis et la distance euclidienne.

8.3. *Complexité et Comparaison expérimentale*

Deux phases importantes dans CIBLe influencent sa complexité : la construction du sup-demi treillis et la recherche de la valeur optimale du seuil.

L'algorithme utilisé est basé sur celui de Bordat avec un paramètre supplémentaire qui consiste à limiter la hauteur du treillis et donc le nombre de nœuds construits.On a la formule suivante qui donne une borne supérieure pour le nombre de nœuds situés à la hauteur $h = i + 1$ du treillis : $|L_{i+1}| \leq |L_i| \cdot \min(n - i, m - i)$. On obtient une borne supérieure pour la taille du sup-demi treillis :

$$|L| \leq \sum_{i=0}^{h} |L_i| \leq \sum_{i=0}^{h} (\min(n-1, m-1))^i \leq (\min(n, m))^{h+1}$$

Par conséquent, la construction de chaque nœud se faisant en $\mathcal{O}(m^3)$, la complexité de l'algorithme dans le pire des cas est en $\mathcal{O}(\min(n, m)^{h+1} \cdot m^3)$. Ainsi si on

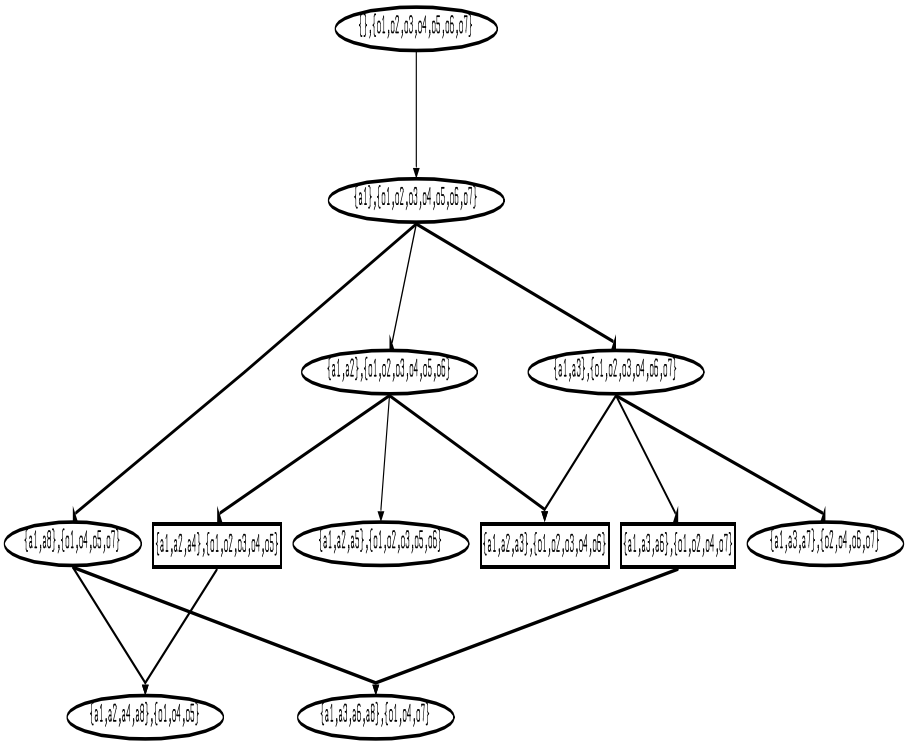


Figure 12. *Le sup-demi treillis construit par CIBLe sur le contexte initial avec* $h = 3$

	a_1	a_2	a_3	a_4	a_6	
O\D	d_1	d_2	d_3	d_4	d_5	Classe
o_1	3.00	2.00	2.00	1.00	1.00	+
o_2	3.00	2.00	2.00	1.00	1.00	+
o_3	2.00	2.00	1.00	1.00	0.00	+
o_4	3.00	2.00	2.00	1.00	1.00	+
o_5	1.00	1.00	0.00	1.00	0.00	-
o_6	1.00	1.00	1.00	0.00	0.00	-
o_7	1.00	0.00	1.00	0.00	1.00	-

Figure 13. *La redescription numérique du contexte initial avec CIBLe*

se limite à une profondeur réduite du treillis, on diminue d'un facteur exponentiel la complexité en temps et espace de génération des concepts pertinents de CIBLe.

Dans la pratique le pas d'incrémentation est de 0.01, ce qui implique au plus 100 parcours du treillis pour déterminer la pertinence de chaque nœud, d'où une complexité dans le pire des cas en $\mathcal{O}(\min(n,m)^{h+1})$.

Le système CIBLe a été comparé en termes de complexité et de capacité de prédiction avec des systèmes à base de treillis de concepts (LEGAL [MEP 94], IGLUE [NJI 97b] ainsi qu'avec des systèmes d'apprentissage utilisant des approches différentes, à savoir : $C4.5$ [QUI 93], K^* [CLE 95], IB1 [AHA 91] et KNNFP [AKK 96] qui font de l'apprentissage à partir d'instances. Pour ce faire, vingt trois ensembles de données provenant de la base de l'UC/-Irvine [BLA 98] ont été utilisés.

Les résultats (taux d'erreur) obtenus par CIBLe sont comparables à ceux des autres méthodes. Ils sont meilleurs en général sur les données contenant au plus deux classes et décrites uniquement avec des attributs nominaux. Cependant les écart-types ne sont pas mentionnés dans cette comparaison expérimentale.

