Available online at www.sciencedirect.com

ScienceDirect

Artificial Intelligence 171 (2007) 939–950

Artificial Intelligence

www.elsevier.com/locate/artint

Introducing possibilistic logic in ILP for dealing with exceptions

Mathieu Serrurier *, Henri Prade

UPS, IRIT 118 route de Narbonne, Toulouse, France

Received 10 August 2005; received in revised form 27 April 2007; accepted 30 April 2007

Available online 22 May 2007

Abstract

In this paper we propose a new formalization of the inductive logic programming (ILP) problem for a better handling of exceptions. It is now encoded in first-order possibilistic logic. This allows us to handle exceptions by means of prioritized rules, thus taking lessons from non-monotonic reasoning. Indeed, in classical first-order logic, the exceptions of the rules that constitute a hypothesis accumulate and classifying an example in two different classes, even if one is the right one, is not correct. The possibilistic formalization provides a sound encoding of non-monotonic reasoning that copes with rules with exceptions and prevents an example to be classified in more than one class. The benefits of our approach with respect to the use of first-order decision lists are pointed out. The possibilistic logic view of ILP problem leads to an optimization problem at the algorithmic level. An algorithm based on simulated annealing that in one turn computes the set of rules together with their priority levels is proposed. The reported experiments show that the algorithm is competitive to standard ILP approaches on benchmark examples.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Inductive logic programming; Non-monotonic reasoning; Possibilistic logic

1. Introduction

Inductive Logic Programming (ILP) [1] provides a general framework for learning classical first-order logic rules. Reasonably efficient algorithms have been developed and several applications have been described in data-mining, especially for biochemistry databases, for natural language processing and more generally, in relational databases, since first-order logic is well-adapted for describing them.

The handling of exceptions is a serious bottleneck in ILP when learning from entailment with the classical separate and conquer methods (Progol [2], LogAn-H [3], ...), due to the formalization in first-order logic. In separate and conquer ILP approaches, the rules are learned one by one. When a rule is induced, all the examples that are correctly classified by it are usually removed (however, some algorithms such as ALEPH [4] can take them into account for the computation of the quality measure). Then, when a rule has some exceptions, i.e. some examples are misclassified by it, there is no way to compensate these exceptions by means of another rule. So, a hypothesis, i.e. a set of clauses, accumulates all the exceptions of the rules that appear in it. Moreover, when dealing with more than two classes, if an example is classified in two different classes (even if one is the right one), it is considered as misclassified since there is no way to prefer one class to the other. Note that this problem still remains for binary classification. This is why

[*] Corresponding author.
E-mail address: serrurie@irit.fr (M. Serrurier).

the most common method in ILP is to describe examples in distinguishing between positive and negative ones. Thus, only one class is learned, the second one being used for default classification. So, in this way, an example cannot be classified into two classes. However, this approach does not allow us to use information on negative examples for inducing simple rules for discriminating them, if such rules exist.

In order to overcome the misclassification problem (at least partially), the ILP algorithm looks for rules with high confidence levels, i.e. rules with no exceptions if possible, or if not, rules with very few exceptions. Finding such hypotheses, if there exist, may be very costly and tricky. Moreover, looking for a hypothesis without exception w.r.t. the training database is not very realistic, since the exception-handling problem will remain with the arrival of new examples. Besides, some rules with exceptions may be really interesting if they are very simple and cover a large number of examples (i.e. with a large support). This kind of rule expresses some default knowledge (which may be useful to get).

Thus, the problem of handling exceptions in ILP appears to be closely related to the default rule reasoning problem. It is known that reasoning with a default rules in a classical logic setting leads to inconsistency in presence of exceptions. It is why methods have been developed in non-monotonic reasoning for processing default rules in a consistent manner. This suggests that a proper handling of exceptions in ILP amounts to equipping ILP with default inference capabilities. This can be made first by adding a logical constraint expressing that an example can be only classified into one class in the standard ILP setting, in order that any exception leads to logical inconsistency. Second, it has been shown that default reasoning can be encoded in possibilistic logic [5]. Indeed, possibility logic [6] handles priority levels associated with classical logic formulas and preserves safe conclusions from consistent subsets of formulas with a sufficiently high level. This leads in this paper to reformalize the ILP problem into a possibilistic first-order logic setting. The basic idea is sketched in the following elementary example that gathers data concerning three objects identified by x_1, x_2, x_3 together with the constraint expressing that an object cannot be classified into two different classes and a hypothesis made of three rules equipped with priorities α, β, γ. Class assignment is stated by the predicate $C(x, y)$, which in the example below represents the flight capability, where y denotes the class of the object x ($y = 1$ if x flies, $y = 0$ if not). The level 1 represents the highest priority level.

$$K = \begin{cases} (C(X,Y) \wedge C(X,Z) \to Y = Z, 1), \\ (\mathit{SuperPenguin}(x) \to C(X,1), \alpha), \\ (\mathit{Penguin}(X) \to C(X,0), \beta), \\ (\to C(X,1), \gamma), \\ (\mathit{SuperPenguin}(x_1), 1), (\mathit{Penguin}(x_1), 1), (\mathit{Penguin}(x_2), 1), (\mathit{Bird}(x_3), 1). \end{cases}$$

Without the levels, the above set of formulas K is inconsistent because the objects x_1 and x_2 are classified as flying and not flying in the same time. However, assuming that $1 > \alpha > \beta > \gamma$ holds, each example can then be classified into one class, without inconsistency. Indeed, possibilistic logic allows the safe deduction of formulas $(C(x_1, 1), \alpha)$, $(C(x_2, 0), \beta)$, $(C(x_3, 1), \gamma)$ from K, as will be explained in Section 2.2 (see especially Definition 4). Thus the ILP problem amounts now not only to finding a set of rules but also their associated priority levels. This idea is reminiscent of the use of first-order decision list [7–9], however decision lists handle orders in a heuristic way that does not necessarily follow non-monotonic requirements and does not handle recursivity properly, as discussed in Section 4. Then, the possibilistic frameworks allows us to describe a general ILP framework which extends decision list and takes advantage of default reasoning in order to overcome the problem of handling exceptions in ILP. Moreover, the framework is easily expendable to the handling of qualitative uncertainty. Non-monotonic reasoning has already influenced research in ILP [10–12], but it is in an incremental learning perspective. Then, the closed world assumption is used for computing a minimal specialization of a rule, in order to cope with known exceptions of the rule. In a nutshell, this type of approach induces rules such as “birds that are not penguins or that are superpenguins fly”, while the proposed approach will rather induce “birds generally fly”, “penguins does not fly”, and “superpenguins fly” together with priority levels to escape inconsistency as viewed in the example. Other non-monotonic approaches have been proposed in ILP [13–15] and are discussed in Section 4.

