Enrichir la programmation logique inductive avec des prédicats flous

Enriching inductive logic programming with fuzzy predicates

Henri Prade Gilles Richard Mathieu Serrurier

IRIT UMR CNRS 5505, 118 route de Narbonne
31062 Toulouse, France

henri.prade@irit.fr, richard@irit.fr, serrurier@irit.fr

Résumé

L'intérêt d'introduire des prédicats flous lors de l'apprentissage de règles est double. D'une part, pour travailler avec des données numériques, cela permet une discrétisation plus flexible tout en préservant la lisibilité de la règle. D'autre part, cela augmente la puissance d'expression de ce que l'on apprend en considérant différents types de règles floues, qui peuvent décrire le comportement graduel entre les attributs ou l'incertitude relative aux conclusions des règles. Dans cet article, nous décrivons les différents types de règle floue en logique du premier ordre et une méthode pour les apprendre. Enfin, nous discutons de leur intérêt sur des exemples représentatifs.

Mots Clef

Programmation Logique Inductive, règle floue, apprentissage relationnel, discrétisation, degré de confiance

Abstract

The interest of introducing fuzzy predicates when learning rules is twofold. When dealing with numerical data, it allows us to have a more flexible discretization and to preserve the readability of the rule. Moreover, it enlarges the expressive power of what is learned by considering different types of fuzzy rules, which may describe gradual behaviors of related attributes or uncertainty pervading conclusions. This paper describes different types of first-order fuzzy rules and a method for learning each type. Finally, we discuss the interest of each type of rules on a benchmark example.

Keywords

Inductive Logic Programming, relational learning, fuzzy rule, discretization, confidence degree

1 Introduction

La Programmation Logique Inductive (PLI) [11] est un domaine de l'apprentissage qui définit un cadre formel général pour l'apprentissage de règles en logique du premier ordre, et pour qui des algorithmes efficaces ont été développés (Progol [8], FOIL [15],...). L'apprentissage relationnel peut être considéré comme un sous-domaine de la PLI qui traite de l'apprentissage dans les bases de données relationnelles, que l'on peut représenter en logique du premier ordre. Dans ce cas on peut se restreindre aux clauses de Horn sans symbole de fonction. La logique du premier ordre n'autorise pas les exceptions qui sont pourtant courantes lorsque l'on parle d'apprentissage. C'est pour cela que les probabilités ont été introduites en PLI. En effet, les probabilités apparaissent implicitement dans la procédure de contrôle de FOIL, où le calcul du gain associé à une règle met en jeu un degré de confiance calculé à partir d'une distribution de probabilités sur le domaine. Ce type de probabilité, avec les probabilités sur les mondes (définies à partir d'une distribution sur les interprétations de Herbrand) sont les notions de base des probabilités sur la logique du premier ordre discutées par Halpern [5]. Les probabilités sur le domaine sont utilisées pour capturer les informations de nature statistique. Dans ce contexte la probabilité d'une règle est obtenue à partir d'une distribution de probabilité sur un domaine et d'une interprétation en calculant la mesure de l'ensemble des valuations qui rendent la règle vraie dans cette interprétation. Il n'y a donc plus de notion de quantificateur dès que la probabilité de rencontrer une exception est non nulle. Les probabilités sur le domaine sont aussi utilisées en PLI dans le cadre des Stochastic Logic Programs [9]. Les probabilités sur les mondes sont utilisées en PLI dans le cadre des Bayesian Logic Programs [7]. Une vue unifiant les prédicats flous, les probabilités sur le domaine et les probabilités sur les mondes est présentée dans [12].

Un des principaux challenges de l'induction de règles à partir d'exemples est de gérer les attributs numériques et l'imprécision dans le cas d'attributs non-binaires. La méthode classique pour gérer les attributs à valeur réelle est de les transformer en attributs (symboliques)

qualitatifs par discrétisation. Les ensembles flous sont connus pour permettre une gestion graduelle des données numériques en évitant les problèmes de transition abrupte entre les différentes catégories. Dans le cadre de la logique propositionnelle, les degrés de confiance on été intégrés dans des méthodes d'apprentissage prenant en compte des propriétés floues. On peut distinguer au moins trois types de travaux différents concernant cette dernière préoccupation. Tout d'abord, les techniques d' "apprentissage neuro-flou" ont été développées pour calculer des fonctions d'appartenance floue dans des règles floues. Les règles floues induites de cette façon sont utilisées comme approximateurs de lois de commande en contrôle automatique. Une autre partie de la recherche concerne plus particulièrement la puissance descriptive des règles floues du point de vue de l'utilisateur en adaptant l'algorithme ID3 de Quinlan [14] et ses successeurs aux arbres de décision flous et en utilisant une description floue des classes puis une mesure d'entropie (étendue aux ensembles flous) pour construire les règles floues [2]. Plus récemment, les fonctions d'appartenance floue ont été utilisées par plusieurs chercheurs afin de créer des règles d'association pour la fouille de données ayant une puissance d'expression accrue [6].

Actuellement la majorité des méthodes pour extraire des règles floues sont propositionnelles. Dans le cadre de la PLI, une adaptation de FOIL a déjà été implémentée de manière à prendre en compte les degrés d'appartenance [17], mais le sens des règles induites reste classique. Dans cet article, dans le cadre de l'apprentissage relationnel, nous proposons une méthode pour induire des règles en logique du premier ordre qui peuvent contenir des prédicats flous. Dans un premier temps, nous expliquerons comment une base de données peut être décrite en termes de prédicats flous, et nous discuterons des différents types de règle récemment introduits dans une perspective d'apprentissage [13]. Pour chaque type de règle, l'algorithme FOIL est adapté en définissant les degrés de confiance correspondant à chacun de ces types. L'article est organisé de la façon suivante. La section 2 apporte un rappel sur la PLI. La section 3 présente les différents types de règle et l'interprétation floue des bases de données. La section 4 décrit notre algorithme et la section 5 illustre notre approche avec un exemple illustratif et un "benchmark".