Les résultats de CIBLe restent stables lorsqu'on fait varier les trois mesures de similarité de la phase de classement. Par contre, la fonction de succession de Laplace fournit des résultats en moyenne meilleurs que les deux autres fonctions de sélection implémentées dans le système (entropie, et entropie améliorée).

9. CLNB & CLNN

CLNB (Concept Lattices Naive Bayes) et CLNN (Concept Lattices Nearest Neighbour) sont deux systèmes de classification qui incorporent respectivement un classifieur bayésien naïf et un classifieur par plus proches voisins, dans le treillis de concepts.

9.1. *Principe et Construction du treillis*

A chaque concept du treillis vérifiant les contraintes de sélection est associé un classifieur contextuel. Si un nouvel exemple vérifie l'intension du concept alors on lui applique le classifieur correspondant pour déterminer sa classe en se limitant à l'extension du concept. Les deux systèmes reposent sur le même principe, tant en apprentissage qu'en décision. La seule différence réside dans le type de classifieur contextuel utilisé par l'un ou par l'autre.

L'idée sous-jacente consiste à utiliser le treillis de concepts pour sélectionner le sous-ensemble d'exemples d'apprentissage qui sera utilisé par l'un des classifieurs (bayésien naïf ou plus proche voisin). Le choix des treillis de concepts est motivé par le fait de vouloir éviter le problème du maximum local que l'on rencontre avec le système $NBTree$ [KOH 96] qui partitionne les exemples avec un arbre de décision avant d'appliquer le classifieur bayésien naïf, ou encore avec le système LBR [ZHE 00] qui est une amélioration du système $NBTree$.

Les auteurs ne donnent aucune précision sur l'algorithme utilisé en dehors du fait qu'il s'agit d'une approche descendante, par niveau et non incrémentale. Ils génèrent les concepts d'un sup-demi treillis et sélectionne comme pertinents les concepts qui vérifient les contraintes d'apprentissage.

9.2. *Apprentissage et Classement*

Ces deux systèmes ne traitent que des attributs nominaux. Ils utilisent l'algorithme de discrétisation basé sur l'entropie [FAY 93] pour traiter les attributs numériques.

Le principe d'apprentissage s'appuie sur une approche descendante guidée par les données. Il consiste à partir du concept le plus général (borne supérieure ; extension contenant tous les exemples) du treillis, et générer les sous-concepts immédiats qui satisfont trois contraintes (support, précision et rejet), et ensuite réitérer le processus avec les nouveaux concepts générés.

– contraintes de support : deux paramètres sont introduits. Le premier définit le nombre d'exemples minimum que doit contenir un concept. La valeur par défaut est fixée à 5%. Cette contrainte est identique au premier critère utilisé par $LEGAL$. En considérant notre exemple (voir figure 2, avec un seuil fixé à 3 objets sur 7, les concepts ayant les singletons ou doublons d'objets ne sont pas générés.

Le second paramètre définit le nombre d'exemples minimum différents qui doit exister entre un concept et un de ses sur-concepts. Ce nombre est inversement proportionnel au taux d'erreur du sur-concept. La valeur par défaut est fixée à trois divisé par le taux d'erreur du sur-concept. Plus le taux d'erreur associé à un sur-concept est faible (taux de précision élevé), plus le nombre d'exemples différents entre ce sur-concept et chacun de ses sous-concepts doit être élevé.

– contrainte de précision : ce paramètre limite la génération des concepts pour ne retenir que ceux dont le taux de précision associé est supérieur à celui associé à un sur-concept augmenté d'un nombre calculé en fonction du nombre d'exemples présents dans chacun de ses deux concepts. La valeur par défaut de ce nombre est fixée à 0. En considérant notre exemple avec un seuil à 0, si le sur-concept $(\{a_1, a_2, a_4\}, \{o_1, o_2, o_3, o_4, o_5\})$ généré est sélectionné et a un taux de précision égal à 80% alors le sous-concept $(\{a_1, a_2, a_4, a_5\}, \{o_1, o_2, o_3, o_5\})$ sera généré, mais pas sélectionné en apprentissage si son taux de précision est inférieur ou égal à 80%. Le sous-concept n'apporte rien de nouveau ni d'important par rapport à son sur-concept.

– contrainte de rejet : étant donné un concept et un de ses sur-concepts, si leurs nombres d'exemples sont proches, alors il ne faut pas sélectionner le concept. L'information additionnelle apportée par le concept est négligeable. La valeur par défaut du seuil pour ce paramètre est fixée à 90%, ce qui signifie que le rapport entre le nombre d'exemples du sous-concept et celui du sur-concept, doit être inférieur à 90%. Cette contrainte introduit une certaine flexibilité pour la génération des concepts. En considérant notre exemple avec un seuil à 75%, si le concept $(\{a_1, a_2, a_4\}, \{o_1, o_2, o_3, o_4, o_5\})$ est généré alors les concepts suivants :

$(\{a_1, a_2, a_4, a_5\}, \{o_1, o_2, o_3, o_5\})$ et $(\{a_1, a_2, a_3, a_4\}, \{o_1, o_2, o_3, o_4\})$ sont générés, mais ne sont pas sélectionnés en apprentissage. En effet l'apport des attributs a_5 et a_3 est considéré comme étant négligeable.