The paper is organized as follows. First, we present a formalization of the ILP setting that explicitly introduces the constraint on the uniqueness of the class for an example. Next we propose an extension of possibilistic propositional logic in the first-order case. In this setting, we reformulate the ILP problem and we establish some properties of this framework. We extend a recent ILP algorithm based on simulated annealing for dealing with possibilistic ILP, which gives birth to an optimization problem. Then related works are discussed, especially the differences with respect to first-order decision list. Lastly, we report some experiments on classical benchmark databases. This paper is an expanded and revised version of [16,17].

2. Preliminaries

2.1. Inductive logic programming and classification

We first briefly recall the standard definitions and notations used. Given a first order language $\mathcal{L}$ with a set of variables *Var*, we build the set of terms *Term*, atoms *Atom* and formulas as usual. The set of ground terms is the Herbrand universe $\mathcal{H}$ and the set of ground atoms or facts is the Herbrand base $\mathcal{B} \subset Atom$. A *literal* l is just an atom a (positive literal) or its negation $\neg a$ (negative literal). A (resp. ground) substitution σ is an application from *Var* to (resp. $\mathcal{H}$) *Term* with inductive extension to *Atom*. We denote $Subst$ the set of ground substitutions. A *clause* is a finite disjunction of literals $l_1 \vee \cdots \vee l_n$ also denoted $\{l_1, \ldots, l_n\}$. A Horn clause is a clause with at most one positive literal. A Herbrand interpretation I is just a subset of $\mathcal{B}$: I is the set of true ground atomic formulas and its complement denotes the set of false ground atomic formulas. Let us denote $S = 2^{\mathcal{B}}$, the power set of $\mathcal{B}$ i.e. the set of all Herbrand interpretations. We can now proceed with the notion of a logical consequence. Given A an atomic formula, $I, \sigma \models A$ means that $\sigma(A) \in I$. As usual, the extension to general formulas F uses compositionality. $I \models F$ means $\forall \sigma$, $I, \sigma \models F$ (we say I is a model of F). $\models F$ means $\forall I \in S$, $I \models F$. $F \models G$ means all models of F are models of G (F entails G).

Stated in the general context of first-order logic, the task of ILP, when learning from entailment, is to find a non-trivial set of formulas H such that:

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

given a background theory B and a set of examples E (training set). E denotes a set of grounds facts. B and H denote sets of Horn clauses. An example is said covered by the hypothesis H if it is deducible from $B \cup H$. When dealing with two class problems, it is common to introduce negative examples in order to describe some exceptions of a rule. A rule in H has an exception if it covers a negative example. Moreover, it allows algorithms to learn only one class, the second class being deduced by default. According to the remarks made in the introduction, we reformulate the ILP problem, by adding a classification constraint D, as follows in the spirit of [18]:

- E is a set of ground facts of the form $C(x, y)$ where x denotes the identification key of an example and y a class.
- B is a set of rules that describes background knowledge
- D is the constraint:

  $$\forall X, Y, Z \quad C(X, Y) \wedge C(X, Z) \rightarrow Y = Z \tag{2}$$

  which expresses that an example can be only classified into one class.

Then, given B, D and E, the goal is to find H such as

$$B \cup D \cup H \models E. \tag{3}$$

Of course, we must have $B \cup D \cup H \not\models \perp$.

Definition 1. $e \in E$ is an exception for the rule $h \in H$ if $B \cup D \cup h \cup e \models \perp$. $e \in E$ is an exception for the hypothesis H if $B \cup D \cup H \cup e \models \perp$.

With this definition, an exception to a rule or a hypothesis is a misclassification. Thus, if e is an exception for $h \in H$, then e is an exception for H and this leads to the problem of handling exceptions described in the introduction. A hypothesis is said to be *complete* if it covers all the examples (i.e. $B \cup H \models E$). The hypothesis is correct if and only if $B \cup D \cup H \cup E \not\models \perp$. There are two ways for not being correct with respect to B, D and E. First, H may classify an example in two different classes and then we have $B \cup D \cup H \models \perp$. Second, H may misclass an example $e \in E$ i.e. $B \cup D \cup H \cup e \models \perp$. Thanks to the use of D, there is no need to introduce negative examples for having a notion of exception and then the ILP problem can be completely stated in logical terms. The accuracy of a hypothesis, which is the proportion of correctly and uniquely classified examples is computed as follows:

$$acc(H) = \frac{|\{C(x,y) \in E \text{ s.t. } B \cup H \models C(x,y) \text{ and } \nexists C(x,z), z \neq y, B \cup H \models C(x,z)\}|}{|E|}$$

where $|\,|$ denotes the cardinality of the considered set.

2.2. First-order possibilistic logic

We extend propositional possibilistic logic [6] to the first-order case. Possibilistic logic is sound and complete for refutation, using an extended resolution rule, with respect to a semantics in terms of a complete plausibility preorder (a reflexive and transitive binary relation) on the interpretations (encoded by a possibility distribution) [6]. This resolution rule associates its result with the minimum of the levels of the formulas involved. S denotes a set of Herbrand interpretations. A possibility distribution can be defined on Herbrand interpretations as well. A possibility distribution π on S is a function from S to $[0,1]$. π is said to be normalized iff $\exists I \in S$ such that $\pi(I) = 1$. A possibility distribution, assumed here to be normalized, encodes a complete pre-order on interpretations. From π, a possibility measure is defined for first-order formulas ϕ by $\Pi(\phi) = \max\{\pi(I), I \in S, I \models \phi\}$. Necessity is the dual notion of possibility. It refers to the possibility degrees of the counter-models of a formula and is defined by $N(\phi) = 1 - \Pi(\neg\phi)$. We now define a notion of possibilistic entailment $\models_\pi$. In order to do that, we need three auxiliary definitions.