2 Rappels sur la Programmation Logique Inductive

Nous rappelons brièvement les définitions usuelles et les notations. A partir d'un langage de la logique du premier ordre $\mathcal{L}$ avec un ensemble de variables Var, on construit l'ensemble des termes $Term$, des atomes $Atom$ et des formules. L'ensemble des termes clos est l'univers de Herbrand $\mathcal{H}$ et l'ensemble des atomes clos ou faits constitue la base de Herbrand $\mathcal{B} \subset Atom$. Un *littéral* l est un atome a (littéral positif) ou sa négation $\neg a$ (littéral négatif). Une substitution σ est une application de Var vers $Term$ avec une extension inductive pour $Atom$. Une *clause* est une disjonction finie de littéraux $l_1 \vee \ldots \vee l_n$ aussi notée $\{l_1, \ldots, l_n\}$. Une clause de Horn est une clause avec au plus un littéral positif. Une interprétation de Herbrand I est un sous-ensemble de $\mathcal{B}$: I est l'ensemble des formules atomiques closes vraies et son complémentaire est l'ensemble des formules atomiques closes fausses. On note $\mathcal{I} = 2^{\mathcal{B}}$ l'ensemble de toutes les interprétations de Herbrand. Nous pouvons maintenant définir la notion de conséquence logique.

Définition 1 *Soit A une formule atomique, $I, \sigma \models A$ signifie que $\sigma(A) \in I$. L'extension aux formules quelconques F se fait par composition. $I \models F$ signifie $\forall \sigma,\ I, \sigma \models F$ (on dit que I est un modèle de F).*
$\models F$ signifie $\forall I \in \mathcal{I},\ I \models F$.
$F \models G$ signifie que tous les modèles de F sont des modèles de G.

Dans le contexte de la logique du premier ordre, le but de la PLI est de trouver un ensemble de formules H tel que :

$$B \cup H \models E \tag{1}$$

étant donné un ensemble de connaissances B et un ensemble d'observations E (ensemble d'exemples), où E, B et H sont des ensembles de clauses. Un ensemble de formules est ici considéré comme la conjonction de ses éléments. Bien entendu on impose quelques restrictions :

- $B \not\models E$ car, dans le cas contraire, H ne serait pas nécessaire pour expliquer E.
- $B \cup H \not\models \perp$: cela signifie que $B \cup H$ est une théorie consistante.

Dans le cas des bases de données relationnelles, la PLI est souvent restreinte aux clauses de Horn sans symbole de fonction, et E est juste un ensemble de faits clos. De plus, l'ensemble E lui-même n'est pas considéré comme une solution acceptable car elle n'a aucune capacité de prédiction. En effet l'induction de règles est souvent associée à une compression de l'information contenue dans E.

Il existe principalement deux types d'algorithmes, les algorithmes *top down* et les algorithmes *bottom up* . Les algorithmes *top down* partent de la clause la plus générale et la spécialise pas à pas. Les procédures *bottom up* partent quant à elles d'un fait et elles le généralisent. Nous allons, dans cet article, utiliser FOIL [15] qui est un algorithme de type *top down*. Le but de FOIL est d'induire des règles jusqu'à ce que tous les exemples soit couverts. L'algorithme général de FOIL pour un un prédicat cible C est décrit dans Alg. 1.
Le calcul du gain est donné par la formule suivante :

$$gain(l \wedge A \rightarrow C, A \rightarrow C) =$$
$$n * (log_2(cf(l \wedge A \rightarrow C)) - log_2(cf(A \rightarrow C)))$$

Alg. 1 FOIL(B, E, C)

$H = \emptyset$
while $E \neq \emptyset$ **do**
  soit $A \rightarrow C$ la clause la plus générale avec $A = \top$
  while cf$(A \rightarrow C)<$ seuil **do**
    choisir un littéral l tel que $l \wedge A \rightarrow C$ maximise la fonction de gain
    $A = l \wedge A$
  end while
  $H = H \cup A$
  $E = E -$ {ensemble des exemples couverts par $B \wedge H$}
end while
return H

où n est le nombre d'exemples distincts couverts par $l \wedge A \rightarrow C$. Cette mesure de gain est inspirée de la mesure de la quantité d'information au sens de Shannon. Soit une clause de Horn $A \rightarrow C$, le degré de confiance est $cf(A \rightarrow C) = \frac{P(A \wedge C)}{P(A)}$. Le degré de confiance est calculé en accord avec les probabilités sur le domaine [5]. Dans le cadre de la PLI, il existe deux manières de décrire les exemples. La première est de décrire l'ensemble des exemples E^+ et l'ensemble des contre-exemples E^-. Une seconde consiste à ne mettre que les exemples positifs dans E et faire l'hypothèse du monde clos. C'est cette seconde manière que nous utiliserons dans cet article. Les données de la PLI ne décrivent qu'une seule interprétation sous l'hypothèse du monde clos. On appelle I_{PLI} cette interprétation. Ainsi, soit un fait f :

$$I_{PLI} \models f \text{ ssi } B \wedge E \models f.$$

Le domaine $\mathcal{H}$ est le domaine de Herbrand décrit par B et E. Si P est une mesure de probabilité uniforme sur $\mathcal{H}$, alors on déduit que le degré de confiance d'une clause $A \rightarrow C$, avec $\overrightarrow{t}$ le vecteur des n variables libres de la clause, est :

$$cf(A(\overrightarrow{x}) \rightarrow C(\overrightarrow{x}))_{I_{PLI}} = \frac{|\{\overrightarrow{t} \in \mathcal{H}^n \mid I_{PLI} \models \sigma[\overrightarrow{t}/\overrightarrow{x}](A(\overrightarrow{x}) \wedge C(\overrightarrow{x}))\}|}{|\{\overrightarrow{t} \in \mathcal{H}^n \mid I_{PLI} \models \sigma[\overrightarrow{t}/\overrightarrow{x}](A(\overrightarrow{x}))\}|} \quad (2)$$

où $|\ |$ dénote la cardinalité.