Notations : Support(r) est vrai si la règle r vérifie les 2 contraintes de support. Précision(r) est vrai si r vérifie la contrainte de précision. Rejet(r) est vrai si r vérifie la contrainte de rejet. R_i est l'ensemble des règles ayant pour antécédent un concept dont le nombre d'attributs est égal à i. $SousCon$ est l'ensemble des sous-concepts engendrés à partir du concept courant. CLS est le classifieur de base choisi ; NB ou NN (PPV).

```
DEBUT
1.  R_i ⟵ {} , i = 0, 1, ..., |A|
2.  R_|f(O)| ⟵ R_|f(O)| ⋃ {{f(O), O} ⟹ nil}
3.  POUR i = 0 to |A| − 1 FAIRE
4.    POUR chaque règle r : C ⟹ nil ∈ R_i FAIRE
5.      SI Support(r) et Rejet(r) ALORS
6.        Executer classifieur de case CLS sur |Extent(C)|
7.        r : C ⟹ CLS
8.        SI Précision(r) ALORS
9.          SousCon ⟵ {(f(g(Intent(C) ⋃ {a_k})), g(Intent(C) ⋃ {a_k}))
                  pour tout a_k ∈ A − Intent(C)}
10.         POUR chaque sous-concept C_j ∈ SousCon
                      tel que |Extent(C_j)| ≥ α * |O| FAIRE
11.           R_|Intent(C_j)| ⟵ R_|Intent(C_j)| ⋃ {C_j ⟹ nil}
12.         FPOUR
13.       SINON
14.         Enlever r de R_i et le supprimer
15.       FSI
16.     SINON
17.       Enlever r de R_i et le supprimer
18.     FSI
19.   FPOUR
20. FPOUR
FIN
```

Figure 14. *Algorithme d'apprentissage de $CLNN$ et $CLNB$*

L'algorithme commence par initialiser un ensemble de classes de règles R_i, chaque classe contenant les concepts ayant le même nombre d'attributs dans l'intension (*ligne* 1). En considérant le concept supremum $f(O), O$, on rajoute la règle $\{\{f(O), O\}\} \Longrightarrow nil$ dans $R_{|f(O)|}$ (*ligne* 2). On parcourt chaque ensemble de classes de règles R_i (*ligne* 3). Dans la mesure où l'on génére les concepts par niveau, et où le nombre d'attribut d'un sur-concept est inférieur au nombre d'attribut de ses sous-concepts, le sur-

concept sera toujours traité avant le sous-concept. En outre à partir de R_i on ne peut pas générer un concept dont le nombre d'attributs est inférieur à i. Pour chaque règle r dans R_i, si le concept associé ne vérifie pas les contraintes de support et de rejet alors elle est retirée de R_i (*d'où la condition de la ligne* 5). Si ces contraintes sont satisfaites alors on va exécuter le classifieur de base CLS (NB ou NN) (*ligne* 6) sur l'extension du concept. On modifie ensuite la règle r en $C \Longrightarrow CLS$. Si la contrainte de précision de r n'est pas vérifiée alors on retire r de R_i (*ligne* 14). Si cette contrainte est vérifiée, on génère les sous-concepts C_j (SousCon) du concept C associé à r en utilisant l'opération de fermeture par concaténation avec chaque attribut a_k non présent dans l'intension de C (*ligne* 9). Si le sous-concept vérifie C_j vérifie la première contrainte de support alors la règle est rajoutée dans $R_{|\text{Intent}(C_j)|}$ (*lignes* 10 *à* 12).

Les deux systèmes utilisent un principe identique de vote parmi l'ensemble des classifieurs pour affecter une classe à un nouvel exemple. Ce principe repose sur les quatre étapes suivantes :

– Marquer tous les classifieurs contextuels comme étant activés, si le nouvel exemple vérifie la règle associée au classifieur ;

– Etant données deux règles vérifiées r_1 et r_2 telles que l'intension de la partie gauche de r_1 est incluse dans l'intension de la partie gauche de r_2, désactiver le classifieur associé à r_1 ; on ne conserve que les règles maximales au sens de l'inclusion, vérifiées par le nouvel exemple ;

– Si r_2 a un taux de précision sur l'ensemble des données, qui est statistiquement plus significatif que celui de r_1, alors désactiver r_1 ; La mesure du χ_2 est utilisée pour ce test, avec une valeur par défaut égale à $3,84$.

– Faire un vote majoritaire avec les classifieurs contextuels activés ; en cas d'égalité entre classifieurs, choisir celui qui fournit la précision maximale.

9.3. *Complexité et Comparaison expérimentale*

Les auteurs ne donnent aucune indication sur la complexité théorique de leur système. Ils font référence à des travaux antérieurs montrant que la complexité est fonction du contexte initial. Bien qu'en théorie elle soit exponentielle dans le pire des cas en fonction de la taille des données, le pire des cas est rarement rencontré en pratique.

Il ressort de l'analyse de ces systèmes qu'un sup-demi treillis est généré, sur la base de la première contrainte de support qui permet de ne pas générer le treillis entier, de par la relation de monotonie. Ainsi la complexité de construction du sup-demi treillis est au moins identique à celle du système LEGAL.

La comparaison expérimentale est effectuée sur vingt-six ensembles test de l'UC/-Irvine [BLA 98]. Les taux d'erreur, calculés à partir d'une validation croisée d'ordre dix, sont indiqués et comparés à ceux de trois autres systèmes : $NBTree$ [KOH 96], CBA [LIU 98] qui est un système de classification basée sur les règles d'association,

et $C4.5Rules$ (version 8) [QUI 93]. Les valeurs des paramètres sont les valeurs par défaut.

Il ressort des résultats en prédiction qu'en général, les méthodes CLNB et CLNN améliorent respectivement la prédiction des méthodes NB et NN prises individuellement. En outre, CLNB obtient des résultats en moyenne supérieurs à ceux des trois autres systèmes sur ces jeux de données.