Definition 2. A possibilistic first-order formula is a pair (ϕ, α) where ϕ is a first-order formula and $\alpha \in]0,1]$.

It is understood as a constraint on an unknown necessity measure of the form $N(\phi) \geqslant \alpha$. Given a set K of possibilistic formulas, we define the α-cut of K as:

Definition 3. The α-cut of K is $K_\alpha = \{\phi \mid (\phi, \beta) \in K, \beta \geqslant \alpha\}$.

In standard propositional possibilistic logic, a formula (ϕ, α) is a logical consequence of a possibilistic set of formulas K if and only if there exists β such that $\beta \leqslant \alpha$, $K_\beta \not\models \bot$ and $K_\beta \models \phi$. Nevertheless, it has been pointed out [5] that this definition encounters some drowning effect, i.e. consequences are lost when their implicants are taken in inconsistent sets of formulas (due to their low priority levels), although they do not contribute to inconsistency. This becomes very critical in the first-order case. It leads to extending the notion of possibilistic models for first-order inference in the following way.

Definition 4. Given T a set of classical first-order formulas, T is minimal w.r.t. a formula ϕ iff $T \models \phi$ and $\forall \psi \in T$, $T - \psi \not\models \phi$.

Definition 5. Given K a set of possibilistic first-order formulas, $K \models_\pi (\phi, \alpha)$ iff $\exists K' \subseteq K_\alpha$, $K' \not\models \bot$ such that K' minimal w.r.t. ϕ and $\nexists K''$ minimal w.r.t. $\bot$ such that $K' \subset K'' \subseteq K_\alpha$.

Definition 5 avoids the drowning effect by checking if there exists a proof of ϕ in K_α, which is free from inconsistencies at level α. Let us apply the definition on the example in the introduction. Without the weights, K is inconsistent since it is possible, for instance, to deduce $C(x_1, 0)$ and $C(x_1, 1)$ in the same time and this is contradiction with the clause $C(X,Y) \wedge C(X,Z) \to Y = Z$. If we consider the classical propositional possibilistic framework, the only safe deduction is $(C(x_1,1), \alpha)$ since K_β is inconsistent. By using Definition 5, we can also deduce $(C(x_2,0), \beta)$ and $(C(x_3,1), \gamma)$. Let us for instance take the case of $(C(x_3,1), \gamma)$. The possibilistic formula $(C(x_3,1), \gamma) \in K_\gamma$ is minimal with respect to $(C(x_3,1), \gamma)$ according to Definition 4 and cannot contains subsets of clauses that are minimal with respect to $\bot$, so $K \models_\pi (C(x_3,1), \gamma)$.

In practice, this definition makes sure that, if two logical consequences are inconsistent, the one which has the highest necessity degree is preferred. If the two logical consequences have the same necessity level, neither is deduced. As already advocated in the introduction, we propose to use possibilistic logic in ILP for a better handling of exceptions.

3. Possibilistic ILP

3.1. Definitions and propositions

Given H a standard ILP hypothesis, let N_H denote a function (called priority function), to each rule in H, associates a priority level in $]0,1]$, to be understood as a necessity degree. This gives birth to a possibilistic hypothesis $H_p = \{(h, N_H(h)); h \in H\}$. Without any information, we assume that B_p and D_p are composed of all formulas that appear respectively in B and D, with 1 as necessity level. But, if the information is available, B_p may contain formulas with levels different from 1 in order to express qualitative uncertainty or to encode default reasoning behaviors. Then, given D, B and E, the goal of possibilistic ILP is to find H_p, composed by a classical hypothesis H and an associated priority function N_H s.t.:

$$\exists N_E \quad \text{s.t.} \quad B_p \cup D_p \cup H_p \models_\pi E_p \tag{4}$$

where N_E is a priority function over E and $E_p = \{(e, N_E(e)); e \in E\}$. Note that, if there exists such H_p, this hypothesis will be correct and complete (i.e. it classifies all examples and make no misclassification). Given H_p and N_H a solution of Eq. (4), the maximal priority function N_E is

$$\forall e \in E, \quad N_E(e) = \max\{\alpha > 0, D_p \cup B_p \cup H_p \models_\pi (e, \alpha)\}. \tag{5}$$

It corresponds to the necessity level of the clause in H_p that allows the coverage of the example. However, N_E, which stratifies the set of examples, has no practical use in the following, but is just a byproduct of the possibilistic inference that propagates levels. The corresponding definition of an exception for possibilistic ILP is the following.

Definition 6. $e = C(x, y) \in E$ is an exception for the possibilistic rule $(h, N_H(h)) \in H_p$ if $\exists \alpha > 0$, $z \neq y$, $B_p \cup D_p \cup (h, N_H(h)) \models_\pi (C(x, z), \alpha)$. $e = C(x, y) \in E$ is an exception for the possibilistic hypothesis H_p if $\exists \alpha > 0$, $z \neq y$, $B_p \cup D_p \cup H_p \models_\pi (C(x, z), \alpha)$.

By contrast with classical settings, if e is an exception for $(h, N_H(h)) \in H_p$, e is not necessarily exception for H_p. This enlarges the scope of classical ILP by learning sets of prioritized rules in the possibilistic logic framework. This is now illustrated.