Une autre définition possible du degré de confiance peut être la proportion d'exemples positifs couverts par la règle par rapport au nombre total d'exemples couverts (positifs ou négatifs). Ce degré de confiance représente la probabilité qu'un fait déduit par une règle soit vrai. Mais cette définition ne prend pas en compte le fait qu'en logique du premier ordre, il peut exister plusieurs valuations d'une même règle qui permettent de déduire un exemple donné.

En PLI, le but est d'apprendre un concept représenté par un prédicat. E est l'ensemble de tous les faits concernant le prédicat cible. B est l'ensemble des faits concernant les autres prédicats. Les règles apprises sont donc (dans le cas non récursif) composées par des prédicats apparaissant dans B dans la partie condition et par le prédicat cible dans la partie conséquence.

3 Induction dans une base de données floue

3.1 Base de données floue et règles floues

Une base de données floue K avec des prédicats flous (e.g., lourd, jeune...) est un ensemble de faits positifs, chacun associé à un nombre réel dans $[0, 1]$. Par exemple, dans la section 5, on travaillera avec une base de données qui contient des faits tels que $(poids(a, lourd), 0.9)$ qui signifie que la voiture a est très représentative des voitures lourdes. Donc K est un ensemble de couples de la forme $(A(\overrightarrow{x}), \mu(A(\overrightarrow{x})))$ avec $\overrightarrow{x} \in \mathcal{H}^n$, $A(\overrightarrow{x})$ est un fait, et $\mu(A(\overrightarrow{x}))$ est le degré d'appartenance floue de $\overrightarrow{x}$ à A.

Il existe au moins deux raisons pour introduire des prédicats flous dans des règles en logique du premier ordre. Cela peut être pour rendre les règles plus flexibles ou bien plus expressives. En effet, un prédicat flou peut être considéré comme une famille de prédicats classiques avec comme fonctions caractéristiques les coupes de niveau μ_{F_α} associées aux fonctions d'appartenance floues μ_F de la façon suivante : $\mu_{F_\alpha}(\overrightarrow{x}) = 1$ si et seulement si $\mu(F(\overrightarrow{x})) \geq \alpha$ et $\mu_{F_\alpha}(\overrightarrow{x}) = 0$ sinon. Ainsi une règle floue "$A(\overrightarrow{x}) \rightarrow C(\overrightarrow{x})$" est naturellement associée aux règles non floues de la forme "$A_\alpha(\overrightarrow{x}) \rightarrow C_\beta(\overrightarrow{x})$". On peut remarquer que, si $A_\beta(\overrightarrow{x})$ est vrai alors $A_\alpha(\overrightarrow{x})$ est vrai aussi pour $\alpha \leq \beta$. Dans la suite, on ne considérera que les approximations non-floues du type "$A_\alpha(\overrightarrow{x}) \rightarrow C_\alpha(\overrightarrow{x})$", qui correspondent à un niveau uniforme de flexibilité des conclusions et des conséquences.

Si on est intéressé par la *flexibilité*, une signification possible de la règle "$A(\overrightarrow{x}) \rightarrow C(\overrightarrow{x})$" peut être

$$\forall \overrightarrow{x}, \exists \alpha \ \ A_\alpha(\overrightarrow{x}) \rightarrow C_\alpha(\overrightarrow{x}), \quad (3)$$

c'est-à-dire qu'il existe une manière non-floue d'interpréter la règle floue qui couvre chaque exemple (mais ce n'est pas forcément la même pour chaque exemple car α dépend de $\overrightarrow{x}$). Ce type de règle a déjà été étudié dans [12]. Par règles flexibles, on entend des règles qui sont robustes car leur prédicats peuvent couvrir des situations limites. La recherche de règles flexibles peut être assimilée à une induction de règles classiques assortie d'une étude de sensibilité. En effet, les contre-exemples d'une règle classique qui pourraient se transformer en exemples pourvu que l'extension du prédicat C soit un peu plus grande, ne doivent pas pénaliser autant la règle que des contre-exemples plus manifestes. On peut alors

aussi vouloir élargir d'une façon analogue l'extension de A pour pouvoir éventuellement couvrir plus d'exemples.

On peut aussi chercher à induire des règles plus *expressives* que celles permises par la logique classique. On peut ainsi chercher des règles qui sont *vraies* pour toute coupe de niveau. On a donc

$$\forall \overrightarrow{x}, \forall \alpha \;\; A_\alpha(\overrightarrow{x}) \rightarrow C_\alpha(\overrightarrow{x}). \qquad (4)$$

Clairement, ce type de règle est plus restrictif que (3) car il est équivalent à un *ensemble* de règles ordinaires correspondant à chaque niveau d'appartenance, résumé en une règle floue unique. Ainsi (4) n'est autre qu'une règle graduelle [4] signifiant "plus $\overrightarrow{x}$ satisfait A, plus $\overrightarrow{x}$ satisfait C" (ces règles sont représentées par une contrainte de la forme $\mu(A(\overrightarrow{x})) \leq \mu(C(\overrightarrow{x}))$).