Les temps de calcul ne sont indiqués que pour CLNB et CLNN et aucune comparaison n'est faite avec les autres méthodes, afin de mesurer l'efficacité en temps des algorithmes par rapport à $C4.5Rules$, CBA et $NBTree$. De même les écart-types ne sont pas indiqués dans les tableaux de comparaison.

10. Discussion

Nous avons présenté six systèmes de classification à partir de treillis qui appliquent des méthodes différentes pour réduire l'espace de recherche, en introduisant généralement des biais d'apprentissage. Parmi ces systèmes, on peut noter deux approches de génération du classifieur. La première approche construit le treillis dans son ensemble, et ensuite l'explore pour engendrer le classifieur. Les méthodes concernées par cette approche sont : GRAND, GALOIS et RULEARNER. Ces méthodes ont en général la particularité d'utiliser un algorithme incrémental pour construire le treillis. La seconde approche autour des systèmes LEGAL, CIBLe, CLNN et CLNB n'engendre pas l'ensemble du treillis. Des heuristiques sont utilisées pour limiter la partie générée du treillis et permettre de réduire considérablement la taille de l'espace de recherche des hypothèses. Ces méthodes utilisent une démarche non incrémentale, descendante, et par niveaux, permettant d'élaguer le treillis à partir du biais d'apprentissage utilisé.

CIBLe, CLNN et CLNB sont des méthodes combinant une approche inductive et une autre technique de classification (PPV ou Bayésien naïf). Ces méthodes produisent des résultats qui viennent confirmer l'intérêt de la démarche multistratégique dans un processus de classification.

Nous récapitulons dans les deux tableaux (figures 15 et 16) les résultats de cette étude en fonction des critères utilisés : la façon dont le treillis est construit, la méthode d'apprentissage utilisée, la technique de classement employée, la complexité théorique et les expérimentations effectuées par les auteurs.

10.1. *Construction du treillis*

Pour la construction du treillis, trois algorithmes sont principalement utilisés (Bordat, Oosthuizen, Carpineto & Romano). Les deux derniers algorithmes sont incrémentaux. Les systèmes LEGAL, CIBLe, CLNN et CLNB ont la particularité de ne construire qu'un sup-demi treillis, ce qui réduit leur complexité théorique et leurs temps d'exécution.

10.2. *Apprentissage et Classement*

Le critère d'apprentissage est évalué suivant cinq éléments :

– la façon dont les données sont représentées : binaire, attribut/valeur, symbolique et numérique. Une représentation riche permet de mieux traiter les problèmes réels en évitant la perte d'information liée au prétraitement des données. C'est le cas des systèmes CIBLe, CLNB et CLNN. Il faut noter que le traitement des attributs numériques pose un vrai problème. Dans CIBLe, ils ne sont pris en compte que lors du calcul de la similarité entre objets (phase de classement), alors que dans les deux autres systèmes une étape de discrétisation est effectuée au préalable. Ces systèmes gagneraient à prendre en compte les travaux portant sur la construction de treillis de concepts sur des données complexes, comme les données symboliques définies par Diday et Polaillon [POL 98].

– le nombre de classes apprises simultanément. Seul LEGAL n'apprend qu'une seule classe à la fois, ce qui renforce la complexité de son utilisation dans le cas de données multi-classes. Le traitement de plusieurs classes n'est pas explicitement décrit dans la publication de RULEARNER. L'algorithme publié se limite à 2 classes (positif et négatif).

– la méthode de sélection des concepts utilisée : à l'exception de CIBLe qui utilise des méthodes statistiques pour mesurer la quantité d'information présente dans chaque concept, les autres systèmes utilisent des méthodes empiriques basées sur les notions de support, complétude, cohérence, précision, rejet et maximalité.

– la combinaison de méthodes : l'apprentissage multi-stratégique a fait ses preuves en montrant que la combinaison de deux approches différentes permet de bénéficier des avantages de chacune. Cette démarche permet en général d'éviter le sur-apprentissage. Les systèmes CIBLe, CLNB et CLNN proposent d'induire des concepts par la construction d'un sup-demi treillis, puis d'appliquer dans la phase de classement une autre technique (ici classifieur bayésien naïf, k-PPV ou vote majoritaire) sur la connaissance apprise (concepts sélectionnés ou règles de décision).

– la connaissance apprise. Elle est à la base constituée par les concepts sélectionnés (concepts pertinents) dans le treillis. Alors que certains systèmes choisissent de la conserver sous cette forme, les autres (GRAND, RULEARNER, CLNN et CLNB) extraient des règles. Ces dernières ont l'avantage d'être plus compréhensibles pour l'utilisateur. Cependant la relation d'ordre matérialisée par les liens de dépendance entre les concepts est perdue lors de la génération des règles. Il y a donc un choix à faire entre la compréhension et la richesse de la connaissance acquise à partir des données.

Pour le classement des nouveaux objets, la méthode k-PPV est la plus utilisée (avec $k = 1$ souvent). Un vote majoritaire est appliqué sur la connaissance apprise (sous-ensemble de règles ou concepts) qui caractérise le mieux le nouvel objet.