Example 1. We consider the following background B and set of examples E which describes capabilities of flying split into three classes: 2 for flight capabilities, 1 for only gliding capabilities and 0 otherwise:

$$B = \begin{cases} \mathit{Fish}(x_1), \mathit{Wings}(x_1), \mathit{Penguin}(x_2), \mathit{Fish}(x_3), \\ \mathit{Wings}(x_4), \end{cases}$$

$$E = \begin{cases} C(x_1, 1), C(x_2, 0), C(x_3, 0), \\ C(x_4, 2). \end{cases}$$

We induce a prioritized hypothesis H_p. Then, we compute the set E_p associated with the maximal priority function N_E according to Eq. (5):

$$H_p = \begin{cases} (\mathit{Fish}(X) \wedge \mathit{Wings}(X) \to C(X, 1), 0.9), \\ (\mathit{Penguin}(X) \to C(X, 0), 0.7), \\ (\mathit{Fish}(X) \to C(X, 0), 0.4), \\ (\to C(X, 2), 0.3), \end{cases}$$

$$E_p = \begin{cases} (C(x_1, 1), 0.9), \\ (C(x_2, 0), 0.7), (C(x_3, 0), 0.4), \\ (C(x_4, 2), 0.3). \end{cases}$$

H_p, as proposed above, is an acceptable solution for Eq. (4) and is correct and complete since it covers all the examples without making any error. We have, for instance, $D_p \cup B_p \cup H_p \models_\pi (C(x_1, 1), 0.9)$ because, according to Definition 5, $(C(x_1, 1), 0.9)$ is minimally entailed by $K = \{\mathit{Fish}(X) \wedge \mathit{Wings}(X) \to C(X, 1), \mathit{Fish}(x_1), \mathit{Wings}(x_1)\} \subset (D_p \cup B_p \cup H_p)_{0.9}$, and is not contained in a minimal inconsistent subset of $(D_p \cup B_p \cup H_p)_{0.9}$. On the contrary $D_p \cup B_p \cup H_p \not\models_\pi (C(x_1, 0), 0.4)$ since $(C(x_1, 0), 0.4)$ is minimally entailed by $K = \{\mathit{Fish}(X) \to C(X, 0), \mathit{Fish}(x_1)\} \subset$

$(D_p \cup B_p \cup H_p)_{0.4}$, but it is contained in $\{Fish(X) \wedge Wings(X) \rightarrow C(X,1), Fish(x_1), Wings(x_1), Fish(X) \rightarrow C(X,0)\}$, which is minimal inconsistent.

Then, having N_H injective contributes to enforcing that an example be classified into only one class. Note that possibilistic ILP allows us to make use of a default class. Indeed, a rule with the lowest necessity level will put any example that remains unclassified by the other rules into this default class (this is the case of the rule $(\rightarrow C(X,2), 0.3)$ in the example). The classical hypothesis H, which corresponds to H_p without necessity level, is not correct in standard ILP because it classifies all the examples, except x_4, in at least two different classes. For instance, $Fish(X) \wedge Wings(X) \rightarrow C(X,1)$ states that x_1 has only the ability to glide, $Fish(X) \rightarrow C(X,0)$ states that it has no flight ability and $\rightarrow C(X,2)$ states that it flies. Without any priority information over rules, H cannot assign a class for x_1.

Note that a possibilistic hypothesis H_p such that $\nexists (h,1) \in H_p$ can always be completed (if needed) in order to satisfy Eq. (4) by adding the mis- or unclassified examples with 1 as necessity degree in the hypothesis. This completion is not always possible in classical ILP. Obviously, this completion is not interesting from a machine learning point of view. In order to have a more adequate description of the possibilistic ILP problem we reformulate it as an optimization problem:

Given D, B and E, the goal of possibilistic ILP is to find a hypothesis H with $H \cap E = \{\emptyset\}$, associated with its priority function N_H, such that H_p maximizes the accuracy. We compute the accuracy (which have the same meaning than the classical one) for a possibilistic hypothesis H_p as follows:

$$acc_p(H_p) = \frac{|\{e \in E \mid \exists \alpha > 0, D_p \cup B_p \cup H_p \models_\pi (e,\alpha)\}|}{|E|}.$$

This definition of possibilistic ILP problem is fully in the spirit of minimizing empirical risk.

Proposition 1. *Given a hypothesis H, for any N_H, $acc(H) \leqslant acc_p(H_p)$ where $H_p = \{(h, N_H(h)); h \in H\}$.*

This proposition shows that, when dealing with multiple classifications, associating a priority level to each rule in a hypothesis is never any worse than not doing it. This is because ordering rules by associating necessity degrees to them may in the best case prevent an example from being classified into two different classes, while it is innocuous in the worst case. Since, in the standard ILP framework, such a kind of examples are considered as misclassified, favoring the class that has the highest necessity degree could only increase the accuracy. Of course, if only one class is learned, ordering rules does not make any difference.

When necessity levels are only used for obtaining an ordering of the formulas in H, we can describe equivalence classes of priority functions N_H.

Definition 7. Given H, and two priority functions N_H and N'_H, they belong to same equivalence class i.e. $N_H \equiv N'_H$ if and only if $\forall h1, h2 \in H$ if $N_H(h1) > N_H(h2)$ then $N'_H(h1) > N'_H(h2)$. $\overline{N_H}$ denotes the class of equivalence of N_H.

The class of equivalence of priority functions N_H is the set of all priority functions that induce the same ordering of the formulas in H. This notion corresponds to the idea of well-ordered partitions in possibility theory [19]. Thus we have:

$$\forall N'_H, N''_H \in \overline{N_H}, \quad acc_p(\{(h, N'_H(h)) \mid h \in H\}) = acc_p(\{(h, N''_H(h)) \mid h \in H\}).$$

Proposition 2. *Given H, finding the class of equivalence of priority functions such that $acc_p(H_p)$ is maximized is NP-complete with respect to the number of formulas in H.*

In fact, for finding the equivalence class $\overline{N_H}$ that maximizes $acc_p(H_p)$, all possible $\overline{N_H}$ need to be tested. It corresponds to testing all possible orders on the formulas in H. It shows that, although using a possibilistic rather than the classical setting is always more effective, finding the best priority function over formulas in H may be computationally very costly. Moreover, choosing a particular ordering based on the confidence or the support degrees of the rules is not optimal in general. It suggests using some heuristics for inducing hypotheses and finding the priority levels of rules in them.