Les règles graduelles constituent un des quatres types principaux de règles floues [4]. Deux de ces types de règle, nommés règles graduelles et règles à certitude, sont basés sur des connecteurs d'implications multi-valuées et expriment des contraintes sur les mondes modèles de la règle. Les deux autres types, nommés règles possibilistes et règles antigraduelles, expriment quant à eux que certaines valeurs sont garanties possibles (étant donné les exemples de la base). Les règles possibilistes sont de la forme "plus $\overrightarrow{x}$ est A, plus toutes les interprétations qui rendent C vraie sont garanties possibles" (la véracité dépend du degré d'appartenance quand C est flou). En pratique, cela signifie "plus $\overrightarrow{x}$ est A, plus il y a de bons exemples pour chaque interprétation possible de C". On remarque que, pour ce type de règle, il n'y a pas, à proprement parler, de contre-exemples, car on s'intéresse à la répartition des degrés de satisfaction pour chaque interprétation de C de façon globale.

Dans la suite, on ne considérera que les règles de type graduelle et à certitude. Les règles à certitude (qui ne doivent pas être confondues avec les règles possibilistes) expriment que "plus $\overrightarrow{x}$ est A, plus il est certain que $\overrightarrow{x}$ soit C". La règle signifie "plus grand est α tel que $\mu(A(\overrightarrow{x})) \geq \alpha$, moins la règle $A_\alpha(\overrightarrow{t}) \rightarrow C(\overrightarrow{t})$ a d'exceptions". En effet, plus α décroît, plus le nombre d'exceptions peut augmenter car la taille de A_α est élargie. Pour les règles à certitude, les exceptions sont donc davantage tolérables si elles correspondent à des interprétations limites des prédicats de la partie condition.

3.2 Application à la PLI

Il est bien connu qu'une des difficultés de l'apprentissage de règles est la gestion des attributs à valeur réelle. La présence d'attributs numériques rend généralement l'espace des hypothèses infini. La difficulté augmente d'autant plus quand des nombres réels apparaissent aussi comme arguments du prédicat cible. Ainsi les attributs numériques sont essentiellement traités de deux manières : soit par ajout de contraintes, soit par discrétisation. Introduire des contraintes en logique du premier ordre consiste à utiliser des opérateurs tels que les inégalités ou des fonctions mathématiques (moyenne, ...) [16]. Les règles induites de cette façon peuvent souffrir d'un manque de lisibilité et de généralité. De plus, les algorithmes qui permettent de gérer les contraintes dépassent le simple cadre de la résolution en logique du premier ordre. L'autre méthode est d'utiliser une discrétisation afin de transformer des informations continues en informations qualitatives [1]. Dans ce cas, on peut traiter les données dans le cadre standard de la logique du premier ordre. C'est la méthode la plus utilisée car elle permet de contourner le problème des données réelles tout en améliorant la lisibilité des règles. Cependant, comme les classes sont souvent définies a priori avant le processus d'induction, les règles produites dépendent de la qualité des classes. De plus, ces classes représentent souvent des prédicats qui ont un sens imprécis. Par exemple, pour la base de données auto-mpg de UCI [1], la valeur de mpg (consommation en miles par gallon) est représentée en termes des catégories "faible consommation", "consommation moyenne" et " consommation forte". Dans ce cas, des catégories floues, représentées par des ensembles flous, sont plus appropriées pour décrire la valeur de mpg car ils n'utilisent pas un seuil abrupt et arbitraire de séparation entre faible, moyenne et forte consomation de carburant (voir Fig. 1).

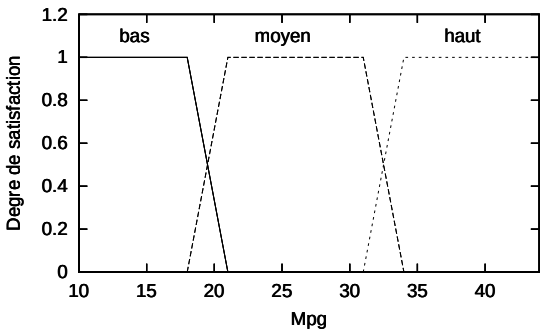

FIG. 1 – ensembles flous pour mpg

En conclusion, l'utilisation de prédicats flous permet de relaxer la rigidité des classes non-floues tout en gardant une grande lisibilité pour les règles induites. De plus, les différents types de règle que nous avons introduits permettent une meilleure description et de nouveaux types de résumé des données. Les *règles flexibles* peuvent être vues comme une adaptation floue des règles classiques au sens où il existe une lecture des prédicats flous, correspondant à une α-coupe de leur représentation, et conduisent à des règles mieux appropriées aux données. Les *règles graduelles* et les *règles à certitude* tirent parti des prédicats flous pour permettre une expressivité plus grande.

[1] http ://www.ics.uci.edu/ mlearn/MLRepository.html

Dans le cadre de la PLI, le but est de trouver des règles qui sont correctes et complètes par rapport aux exemples. Une hypothèse est correcte si elle ne couvre pas de fait sur le concept cible qui soit faux dans l'interprétation définie par les exemples et la connaissance de base. Elle est complète si elle couvre tous les exemples qui sont vraies dans l'interprétation précédente. Dans le cas de la PLI floue, la définition d'un exemple couvert par une règle dépend du type de la règle floue et du degré d'appartenance des faits qui sont validés par la règle.

4 Algorithme

Un certain nombre de facteurs cruciaux interviennent dans l'algorithme FOIL : le degré de confiance, la condition d'arrêt et le nombre d'exemples distincts couverts par la règle. Un exemple est un élément de E, un contre-exemple est un fait sur le prédicat cible qui est faux dans l'interprétation décrite par les données. Nous allons donc définir ces notions pour chaque type de règle (voir [12] [13] pour les détails).