Systèmes	GRAND	LEGAL	GALOIS	RULEAR-NER	CIBLe	CLNN & CLNB
Type de Treillis	complet	partiel demi-treil	complet	complet	partiel demi-treil	partiel demi-treil
Algorithme	Oosthuizen	Bordat	Carpineto & Romano	Oosthuizen	Bordat	?
Données	binaires	binaires	attribut/ val symb	attribut/ val symb	symbol./ numériq.	symbol./ numériq.
Nbre de classes	multi-classes	une classe	multi-classes	multi-classes	multi-classes	multi-classes
Sélection de concepts	cohérence maximalité	cohérence maximalité	cohérence maximalité	support maximalité	hauteur fonction selection	support précision rejet
Combinaison de méthodes	non	non	non	non	induction + k-PPV	induction + k-PPV ou NB
Connaissance apprise	règles	concepts pertinents	concepts pertinents	règles	concepts pertinents	règles
Classement	vote	vote	similarité ou vote	règle la + gle ou vote	k-PPV	règles et votes

Figure 15. *Tableau récapitulatif des propriétés des différents systèmes*

10.3. *Complexité et Expérimentations*

En ce qui concerne l'estimation de l'ordre de complexité, les systèmes CLNN et CLNB ne fournissent aucun élément d'information. On a déjà vu que l'algorithme utilisé par ces deux systèmes pour l'induction du treillis n'était pas explicité. Pour les autres systèmes, la complexité exponentielle apparaît clairement dans la formule, à l'exception de CIBLe pour lequel on a un polynôme de degré $h + 4$.

Les données utilisées par les différents systèmes pour leurs expérimentations proviennent en majorité de la base de l'UCI. L'importance de cette base de données n'est plus à démontrer. On peut regretter qu'en dehors de LEGAL, les autres systèmes n'ont pas été appliqués sur des problèmes concrets. Le nombre de données traitées par chaque système est généralement faible (moins de cinq), sauf pour CIBLe, CLNN et CLNB (plus de vingt chacun). Il serait intéressant de qualifier la base de l'UCI en fonction de la difficulté des concepts à caractériser, et de déterminer pour chaque problème le taux d'erreur minimal admissible. Il serait alors possible dans ces conditions de valider un système en se basant sur les résultats obtenus.

Le holdout ou la validation croisée sont les méthodes utilisées par cinq des six systèmes pour mesurer leurs taux d'erreur en prédiction. Il s'avère que ces deux méthodes sont parmi celles qui fournissent les résultats les moins discutables. Seul le système GALOIS utilise une méthode de validation contestable sans aucune fiabilité statistique (sélection de 50 objets d'apprentissage et de 50 objets de test).

Systèmes	**Complexité Apprentissage**	**Comparaison expérimentale**	
		Paramètres	**Autres systèmes**
GRAND	$\mathcal{O}(2^m \cdot m^4)$	UCI : 2 validation : holdout	Assistant, AQ15, AQR, Bayes, CN2 résultats indicatifs
LEGAL	$\mathcal{O}\left(\lvert L\rvert \cdot n \cdot (1-\alpha)\right)$.	UCI : 4 validation : 5-VC	C4.5, IB1, GLUE résultats expérimentaux
GALOIS	$\mathcal{O}((v+1)^{2m} \cdot n)$	UCI : 4 validation imprécise	
RULEARNER	$\mathcal{O}(2^m \cdot m^4)$	UCI : 2 validation : holdout	$C4.5$, $CN2$ résultats expérimentaux
CIBLe	$\mathcal{O}(m^{h+4})$	UCI :23 validation : 5-VC	LEGAL, IGLUE, C4.5, K^*, IB1 et KNNFP résultats expérimentaux
CLNN CLNB	pas d'étude des auteurs $\mathcal{O}(\lvert L\rvert \cdot n \cdot (1-\alpha))$.	UCI : 26 validation : 10-VC	$NBTree$, CBA, C4.5 résultats expérimentaux

Figure 16. *Complexité et comparaison expérimentale des différents systèmes*

Cette étude vient aussi confirmer que le principal système de référence en apprentissage supervisé reste C4.5. Ce dernier est régulièrement utilisé par les différents auteurs pour comparer et justifier la pertinence des résultats obtenus avec leurs systèmes. On peut déplorer qu'en dehors des systèmes LEGAL et CIBLe, aucune comparaison expérimentale n'est effectuée entre les différents systèmes à base de treillis.

Toutefois ces systèmes introduisent généralement un certain nombre de paramètres (et donc de valeurs par défaut) pour guider l'apprentissage. Aucune étude de la pertinence de ces paramètres n'est effectuée dans la littérature. Pour des questions d'usage il peut être très difficile pour un utilisateur novice de recourir à ces méthodes pour un problème donné. De plus, il paraît indispensable de faire une comparaison expérimentale de l'ensemble de ces systèmes, en utilisant des procédures de comparaison identiques, et sur un grand nombre de jeux de données tests. Cette comparaison expérimentale aura le mérite de fournir des indications sur ces systèmes basés sur les treillis de Galois.

En outre l'ensemble de ces systèmes n'exploitent pas les connaissances du domaine qui pourraient permettre d'améliorer leur complexité et leur efficacité. En effet les connaissances du domaine peuvent permettre de faire un prétraitement sur les données initiales, conduisant à une simplification du contexte initial et à une rapide génération du treillis. Elles peuvent permettre de guider l'exploration des concepts pertinents par élagage du treillis, évitant ainsi la génération de l'ensemble du treillis.

11. Conclusion

Dans cet article, après avoir défini le problème de la classification, nous avons présenté le treillis de concepts (ou de Galois) et plusieurs systèmes de classification basés sur cette structure mathématique. A travers les expérimentations pratiques de chacun de ses systèmes, on peut conclure que le treillis de Galois offre un cadre tant pour la classification supervisée que pour la classification non supervisée. En effet, la construction du treillis peut se faire en tenant compte ou non de la classe des éléments du contexte.