3.2. Algorithm

In order to take full advantage of possibilistic ILP, the hypothesis and its associated priority function must be learned jointly for at least two reasons. First, using a standard ILP algorithm which learns rules without exception (or with very few exceptions), and then looking for a priority function on them cannot lead to a great improvement since the approach is especially beneficial for rules with exceptions. Second, an algorithm that would learn rules one by one may miss a global hypothesis which as a whole may reach a better accuracy. This agrees with the NP-complete nature of the problem. In this paper we extend the simulated annealing method for inducing hypotheses directly, which is described in [20] and denoted by SAHILP. The SAHILP algorithm performs a stochastic search, guided by an empirical temperature parameter T, in the hypothesis space in order to maximize a quality measure F on classical hypotheses. In order to learn a possibilistic hypothesis, we associate to each randomly chosen hypothesis H a priority function N_H and we adapt the quality measure. The general form of the algorithm, denoted as PossILP is the following, where F_p is a quality measure for possibilistic hypotheses:

```
PossILP($T$, $H_p$)
1.  while ($T > T_{\min}$)
        1.1. choose randomly $H'$ in the neighborhood of $H$
        1.2. compute $N_{H'}$
        1.3. if ($F_p(H'_p) > F_p(H_p)$) then $H_p = H'_p$
        1.4. else $H_p = H'_p$ with a probability equal to $e^{-\frac{F_p(H'_p)-F_p(H_p)}{T}}$
        1.5. decrease T
2.  return $H_p$
```

The neighborhood of a hypothesis is described from a refinement operator on clauses. We use the FOIL refinement operator. With this operator, the neighborhood of a hypothesis is obtained by adding or removing a rule in the hypothesis or adding or removing a literal in a rule of the hypothesis (see [20] for details). Using a neighborhood operator on a hypothesis allows us to induce a hypothesis directly and then to take full advantage of the possibilistic settings. The different parameters used in simulated annealing are automatically computed with respect to the database used in order to avoid the bias associated with the tuning of machine learning parameters. In this algorithm, we only deal with a classical background where all the formulas that appear in B have a level equal to 1. In this case, the weights associated with rules in the hypothesis only encode order. Thus, the levels used for the priority function are of the form $\frac{i}{|H|}$ with $i \in N^*, i \leqslant |H|$. By using a different level for each rule, we can describe all the possible injective priority functions modulo the equivalence relation. According to Proposition 3, finding the optimal priority functions with respect to the accuracy is not realistic, so we use a heuristic based on the 2-opt algorithm [21]. The 2-opt method is a greedy algorithm initially proposed for the traveler salesman problem that works as follows. An initial priority function is built randomly. Then, while accuracy increases, the priority values of the rules are swapped two by two. All combinations are tested until no increase of accuracy can be obtained. The function obtained is not necessarily optimal but it allows the algorithm to find quickly an acceptable solution. By this way, the function $N_{H'}$ is computed at the step 1.2 independently of N_H. One other solution is to allow the neighborhood operator to change the value of the necessity level. Since it greatly increases the size of the state space, this method is less efficient in terms of computation time and performs as well as the 2-opt approach. Even this method is less efficient in terms of computation time when the background remains classical, it could be useful if the background is weighted. This is not explored in the paper, but in this case, since the weight cannot be fixed as previously, considering the possibility to change the levels of the rule in the neighborhood will allow the algorithm to handle together constraints due to the exceptions of the rules and weights in the backgrounds. As well, more complex methods such as 3-opt or exact approach have been investigated, but they only increase computation time without significant improvement of the effectiveness. This is due to the fact that, since the number of clauses is low (less than 50), results found by 2-opt is very close to the maximal one and 2-opt is the most efficient algorithm tested. The quality measure (in the spirit of [22]) that the algorithm should maximize is a minimum description length measure of the form:

$$F_p(H_p) = \frac{pos(H_p) + c * size(H_p)}{|E|}$$

where $pos(H_p) = |\{e \in E \mid \exists \alpha > 0, D_p \cup B_p \cup H_p \models_\pi (e, \alpha)\}|$ is the number of examples well classified by H_p. $Size(H_p) = \sum h_i \in H_p SizeClause(h_i)$ and $SizeClause$ is computed as follows: the clause counts 1, each predicate symbol counts 2, each function symbol and list counts 2, each variable counts 1 and each constant counts 2. The constant c is used for the control of the relative importance given to the accuracy vs. the complexity of the hypothesis. We consider that the most general hypothesis H_g (i.e. the hypothesis that classes all examples in the majority class) is as acceptable as the most specific hypothesis H_s (i.e. E). Thus

$$c = \frac{pos(H_g) - pos(H_s)}{Size(H_s) - Size(H_g)}.$$

The measure F_p will favor hypotheses with high accuracy and a low complexity. Note that given H, finding a priority function N_H that maximizes $F_p(H_p)$ is equivalent to find the distribution that maximizes $acc_p(H_p)$. PossILP is a non-deterministic algorithm. It has only one parameter that corresponds to the computation time allocated to the algorithm. The more the algorithm takes time, the better the maximum of F_p.

4. Related works

Several methods [23–25] delay the problem of disambiguising multiple classifications after the learning process, by using a majority class procedure or other voting strategies, but these methods do not manage exceptions directly inside the learning process. Some works have been made for combining non-monotonic logic and ILP. Bain et Muggleton [10] is based on the non-monotonic setting of closed world assumption, i.e. a fact that is not derived from the background or the examples is assumed to be false. In this case, only positive examples are considered, and hypotheses induced is the one that satisfies the least Herbrand model of B and E. Other ILP approaches use non-monotonic settings for extending the language capability of ILP, by for instance allowing negation as failure by means of a three valued logic [13,14], or by including default reasoning definitions in the background [15]. Three valued approaches can handle exceptions and multiple classifications, but they are not adapted when more than two classes have to be learned. Moreover, since two hypotheses are learned separately, one of the positive examples and one for the negative ones, the coherence among the hypotheses is not guaranteed as is in our approach.

Besides, the formalization of possibilistic ILP may at the first glance look similar to probabilistic ILP ones (see [26] for a review), since numbers between 0 and 1 are used for encoding priorities. But the purpose of probabilistic ILP, which handles weights in a different manner, is to describe randomness and quantitative uncertainty. The possibilistic approach does not deal here with uncertainty (although possibilistic logic may be used for encoding qualitative uncertainty), but deals with preference about using some pieces of knowledge rather than others in order to encode non-monotonic reasoning. In this context, possibility values are qualitative and are only used for describing ordinal preferences, contrarily to probabilities which are quantitative and describe statistical knowledge. Only very special types of probabilities, named big-stepped probabilities [19] (see also [27]), can encode default reasoning in a proper way, which also agrees with the possibilistic logic treatment of exceptions. Possibilistic ILP may also seem to be related to the graded classification approach proposed by [28], however, the intended concern are quite different since these authors want to introduce tolerance in the learning by a better handling of the discretization of the numerical attributes. A first approach of possibilistic ILP has been proposed in [29]. In this other framework, possibility degrees associated with formulas in the background and the examples are required and are supposed to be given by an external source. In the approach reported here, there is not need of degrees in the input of the procedure (although it would be feasible in principle to consider degrees in the background).