Règles flexibles

Le premier sens possible envisagé ici pour une règle floue "$A(\overrightarrow{x}) \rightarrow C(\overrightarrow{x})$" est proche du sens classique. Bien sûr, on souhaite pour chacune des valuations $\sigma[\overrightarrow{t}/\overrightarrow{x}]$ que le degré de satisfaction de $A(\overrightarrow{t})$ et $C(\overrightarrow{t})$ soit le plus haut possible. On utilisera la définition d'une interprétation classique correspondant à chaque coupe de niveau α.

Définition 2 *Une α-interprétation I_α, étant donné un fait f, est définie par :*

$$I_\alpha \models f \;\; ssi \;\; B \wedge E \models f \; et \; \mu(f) \geq \alpha$$

Seuls les faits de B et E ayant un degré de satisfaction supérieur α sont considérés comme vrais dans I_α. Nous pouvons maintenant calculer le degré de confiance de manière classique (en utilisant (2)) pour chaque α-interprétation. Pour rester proche du sens de ce type de règle floue, on doit favoriser les degrés de confiance pour les hautes α-interprétations. En effet, on préfère que les exemples soient couverts avec un degré de satisfaction haut. La définition suivante, qui est une adaptation en logique du premier ordre de celle définie par [3], prend cela en compte :

$$cf_{flex}(A(\overrightarrow{x}) \rightarrow C(\overrightarrow{x})) =$$

$$\sum_{\alpha_i}(\alpha_i - \alpha_{i+1}) * cf(A(\overrightarrow{x}) \rightarrow C(\overrightarrow{x}))_{I_{\alpha_i}}$$

où $\alpha_1 = 1, ..., \alpha_t = 0$ est la suite décroissante des degrés de satisfaction apparaissant dans la base de données. Ce calcul correspond à une discrétisation d'une intégrale de Choquet sur les α-interprétations. On appelle $\overrightarrow{x_1}$ les q variables libres qui apparaissent dans la partie conséquence de la règle et $\overrightarrow{x_2}$ les r autres variables libres qui apparaissent dans la règle. On en déduit un comptage pondéré des exemples couverts distincts :
$n_{flex}(A(\overrightarrow{x}) \rightarrow C(\overrightarrow{x})) =$

$$\sum_{\alpha_i}(\alpha_i - \alpha_{i+1}) * |\{\overrightarrow{t_1} \in \mathcal{H}^q \; | \exists \overrightarrow{t_2} \in \mathcal{H}^r, \; I_{\alpha_i} \models$$

$$\sigma[\overrightarrow{t_1}, \overrightarrow{t_2}/\overrightarrow{x_1}, \overrightarrow{x_2}](A \wedge C)\}|$$

Les règles flexibles constituent la contrepartie floue des règles induites par une discrétisation non-floue, elles peuvent donc être utilisées au même titre que des règles classiques.

Règles graduelles
Pour ce type de règles, les degrés de satisfaction ne sont utiles que pour comparer les degrés de confiance des parties condition et conclusion de la règle. Il est donc inutile de favoriser les hauts degrés de satisfaction comme précédemment.

$$cf_{grad}(A(\overrightarrow{x}) \rightarrow C(\overrightarrow{x})) =$$

$$\frac{|\{\overrightarrow{t} \in \mathcal{H}^n \mid I_{PLI} \models \sigma[\overrightarrow{t}/\overrightarrow{x}](A \wedge C), \mu(\sigma[\overrightarrow{t}/\overrightarrow{x}]C) \geq \mu(\sigma[\overrightarrow{t}/\overrightarrow{x}]A)\}|}{|\{\overrightarrow{t} \in \mathcal{H}^n \mid I_{PLI} \models \sigma[\overrightarrow{t}/\overrightarrow{x}](A)\}|}$$

Le degré de satisfaction d'une conjonction de littéraux clos est le minimum des degrés de chaque littéral. On en déduit un comptage pondéré des exemples couverts distincts :

$$n_{grad}(A(\overrightarrow{x}) \rightarrow C(\overrightarrow{x})) =$$

$$|\{\overrightarrow{t_1} \in \mathcal{H}^q | \; \exists \overrightarrow{t_2} \in \mathcal{H}^r, I_{PLI} \models \sigma[\overrightarrow{t_1}, \overrightarrow{t_2}/\overrightarrow{x_1}, \overrightarrow{x_2}](A \wedge$$

$$C, \mu(\sigma[\overrightarrow{t_1}/\overrightarrow{x_1}]C) \geq \mu(\sigma[\overrightarrow{t_1}, \overrightarrow{t_2}/\overrightarrow{x_1}, \overrightarrow{x_2}]A)\}|$$

Les règles graduelles sont utiles quand les ensembles flous utilisés pour la discrétisation ont un noyau restreint et un support large. Dans ce cas il peut être difficile de trouver des règles flexibles car il peut y avoir dans la base une grande quantité de faits ayant un degré d'appartenance bas. Les règles graduelles, décrivant la façon dont évolue ensemble le degré d'appartenance à la partie condition et à la partie conséquence de la règle, ne sont pas sensibles à la valeur numérique proprement dite du degré d'appartenance et peuvent donc être induites dans de telles bases. Elles sont de fait, plus expressives, car décrivant des variations au travers d'un résumé d'un ensemble de règles classiques.