L'utilisation du treillis de Galois permet de décomposer le concept à caractériser en sous-concepts. Il est alors plus aisé de rechercher une bonne caractérisation d'un ensemble réduit de données. Dans cette approche, les nœuds du treillis qui caractérisent chacun un sous-concept, possèdent une double représentation. L'une est constituée par l'hypothèse qui généralise les exemples du sous-concept concerné (l'intension). L'autre utilise l'ensemble des exemples du sous-concept (l'extension). C'est une instanciation du concept qui peut s'intégrer dans un mécanisme de généralisation basée sur les exemples. Il est donc possible, avec cette structure, de choisir la représentation adéquate en fonction du problème posé. Pour chaque concept, on dispose de ses exemples concrets et de sa caractérisation. Nous avons aussi constaté que la qualité de la connaissance issue du treillis est liée au *choix des concepts* (nœuds) à utiliser. La notion de concept formel est comparable à la notion d'hyperrectangle de dimension 1 qui sous-tend des systèmes tels que NGE [WET 94] ou Tabata [BRé 98]. En outre Ying et Wang [YIN 02] présentent un modèle algébrique général de conjectures et d'hypothèses basé sur les treillis et extrapolent sur le fait que des techniques futures en extraction de connaissances s'appuiront sur ce modèle pour aider les chercheurs en physique quantique dans leurs travaux de recherche.

Pour s'attaquer aux grandes bases de données, il semble utile de mettre en oeuvre des algorithmes permettant de traiter un volume de données important. Pour cela l'usage de la mémoire secondaire ou du parallélisme constitue une voie de recherche pouvant accroître le recours à l'utilisation des méthodes basées sur les treillis pour des applications réelles. Le traitement de données floues devra aussi être pris en considération dans les systèmes de classification à partir de treillis de concepts [BEL 00].

Remerciements

Nous remercions les rapporteurs pour leurs remarques détaillées et constructives sur la version soumise. Ce travail a bénéficié du support financier de l'IUT de Lens, de l'Université d'Artois, de la région Nord-Pas-de-Calais, et du CNRS par l'action spécifique Gafodonnées.

12. Bibliographie

[AHA 91] AHA D., KIBLER D., ALBERT M., « Instance-Based Learning Algorithms », *Machine Learning*, vol. 6, 1991, p. 37-66.

[AKK 96] AKKUS A., GÙVENIR H., « K-Nearest Neighbor Classification on Feature Projections », *Proc ICML*, Bari, Italy, Juillet 1996, p. 12-19.

[BAR 70] BARBUT M., MONJARDET B., *Ordre et Classification*, Hachette, Paris, 1970.

[BEC 01] BECKER P., « Multi-dimensional Representations of Conceptual Hierarchies », *Extracting and Representing Semantics. Contributions to the 9th ICCS.*, vol. 41, http ://CEUR-WS.org/Vol-41/, July 2001, p. 145-158.

[BEL 00] BELOHLAVEK R., « Similarity Relations in Concept Lattices », *Journal of Logic Computation*, vol. 10, n° 6, 2000, p. 823-845.

[BIR 40] BIRKHOFF G., *Lattice Theory*, American Mathematical Society, Providence, RI, 1st édition, 1940.

[BIR 67] BIRKHOFF G., *Lattice Theory*, American Mathematical Society, Providence, RI, 3rd édition, 1967.

[BLA 98] BLAKE C., KEOGH E., MERZ C., « UCI Repository of machine learning databases », 1998, http ://www.ics.uci.edu/∼mlearn/MLRepository.html.

[BOR 86] BORDAT J., « Calcul pratique du treillis de Galois d'une correspondance », *Mathématiques, Informatiques et Sciences Humaines*, vol. 24, n° 94, 1986, p. 31-47.

[BRé 98] BRÉZELLEC P., SOLDANO H., « Tabata : a learning algorithm performing a bidirectional search in a reduced search space using a Tabu strategy », *Proceedings of ECAI'98*, 1998, p. 420-424.

[CAR 93] CARPINETO C., ROMANO G., « Galois : An order-theoretic approach to conceptual clustering », *Proceedings of ICML'93*, Amherst, Juillet 1993, p. 33-40.

[CAR 96] CARPINETO C., ROMANO G., « A Lattice Conceptual Clustering System and its Application to Browsing Retrieval », *Machine Learning*, vol. 24, 1996, p. 95-122.

[CHE 69] CHEIN M., « Algorithme de recherche des sous-matrices premières d'une matrice », *Bulletin Math. de la Soc. Sci. de la R.S. de Roumanie*, vol. 61, n° 1, 1969, Tome 13.

[CLA 89] CLARK P., NIBLETT T., « The CN2 Induction Algorithm », *Machine Learning*, vol. 3, 1989, p. 261-283.

[CLE 95] CLEARY J., TRIGG L., « K^* : An Instance-Based Learner Using an Entropic Distance Measure », *Proceedings of Machine Learning Conference*, Tahoe City, CA, USA, 1995, p. 108-114.

[COR 02] CORNUEJOLS A., MICLET L., *Apprentissage Artificiel : Concepts et Algorithmes*, Eyrolles, Paris, France, 2002, 630 pages.

[DIE 97] DIETTERICH T., « Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms », Research report, December 30 1997, Computer Science Dept., Oregon State University.