Due to the use of priority functions in possibilistic ILP, our approach can be compared with induction of decision lists [7], and decision trees [30] which are a particular case of decision lists. Some applications in planning [8] or in computational linguistics [9] have required the introduction of first-order decision list. In this case, a hypothesis is a set of rank-ordered rules ended by a cut (which stops the firing of a list of rules at the first classification success). This approach does not handle inconsistency, but rather uses a clever logic programming trick, the cut, for avoiding the multiple classification of an example. Then, possibilistic inductive logic programming can be used for simulating the behavior of a decision list. In fact, if N_H is injective and B and H contain no recursive rules, possibilistic ILP and decision list ILP (with rules ordered in the same way as the one described by N_H), will have the same behavior. However, possibilistic ILP handles exceptions and inconsistency inside a unified logical framework. Possibilistic logic can be also used for introducing default rules in the background and N_H can encode more than a simple ordering on rules. Moreover, if the hypothesis contains recursive rules, the behaviors of possibilistic logic and decision list may be different. For instance, let us consider the following set K of possibilistic formulas

$$K = \begin{cases} (\mathit{SameSpecies}(X,Y) \wedge C(Y,0) \rightarrow C(X,0), 0.9), \\ (\mathit{Fish}(X) \wedge \mathit{Wings}(X) \rightarrow C(X,1), 0.5), \\ (\mathit{Fish}(X) \rightarrow C(X,0), 0.1), \\ (\mathit{Fish}(x_1), 1), (\mathit{Wings}(x_1), 1), (\mathit{Fish}(x_2), 1), (\mathit{SameSpecies}(x_1, x_2), 1) \end{cases}$$

and the first three rules viewed as a decision list (i.e. rules are ordered according to the decreasing order of the necessity levels and ended by a cut). Using decision lists, we obtain that example x_1 is in the class 0 by application of the first and third rules. With possibilistic logic, if we add the condition D the example x_1 is put in the class 1 and not in the class 0. Indeed, according to definition 5 $D \cup K \models_\pi (C(x_1, 1), 0.5)$ and $D \cup K \not\models_\pi (C(x_1, 0), 0.1)$. This can be seen because, although $(C(x_1, 1), 0.1)$ can be minimally deduced from a subset of $D \cup K_{0.1}$ (the first and the third rules together with the last two facts), this potential consequence is annihilated because this subset of clauses in included in $D \cup K_{0.1}$ which is minimally inconsistent. It illustrates the fact that first-order decision lists are not designed to be chained, and if a recursive rule is on the top of the list, the class induced by this rule will be preferred, regardless the level of the other rules that are needed for the deduction. On the contrary, when dealing with possibilistic logic, the level of all rules used is taken into account.

Lastly, the algorithms that are described in [8,9] are really different from the one described in this paper. Indeed, in [8,9] rules in the decision list are learned in order, one by one. When rules are learned one by one, the order on rules is fixed and not modified, and the interaction between rules is not fully exploited. In this paper, we learn directly the hypothesis with its priority function and then we can fully take advantage of having sets of prioritized rules. If we restrict learning to function free non-recursive rules, the framework of possibilistic ILP is equivalent to the decision lists. However, possibilistic ILP is more general, and moreover can be easily extended in order to handle qualitative uncertainty in the background knowledge.

5. Experiments

In this section, we present some experiments with the algorithm described in Section 3.2. Since PossILP is a non-deterministic algorithm, the results that are presented are average results on 50 running steps with the same settings. The values in the brackets are averages of standard deviations for the 10-cross validations, the $\pm$ symbol denotes the standard deviation about a results for the 50 tests. The symbol $*$ (resp. $\circ$) indicates that the result marked by this symbol is significantly worst (resp. better) than the one provided by PossILP (paired two-sided t-test, $p = 0.05$; when the variance of the other algorithm is unknown, the same variance as PossILP is used). SAHILP denotes the non-possibilistic version of the PossILP algorithm. Results for SAHILP are also average for 50 tests. Algorithms are implemented in JAVA, and have run on a 3 GHz PC. In order to check the effectiveness of the algorithm, we first used two classical propositional benchmarks from UCI.[1] The Post-Op database describes 3 classes with 90 examples. The Yeast database contains 1484 examples that describe 8 classes. We compare our results with the Support Vector Machine algorithm (with Gaussian kernel) and C4.5 algorithm (with post-pruning) in order to compare our algorithm with decision trees frameworks, which can be viewed as ordered sets of rules, and kernel approaches using pairwise learning scheme. The results presented in Table 1 are for 10-cross validation. All experiments where performed using the same folds.

The results of PossILP are at least as good as the SVM ones. It shows the effectiveness of our approach on propositional databases with respect to other methods for dealing with multiple classes.

Table 1

Alg.	C4.5	SVM	SAHILP	PossILP
Post-Op	*62.2	67.8	70.3 ± 0.1 [3.5]	71.1 ± 0.1 [3.1]
Yeast	*45.8	57.8	*43.6 ± 0.4 [4.1]	57.7 ± 0.2 [0.8]

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

Table 2

Alg.	Progol	SAHILP	SAHILP-biclass	PossILP
acc	86 [6]	89.5 ± 0.4 [5]	*84.1 ± 0.4 [5]	90.1 ± 0.4 [5]

Table 3

Alg.	ICL	TIDLE	RIBL	SAHILP	PossILP
red	*63.5	81.6	∘86.5	*61.6 ± 0.1 [6.2]	79.1 ± 0.1 [2.8]
prop	*79.1	*78.5	*79.0	*74.2 ± 0.1 [2.8]	81.3 ± 0.1 [2.2]
prop + red	*86.0	90.4	91.2	*84.1 ± 0.1 [4.9]	90.8 ± 0.1 [3.3]

We now illustrate the use of PossILP with the database "Mutagenesis"[2] which is a classical benchmark for ILP. This database describes molecules. The concept we want to learn is the activity level of a molecule. The database contains 188 instances described with 7 predicates. The two classes to be learned are "active" and "inactive" molecules. We compare PossILP with SAHILP and Progol. SAHILP and Progol learn only the "active" class, the "inactive" class correspond to negative examples. PossILP learns "active" and "inactive" together. The results presented in Table 2 correspond to a 10-cross validations with the background B2 as explained in [31]. The cross validation schema is the same as the one used in [31].