Règle à certitude
Le sens de la règle "$A(\overrightarrow{x}) \rightarrow C(\overrightarrow{x})$" est dans ce cas "plus $\overrightarrow{x}$ est A, plus il est certain que $\overrightarrow{x}$ soit C". Pour ce type de règle, on ne s'intéresse pas au degré de satisfaction de la partie conclusion. Les coupes de niveau α correspondent, pour ce type de règle aux interprétations classiques suivantes :

Définition 3 *Etant donné un fait f, on définit la satisfaction de f par une interprétation α-certaine $I_{\alpha-cert}$ de la manière suivante :*

$$I_{\alpha-cert} \models f \text{ ssi } (B \models f \text{ et } \mu(f) \geq \alpha) \text{ ou bien } E \models f$$

Comme on le voit dans cette définition, on ne se préoccupe plus ici du degré de satisfaction sur le concept cible. Avec les règles à certitude, le degré de confiance pour la règle doit être haut dans les interprétations α-certaines pour les hautes valeurs de α. Cela vient de l'idée que l'on est plus permissif par rapport aux exceptions de la contrepartie classique de la règle "$A(\overrightarrow{t}) \rightarrow C(\overrightarrow{t})$" correspondant aux petites valeurs de α. On utilise donc de nouveau une intégrale de Choquet.

$$cf_{cert}(A(\overrightarrow{x}) \rightarrow C(\overrightarrow{x})) =$$

$$\sum_{\alpha_i}(\alpha_i - \alpha_{i+1}) * cf(A(\overrightarrow{x}) \rightarrow C(\overrightarrow{x}))_{I_{\alpha_i-cert}}$$

On en déduit un comptage pondéré des exemples couverts distincts :
$$n_{cert}(A(\overrightarrow{x}) \rightarrow C(\overrightarrow{x})) =$$

$$\sum_{\alpha_i}(\alpha_i - \alpha_{i+1}) * |\{\overrightarrow{t_1} \in \mathcal{H}^q | \exists \overrightarrow{t_2} \in \mathcal{H}^r,\ I_{\alpha\ cert} \models$$

$$\sigma[\overrightarrow{t_1}, \overrightarrow{t_2}/\overrightarrow{x_1}, \overrightarrow{x_2}](A \wedge C)\}|$$

Ce type de règle peut être utilisé pour apprendre un concept non-flou.

Nous sommes désormais en mesure d'utiliser l'algorithme FOIL afin d'apprendre chaque type de règle définie précédemment en utilisant le calcul du degré de confiance et du nombre d'exemples couverts correspondants.

5 Résultats

5.1 Exemple illustratif

Considérons une base de données qui décrit 21 maisons dans une ville (Voir la Fig.2 pour un aperçu de la base). Les maisons sont décrites par des prédicats relationnels flous tels que $(proche(x, y), \alpha)$ qui signifie que la maison x est proche de la maison y à un degré α, ou $(connait(x, y), \alpha')$ qui signifie que le propriétaire de la maison x connaît le propriétaire de la maison y au degré α' ($\alpha' = 0$ pour inconnu, $\alpha' = 1$ pour ami). Les maisons sont aussi décrites par des prédicats flous propositionnels comme $cher(x)$ ou $petit(x)$.
Dans cette base, nous pouvons trouver des règles de chaque type avec un bon degré de confiance. Par exemple on trouve la règle *flexible*

$$proche(x, y), cher(y) \rightarrow cher(x),$$

avec un degré de confiance de 0.81, ce qui vient du fait que l'on peut raisonnablement s'attendre à ce que, si une maison est proche d'une maison chère, elle est chère aussi (les

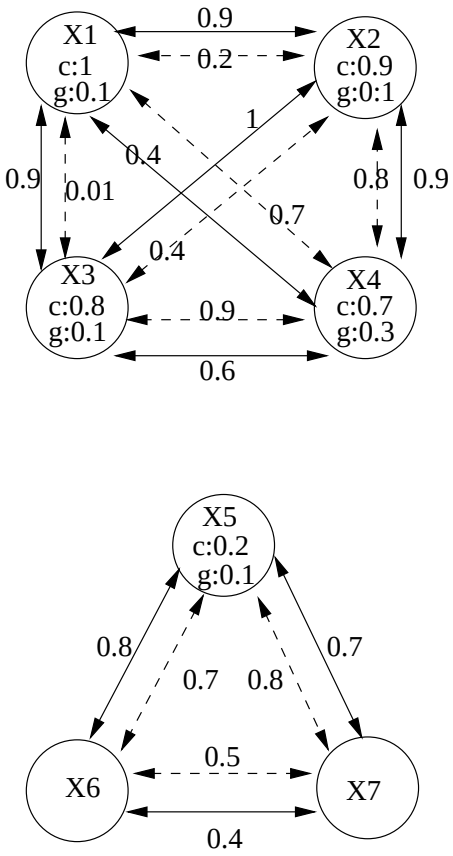

FIG. 2 – exemples de la base de données "ville"

maisons chères sont souvent regroupées dans les mêmes quartiers). Une règle *graduelle* typique est

$$grand(x) \rightarrow cher(x),$$

c'est-à-dire "plus la maison est grande, plus elle est chère". Son degré de confiance est 0.80. Un bon exemple de règle *à certitude* est

$$proche(x, y) \rightarrow connait(x, y)$$

avec 0.95 de degré de confiance. Cette règle signifie "plus les maisons sont proches, plus on est certain que les propriétaires se connaissent". Le fait que deux proches voisins se connaîssent est très plausible. Cette plausibilité peut décroître quand la distance entre les maisons augmente.