[DUQ 96] DUQUENNE V., « GLAD (General Lattice Analysis and Design) : A Fortran Program for a glad user », RIVAL I., Ed., *Proc. ORDAL*, Ottawa, Canada, Août 1996.

[DUQ 99] DUQUENNE V., « Latticial structures in data analysis », *Theoretical Computer Science*, vol. 217, n° 2, 1999, p. 407-436.

[FAY 93] FAYYAD U., IRANI K., « Multi-interval discretization of continuous-valued attributes for classifi cation learning », *Proc. IJCAI*, Chambéry, France, August 1993, p. 1022-1027.

[FU 03] FU H., MEPHU NGUIFO E., « How well go lattice algorithms on currently used machine learning testbeds ? », WILLE R., Ed., *First Intl. Conf. on Formal Concept Analysis*, Darmstadt, Allemagne, February 2003.

[GAN 84] GANTER B., « Two basic algorithms in Concept Analysis », rapport n° 831, 1984, Technische Hochschule, Darmstadt, Germany, preprint.

[GAN 87] GANASCIA J., « CHARADE : A Rule System Learning System », *IJCAI'87*, Milan, Italy, 1987, p. 345-347.

[GAN 93] GANASCIA J., « TDIS : an Algebraic Formalization », *Proceedings of IJCAI'93*, vol. 2, 1993, p. 1008-1013.

[GAN 99] GANTER B., « Attribute exploration with background knowledge », *Theoretical Computer Science*, vol. 217, n° 2, 1999, p. 215-233.

[GAN 00] GANASCIA J., « *Induction Symbolique Numérique à partir de données* », chapitre CHARADE & Fils : Evolutions, Applications et Extensions, CEPADUES, 2000, 23 pages, http ://www-poleia.lip6.fr/ ganascia/.

[GOD 91] GODIN R., MISSAOUI R., ALAOUI H., « Learning Algorithms using a Galois Lattice Structure », *Proc. ICTAI*, San José, CA, USA, Novembre 1991, IEEE, p. 22-29.

[GOD 95] GODIN R., MINEAU G., MISSAOUI R., « Incremental structuring of knowledge bases », *Proc. of the 1st Intl. Symp. on Knowledge Retrieval, Use and Storage for Efficiency (KRUSE'95)*, Santa Cruz, CA, USA, 1995, p. 179-198.

[GUé 91] GUÉNOCHE A., « Construction du treillis de Galois d'une relation binaire », *Mathématiques Informatique et Sciences Humaines*, vol. 109, 1991, p. 5-47.

[GUI 86] GUIGUES J., DUQUENNE V., « Familles minimales d'implications informatives résultant d'un tableau de données binaires », *Mathématiques, Informatique et Sciences Humaines*, vol. 24, n° 95, 1986, p. 5-18.

[HUT 94] HUTCHINSON A., *Algorithmic Learning*, Clarendon Press, Oxford, 1994, pp. 217-219, A more accurated estimate of entropy.

[KOH 96] KOHAVI R., « Scaling up the accuracy of naïve-Bayes classifi ers : A decision-tree hybrid », *Proceedings of the second International Conference on Knowledge Discovery and Data Mining*, Menlo Park, CA, USA, 1996, AAAI Press, p. 202-207.

[KOU 98] KOURIE D., OOSTHUIZEN G., « Lattices in Machine Learning : Complexity Issues », *Acta Informatica*, vol. 35, n° 4, 1998, p. 269-292.

[KUZ 01] KUZNETSOV S., OBIEDKOV S., « Comparing Performance of Algorithms for Generating Concept Lattices », MEPHU NGUIFO E., DUQUENNE V., M. L., Eds., *Proc. of ICCS workshop on Concept Lattices for Knowledge Discovery in Databases (CLKDD)*, vol. 42, Stanford, CA, USA, July 2001, http ://CEUR-WS.org/Vol-42/, p. 35-47.

[LIQ 90] LIQUIÈRE M., MEPHU NGUIFO E., « LEGAL : LEarning with GAlois Lattice », *Actes des JFA*, Lannion, France, Avril 1990, p. 93-113.

[LIU 98] LIU B., HSU W., MA Y., « Integrating classifi cation and association rule mining », *Proceedings of 4th International conference on Knowledge Discovery and Data Mining*, New-York, NY, USA, 1998.

[MEP 93] MEPHU NGUIFO E., SALLANTIN J., « Prediction of Primate Splice Junction Gene Sequences with a Cooperative Knowledge Acquisition System », HUNTER L. et al., Eds., *Proc. of* 1^{st} *ISMB*, Washington, DC, USA, July 1993, AAAI Press, p. 292-300.

[MEP 94] MEPHU NGUIFO E., « Galois Lattice : A framework for Concept Learning. Design, Evaluation and Refinement », *Proceedings of ICTAI'94*, New Orleans, LA, USA, Novembre 1994, IEEE Press, p. 461-467.

[MEP 98] MEPHU NGUIFO E., LAGRANGE M.-S., RENAUD M., SALLANTIN J., « PLATA : An Application of LEGAL, a Machine Learning Based System, to a Typology of Archeological Ceramics », *Computers and the Humanities*, vol. 31, 1998, p. 169-187, Kluwer Academic Pub.

[MIC 86] MICHALSKI R., MOZETIC I., HONG J., LAVRAC N., « The Multi-purpose Incremental Learning System AQ15 and its Testing Application to three Medical Domains », KEHLER T., ROSENSCHEIN S., Eds., *Proceedings of 5th AAAI Conference*, Philadelphia, PA, USA, August 11-15 1986, AAAI Press, p. 1041-1045.