When applying separate and conquer methods, as PROGOL, and to some extent, SAHILP do, learning two classes together is harder than learning only one class due to the possibility for the algorithm to class an example in two different classes. Clearly, despite this fact, PossILP outperforms SAHILP and Progol on this dataset. SAHILP-biclass is the application of SAHILP to the learning of the two classes together. The comparison between SAHILP-biclass and PossILP shows the advantages of using the possibilistic setting (which leads to good results in multiclass problems). It shows also that the effectiveness of the algorithm is not essentially due to the use of a stochastic search. An overview of the results on mutagenesis databases can be found in [32]. It shows that our algorithm compete with the best ILP methods. The best result (95.8 [3]) is achieved by ensemble ILP method based on random seed that learns a set of hypotheses and uses a voting scheme for classification. There are some reasons for the effectiveness of the algorithm on biclass problems. First, a default classification, associated with the lowest level, can be described in possibilistic logic, as in Example 1. Moreover, it uses all the information about "active" and "inactive" molecules. Lastly, the exceptions of some rules can be captured by adding a more specific rule with a higher priority level.

We next apply our method on harder ILP multiclass problem. The diterpene database [33] describes 1503 separate in 23 classes. Three different backgrounds are available: one with only the *red* predicate, one with only *prop* predicate and the last one that uses both *prop* and *red*. The results are shown in Table 3 are for 10-cross validations (see [33] for the results of ICL, TIDLE, RIBL).

On this database, the results of the different algorithms are very close. RIBL performs very well since it is an algorithm based on nearest neighbors and it has then no difficulties with multiclass problems and numerical attributes. The results obtained are close to the ones with TIDLE; this can be explained by the fact that TIDLE uses a decision tree learning algorithm that describes approximately the same kind of hypotheses as the ones induced by PossILP. Therefore, PossILP appears to be more simple to implement.

A last experiment is made with the finite element MESH Design dataset because it represents a typical hard ILP multiclass problem. It describes the structure of a mesh with unary and binary predicates. The dataset contains 277 examples that describe 13 classes. The dataset is split in 5 sub datasets denoted by A, B, C, D and E. The test of the accuracy of the algorithm is made by testing on one subset a hypothesis induced from the other sub datasets. The results are shown in Table 4 presents the number of correctly and uniquely classified examples detailed for each sub datasets.

The results for the algorithms other than PossILP can be found in [34]. The results are clearly in the favor of PossILP algorithm which increases the number of examples covered by the most effective algorithm up to 19% in average, and of 33% the classic version of SAHILP which perform the same type of hypothesis search than PossILP but without the possibilistic settings. In average, PossILP has the best result for three of the five sub datasets. For the

[2] http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/mutagenesis.html.

Table 4

	A	B	C	D	E	$\sum$
Maximum	52	38	28	57	89	277
Golem	17	9	5	11	10	*52
SAHILP	20.7 ± 2.0	14.3 ± 1.9	6 ± 0.5	13.5 ± 0.4	35.7 ± 1.2	*92.2 ± 4.6
Indigo	21	14	9	18	33	*95
Cilgg	26	12	10	18	37	*103
PossILP	23.3 ± 1.4	12.3 ± 1.7	10.1 ± 0.2	23 ± 0	55.3 ± 0.8	123.1 ± 2.7

Table 5

	postop	yeast	mutagenesis	red	prop	prop + red	mesh
SAHILP	4 s	950 s	470 s	1350 s	1030 s	1850 s	158 s
PossILP	7 s	2230 s	820 s	3540	2830 s	6520 s	335 s

other, results are very close to the best ones. PossILP is only overcome in the sub-dataset *A* by Cilgg which uses an extended language that has the increased expressive power of description logic.

The low standard deviations observed for the 50 tests show that PossILP converges. These results compete with the best ILP algorithms. Moreover, the better performance of PossILP with respect to SAHILP shows that the effectiveness of the algorithm is not only due to the use of metaheuristics for guiding the learning process. Results with *Diterpen* + *red* and *Yeast* databases show that the algorithm has more difficulties with purely numerical datasets, but results on *Mesh* shows that is very efficient with a multiclass relational database. Table 5 exhibits the average time for SAHILP and PossILP on each dataset:

The average time of our algorithm is quite high due to the use of simulated annealing but remains acceptable. PossILP is between two or three times slower than SAHILP. This difference is explained by the time consumption of the computation of the priority function. The higher the number of classes, the higher the difference between SAHILP and PossILP.

6. Conclusion

In this paper, we have proposed a new ILP formalization for dealing with exceptions in a multiclass problem, which is rather simple to apply. In order to do that, we have extended possibilistic propositional logic to a first-order setting. Then, by treating exceptions as inconsistency, we have reformulated the ILP problem in first-order possibilistic logic. In this reformulation, the ILP problem is turned in an optimization problem. This formalization allows our algorithm to learn sets of prioritized rules. In this context, it may exist a correct and complete hypothesis which contains rules that have some exceptions. We have shown that the possibilistic ILP is more flexible and more general than first-order decision lists and allows us to correctly cope with recursive hypotheses. Experiments have proved that an implementation of possibilistic ILP may be very effective and can compete with the best machine learning algorithms.

In the future, it may be interesting to exploit more the power of possibilistic logic. In this scope, it can be used for allowing for prioritized knowledge in the background, or for expressing uncertainty on formulas. Necessity levels could also reflect the exceptionality levels of examples if they are available.

Acknowledgements

The authors are grateful to the anonymous reviewers for making useful suggestions that helped to improve the paper.