5.2 Benchmark

Nous utilisons tout d'abord la base de données "auto-mpg" du site UCI [2]. Cette base de données est constituée d'informations sur des voitures, le concept à apprendre est la consommation en ville de fuel en miles par gallons (mpg). Il y a 398 instances de voitures décrites par 9 attributs dont 5 sont continus, y compris le concept à apprendre. Cette base de données peut être décrite en logique propositionnelle. Tout d'abord, les prédicats continus de la base ont été transformés en prédicats flous en utilisant des ensembles flous décrits de la même façon que pour la consommation de fuel (Fig.1). En plus, trois discrétisations non-floues ont été effectuées de manière à générer trois bases de données différentes correspondant à i) la partition non-floue des attributs du domaine la plus proche de la partition floue (qui correspond à la coupe de niveau 0.5), ii) la partition non-floue correspondant au support des ensembles flous, iii) la partition non-floue correspondant au noyau des ensembles flous. Pour la base de données floue, on utilise la version floue de FOIL pour induire chaque type de règle. L'induction d'un type de règle est faite de manière indépendante des autres. On utilise l'algorithme de FOIL pour chacune des bases de données non floue. Pour chaque processus d'apprentissage, on indique dans un tableau le nombre de règles trouvées, le pourcentage d'exemples couverts et la moyenne des degrés de confiance des règles. Un exemple de règle induite par l'algorithme est donné pour chaque type.

type de règles	nbr. de règles	couverture	moy. cf
règles classiques	8	0.77	0.84
règles classiques sur le noyau	11	0.59	0.87
règles classiques sur le support	10	0.85	0.80
règles flexibles	3	0.51	0.84
règles graduelles	4	0.47	0.91
règles à certitude	2	0.62	0.75

[2] http ://www.ics.uci.edu/ mlearn/MLRepository.html

Les termes issus de la discrétisation (floue ou non) apparaissent en **gras**.
règle classique
$cylindres(A,'8') \rightarrow mpg(A,$**bas**$)$
règle flexible
$deplacement(A,$**lent**$), poids(A,$**moyen**$) \rightarrow mpg(A,$**moyen**$)$
règle graduelle
$poids(A,$**bas**$) \rightarrow mpg(A,$**bas**$)$
règle à certitude
$cylindres(A,'6'), poids(A,$**haut**$) \rightarrow mpg(A,$**bas**$)$

Nous utilisont ensuite la base de données "mutagenesis" qui est une base classique pour tester les algorithmes en PLI. Cette base est constituée d'informations sur des molécules, le concept à apprendre est le degré d'activité de celles-ci. Il y a 188 instances de molécule décrites par 7 prédicats dont 4 avec des attributs numériques, y compris le concept à apprendre. Le taux d'activité est initialement représenté par le prédicat "act(A,B)", où "B" est un attribut numérique qui décrit le taux d'activité de la molécule A. Cet attribut se discrétise facilement en non-flou car on considère en général que si sa valeur est négative, la molécule est inactive, et active dans le cas contraire. Cependant on peut raffiner cette discrétisation en définissant par exemple qu'une molécule est complètement représentative des molécules actives si sa valeur d'activité est supérieure à 2, et qu'elle l'est de moins en moins quand son taux d'activité se rapproche de 0. Dans ce cas, par exemple les règles à certitude vont s'intéresser au fait que la molécule est active au sens large, alors que les règles flexibles et les règles graduelles vont décrire la manière dont la molécule est active. Le prédicat "atom(A,B,C,D,E)" indique que la molécule A contient l'atome B de symbole atomique C. Les attributs D et E concernent le type et la charge partielle de l'atome (E est un attribut numérique). Le prédicat "bond(A,B,C,D)" indique que les atomes B et C sont liés dans A par une liaison moléculaire de type D. Les prédicats "ind1(A,B)" et "inda(A,B)" indiquent respectivement si la molécule A est un 3 ou plus anneaux de benzyl et si elle est un acenthryle. Les prédicats "logp(A,B)" et "lumo(A,B)" décrivent respectivement l'hydrophobicité et la plus petit orbite moléculaire non-occupée de la molécule. Dans ces deux prédicats, B est un attribut numérique.

type de règles	nbr. de règles	couverture	moy. cf
règles classiques	6	0.78	0.96
règles classiques sur le noyau	3	0.65	0.90
règles classiques sur le support	4	0.81	0.98
règles flexibles	2	0.76	0.92
règles graduelles	5	0.47	0.95
règles à certitude	3	0.80	0.97

règle classique
$ind1(A,'true'), bond(A,C,B,'2'), inda(A,'false') \rightarrow act(A,$**active**$)$
règle flexible
$lumo(A,$**bas**$), logp(A,$**moyen**$) \rightarrow act(A,$**active**$)$
règle graduelle
$logp(A,$**low**$) + ind1(A,'true') + lumo(A,$**low**$) \rightarrow act(A,$**active**$)$
règle à certitude
$bond(A,B,C,'2'), ind1(A,'1'), atm(A,D,'c',$**moyen**,**bas**$) \rightarrow act(A,$**active**$)$

Comme prévu, le taux de couverture des règles classiques est situé entre celui des règles classiques sur le noyau et celui sur le support des ensembles flous. Cela est dû au fait que, avec la base de données sur le noyau des ensembles flous, les ensembles qui sont à la limite des classes non-floues ne sont pas traités. Au contraire, avec la base de données sur le support des ensembles flous, les ensembles qui sont à la limite des classes non floues peuvent appartenir à deux classes dans cette base. Le score de couverture inférieur des règles floues vient du fait que celles-ci sont plus difficiles à trouver que les règles classiques. En effet les règles floues prennent en compte le degré de satisfaction des valuations de chaque prédicat de la règle. Le degré de confiance des règles classiques n'est pas lié à la distance des données aux limites des ensembles discrétisés. Par exemple, considérons une règle classique F avec un bon degré de confiance et la règle flexible associée F'(qui est syntaxiquement la même). Si la plupart des exemples de F sont à la limite des ensemble non-flous, le degré de confiance de F' va être plus bas que celui de F. Dans le cas contraire, si la plupart des contre-exemples de F sont à la limite des ensembles non-flous, le degré de confiance de F' va être supérieur à celui de F. Ainsi, le degré de confiance de l'équivalent flexible d'une règle classique est un bon indicateur de la robustesse de la règle par rapport à de petites variations des ensembles.