[MIT 82] MITCHELL T., « Generalization as Search », *Artificial Intelligence*, vol. 18, n° 2, 1982, p. 203-226.

[NJI 97a] NJIWOUA P., MEPHU NGUIFO E., « Forwarding the choice of bias, LEGAL-F : Using Feature Selection to Reduce the Complexity of LEGAL », DAELEMANS W. et al., Eds., *BENELEARN-97*, Tilburg University, The Netherlands, October 1997, p. 89-98.

[NJI 97b] NJIWOUA P., MEPHU NGUIFO E., « IGLUE : An Instance-based Learning System over Lattice Theory », *Proceedings of ICTAI'97*, Newport Beach, CA, USA, November 1997, IEEE Press, p. 75-76.

[NJI 99] NJIWOUA P., MEPHU NGUIFO E., « Améliorer l'apprentissage à partir d'instances grâce à l'induction de concepts : le système CIBLe », *Revue d'Intelligence Artificielle (RIA)*, vol. 13, n° 2, 1999, p. 413-440, Hermès Science.

[NJI 00] NJIWOUA P., « Contribution à l'apprentissage symbolique automatique par l'usage du treillis de Galois », Thèse de Doctorat d'Université, Université d'Artois, Janvier 2000, 179 pages.

[NOR 78] NORRIS E., « An algorithm for computing the maximal rectangles in a binary relation », *Revue Roumaine Math. Pures et Appl.*, vol. XXIII, n° 2, 1978, p. 243-250.

[NOU 99] NOURINE L., RAYNAUD O., « A Fast Algorithm for Building Lattices », *Information Processing Letters*, vol. 71, 1999, p. 199-204.

[OOS 88] OOSTHUIZEN G., « The use of a Lattice in Knowledge Processing », Thèse d'Université, University of Strathclyde, Glasgow, 1988.

[OOS 91] OOSTHUIZEN D., « Lattice-Based Knowledge Discovery », *Proceedings of AAAI'91 Knowledge Discovery in Databases Workshop*, Annaheim, 1991, p. 221-235.

[OOS 94] OOSTHUIZEN G., « The Application of Concept Lattices to Machine Learning », Technical Report n° CSTR 94/01, 1994, Department of Computer Science, University of Pretoria, Pretoria, South Africa.

[PAS 99] PASQUIER N., BASTIDE Y., TAOUIL R., LAKHAL L., « Efficient mining of association rules using closed itemsets lattices », *Journal of Information Systems*, vol. 24, n° 1, 1999, p. 25-46.

[POL 98] POLAILLON G., « Organisation et interprétation par les treillis de Galois de données de type multivalué, intervalle ou histogramme », Thèse d'Université, Université Paris IX-Dauphine, 1998.

[QUI 86] QUINLAN J., « Induction of Decisions Trees », *Machine Learning*, vol. 1, 1986, p. 81-106.

[QUI 93] QUINLAN J., *C4.5 : Programs for Machine Learning*, Morgan Kaufmann Publishers, Inc., Los Altos, California, 1993.

[SAH 95] SAHAMI M., « Learning Classifi cation Rules Using Lattices », *Proc ECML*, Heraclion, Crete, Greece, Avril 1995, N. Lavrac and S. Wrobel eds., p. 343-346.

[SEB 02] SEBAG M., GALLINARI P., « Apprentissage artifi ciel : acquis, limites et enjeux », LE MAITRE J., Ed., *Actes des 2èmes assises nationales du GDR i3*, Nancy, France, Décembre 2002, p. 303-333.

[VAL 02] VALTCHEV P., MISSAOUI R., GODIN R., MERIDJI M., « Generating frequent itemsets incrementally : two novel approaches based on Galois lattice theory », *Journal JETAI*, vol. 14, n° 2/3, 2002, p. 115-142.

[VIG 96] VIGNAL L., D'AUBENTON CARAFA Y., LISACEK F., MEPHU NGUIFO E., ROUZE P., QUINQUETON J., THERMES C., « Exon prediction in eucaryotic genomes », *Biochimie*, vol. 78, n° 5, 1996, p. 327-334, Elsevier Science.

[VOG 95] VOGT F., WILLE R., « TOSCANA - a graphical tool for analyzing and exploring data », *Proc. GraphDrawing*, vol. 894, Heidelberg, October 1995, Springer, p. 226-233.

[WET 94] WETTSCHERECK D., « A Hybrid Nearest-Neighbor and Nearest-Hyperrectangle Algorithm », *Proceedings of ECML'94*, LNAI-784, 1994, p. 323-335.

[WIL 82] WILLE R., « Restructuring Lattice Theory », RIVAL I., Ed., *Symposium on Ordered Sets*, University of Calgary, Boston, 1982, p. 445-470.

[WIL 92] WILLE R., « Concept Lattices & Conceptual Knowledge Systems », *Computer Mathematic Applied*, vol. 23, n° 6-9, 1992, p. 493-515.

[XIE 02] XIE Z., HSU W., LIU Z., LEE M., « Concept Lattice based Composite Classifi ers for High Predictability », *Journal JETAI*, vol. 14, n° 2/3, 2002, p. 143-156.

[YIN 02] YING M., WANG H., « Lattice-theoretic models of conjectures, hypotheses and consequences », *Artificial Intelligence*, vol. 139, 2002, p. 253-267.

[ZHE 00] ZHENG Z., WEBB G., « Lazy learning of Bayesian rules », *Machine Learning*, vol. 41, n° 1, 2000, p. 53-84.