Les règles qui gèrent l'incertitude ont tendance à favoriser les prédicats non-flous car ils laissent plus de liberté pour le degré de satisfaction des exemples couverts. Comme on le voit sur les exemples, l'algorithme peut induire des règles "mixtes" avec des prédicats flous et des prédicats non-flous.

6 Conclusion

Dans ce papier, nous avons introduit un cadre formel et un algorithme pour gérer des prédicats flous et pour apprendre des règles en logique du premier ordre, de différents types, dans le cas des bases de données relationnelles. Dans ce contexte, notre but était dans un premier temps d'apprendre les règles et de leur attribuer un degré de confiance. La manière dont on peut utiliser ses règles en classification et en déduction automatique n'a pas été abordée. Comme le calcul des degrés de confiance est une version pondérée de FOIL, la complexité de l'algorithme est du même ordre que celle de FOIL. Il est clair que l'utilisation de prédicats flous pour gérer des données numériques à la place d'une discrétisation non floue est un bon compromis entre la lisibilité des règles et la flexibilité introduite dans la discrétisation. De plus, l'utilisation de prédicats flous permet d'induire des nouveaux types de relations.

Au vu des expérimentations, on s'aperçoit qu'il existe trop de contraintes sur chaque type de règle floue pour permettre de couvrir tous les exemples du concept cible, mais elles apportent des informations sur la robustesse des règles vis-à-vis des exemples au bord des ensembles discrétisés et sur la façon dont les degrés de satisfaction de la partie condition et de la partie conclusion évoluent simultanément. Il apparait donc intéressant d'apprendre ces règles en même temps que les règles classiques.

La description du degré de confiance des règles contenant des prédicats flous permet d'incorporer les prédicats flous dans tous les algorithmes qui utilisent le degré de confiance pour guider le processus d'apprentissage. On s'aperçoit aussi que chercher des règles qui ne couvrent pas de contre exemples n'a pas vraiment de sens dès lors qu'on utilise des prédicats flous. Il serait donc utile de développer une nouvelle description formelle de la PLI incorporant les prédicats flous.

Références

[1] H. Blockeel, L. De Raedt Lookahead and discretization in ILP. In N. Lavrac and S. Dzeroski, editors,*Proceedings of the 7 th International Conference of Inductive Logic Programming, ILP 1997*, pages 77–92, 1997.

[2] B. Bouchon-Meunier and C. Marsala. Learning fuzzy decision rules, in. *Fuzzy Sets in Approximate Reasoning and Information Systems*,(J.C. Bezdek, D. Dubois, H. Prade, eds.), The Handbooks of Fuzzy Sets Series. Kluwer Academic Publishers, 1999, 279-304.

[3] M. Delgado, D. Sanchez, and M.A. Vila. Fuzzy cardinality based evaluation of quantified sentences. *Inter. J. of Approximate Reasoning*, pages 23 :23–66, 2000.

[4] D. Dubois and H. Prade. What are fuzzy rules and how to use them. *Fuzzy Sets and Systems*, 84(2) :169–189, 1996.

[5] J. Halpern. An analysis of first-order logics of probability. *Artificial Intelligence*, 46 :310–355, 1990.

[6] E. Hüllermeier. Implication-based fuzzy association rules. In L. De Raedt and A. Siebes, editors, *Proc. PKDD-01, 5th Conf. on Principles and Pratice of Knowledge Discovery in Databases*, number 2168 in LNAI, pages 241–252, 2001.

[7] K. Kersting, L. De Raedt Bayesian Logic Programs. In J. Cussens and A. Frish, editors,*Proceedings of the work-in-progress track at the 10 th International Conference of Inductive Logic Programming*, pages 138–155, 2000.

[8] S.H. Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13 :245–286, 1995.

[9] S.H. Muggleton. Learning stochastic logic programs. *Electronic Transactions in Artificial Intelligence 4*, 141-153 2000.

[10] D. Nauck and R. Kruse. Neuro-fuzzy methods in fuzzy rule generation, in. *Fuzzy Sets in Approximate Reasoning and Information Systems*,(J.C. Bezdek, D. Dubois, H. Prade, eds.), The Handbooks of Fuzzy Sets Series. Kluwer Acad. Pub., 1999, 305-334.

[11] S-H Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming*. LNAI 1228. Springer, 1997.

[12] H. Prade, G. Richard, and M. Serrurier. Learning first order fuzzy rules. In *Proc. of 10 th Int. Fuzzy Systems Association (IFSA-03)*, Istanbul, 702-709, 2003.

[13] H. Prade, G. Richard, and M. Serrurier. On the induction of differents kind of first-order fuzzy rules. In *Proc7 th European Conference on Symbolic and Qualitative Approaches to Reasoning with Uncertainty (ECSQARU-03)*, Aalborg, 370-381, 2003.

[14] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1) :81–106, 1986.

[15] J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5 :239–266, 1990.

[16] J. R. Quinlan. Learning with continuous classes. In *Proc. Artificial Intelligence (AI'92)*, 343–348, Singapore, 1992.

[17] D. Shibata, N. Inuzuka, S. Kato, T. Matsui and H. Itoh In Ning Zhong and Lizhu Zhou, editors, *Proceedings of The Third Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-99) : Methodologies for Knowledge Discovery and Data Mining*, 268–273, LNAI 1574, Beijing, China, April 1999.