References

[1] S. Muggleton, L.D. Raedt, Inductive logic programming: Theory and methods, New Generation Computing 19 (20) (1994) 629–680.
[2] S. Muggleton, Inverse entailment and Progol, New Generation Computing 13 (1995) 245–286.

[3] R. Khardon, Learning Horn expressions with LogAn-H, in: Proceedings of the 7th International Conference on Machine Learning, ICML '00, Morgan Kaufmann Publishers Inc., 2000, pp. 471–478.
[4] A. Srinivasan, The aleph manual, Tech. rep., Oxford University Computing Laboratory, Oxford, 2000.
[5] S. Benferhat, D. Dubois, H. Prade, Representing default rules in possibilistic logic, in: Proceedings of the 11th International Conference of on Principles of Knowledge Representation and Reasoning KR92, 1992, pp. 673–684.
[6] D. Dubois, J. Lang, H. Prade, Possibilistic logic, in: D.G., et al. (Eds.), Handbook of Logic in Artificial Intelligence and Logic Programming, vol. 3, Oxford University Press, 1994, pp. 439–513.
[7] R.L. Rivest, Learning decision lists, Machine Learning 2 (1987) 229–246.
[8] R. Khardon, Learning to take actions, Machine Learning 35 (1999) 57–90.
[9] M.E. Califf, Efficient and effective induction of first order decision lists, in: S. Matwin, C. Sammut (Eds.), Proceedings of the 12th International Conference of Inductive Logic Programming, ILP 2002, Springer, 2002, pp. 17–31.
[10] M. Bain, S.H. Muggleton, Non-monotonic learning, in: J.E. Hayes, D. Michie, E. Tyugu (Eds.), Machine Intelligence 12, Oxford University Press, Oxford, 1991, pp. 105–119.
[11] A. Srinivasan, S.M.M. Bain, Distinguishing exceptions from noise in non-monotonic learning, in: S. Muggleton (Ed.), Proceedings of the 2nd International Workshop on Inductive Logic Programming, 1992.
[12] S. Wrobel, On the proper definition of minimality in specialization and theory revision, in: Machine Learning: ECML-93, in: European Conference on Machine Learning, Proceedings, vol. 667, Springer, 1993, pp. 65–82.
[13] L. Martin, C. Vrain, A three-valued framework for the induction of general logic programs, in: L. De Raedt (Ed.), Advances in Inductive Logic Programming, IOS Press, 1996, pp. 219–235.
[14] E. Lamma, F. Riguzzi, L.M. Pereira, Learning three-valued logic programs, in: S. Dzeroski, P. Flach (Eds.), ILP-99 Late-Breaking Papers, 1999, pp. 30–35.
[15] C. Sakama, Nonmonotonic inductive logic programming, in: 6th International Conference on Logic Programming and Nonmonotonic Reasoning, vol. 2173, 2001, pp. 62–79.
[16] M. Serrurier, H. Prade, Coping with exceptions in multiclass ILP problems using possibilistic logic, in: Proc of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05), 2005, pp. 1761–1764.
[17] M. Serrurier, H. Prade, Gérer les exceptions en programmation logique inductive multiclasse avec la logique possibiliste, in: 15ème Congrès Francophone AFRIF-AFIA de Reconnaissance des Formes et d'Intelligence Artificielle, RFIA'06, tours, France, 2006.
[18] H. Blockeel, L.D. Raedt, N. Jacobs, B. Demoen, Scaling up inductive logic programming by learning from interpretations, Data Mining and Knowledge Discovery 3 (1999) 59–93.
[19] S. Benferhat, D. Dubois, H. Prade, Possibilistic and standard probabilistic semantics of conditional knowledge bases, Journal of Logic and Computation 9 (6) (1999) 873–895.
[20] M. Serrurier, H. Prade, G. Richard, A simulated annealing framework for ILP, in: R. Camacho, R. King, A. Srinivasan (Eds.), Proceedings of ILP 2004, in: Lecture Notes in Artificial Intelligence, vol. 3194, Springer, 2004, pp. 288–304.
[21] S. Lin, Computer solutions of the traveling salesman problem, Bell System Technical Journal 44 (1965) 2245–2269.
[22] S. Muggleton, W. Buntine, Machine invention of first order predicates by inverting resolution, in: Proceedings of the Fifth International Conference on Machine Learning (ICML 88), 1988, pp. 339–351.
[23] T.G. Dietterich, G. Bakiri, Solving multiclass learning problems by error-correcting output codes, Journal of Artificial Intelligence Research 2 (1995) 263–286.
[24] M. Eineborg, H. Boström, Classifying uncovered examples by rule stretching, in: C. Rouveirol, M. Sebag (Eds.), Proceedings of the 11th International Conference on Inductive Logic Programming, in: Lecture Notes in Artificial Intelligence, vol. 2157, Springer, 2001, pp. 41–50.
[25] S. Sinthupinyo, C. Nattee, T. Okada, B. Kijsirikul, Combining partial rules and winnow algorithm: results on classification of dopamine antagonist molecules, in: Proceedings of the 3rd International Workshop on Active Mining, 2004, pp. 83–92.
[26] L. De Raedt, K. Kersting, Probabilistic logic learning, SIGKDD Explor. Newsl. 5 (1) (2003) 31–48.
[27] P. Snow, Diverse confidence levels in a probabilistic semantics for conditional logics, Artif. Intell. 113 (1–2) (1999) 269–279.
[28] T. Horvath, P. Vojtas, GAP—rule discovery for graded classification, in: Working Notes of the ECML/PKDD 04 Workshop on Advances in Inductive Rule Learning, 2004.
[29] M. Serrurier, H. Prade, Possibilistic inductive logic programming, in: L. Godo (Ed.), Proceedings of European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'05), Barcelona, in: Lecture Notes in Artificial Intelligence, vol. 3571, Springer, Berlin Heidelberg, 2005, pp. 675–686.
[30] C. Ferri, P. Flach, J. Hernandez-Orallo, Learning decision trees using the area under the roc curve, in: C. Sammut (Ed.), Proceedings of the Nineteenth International Conference on Machine Learning (ICML 2002), Morgan Kaufmann, 2002, pp. 139–146.
[31] A. Srinivasan, R. King, S. Muggleton, The role of background knowledge: using a problem from chemistry to examine the performance of an ILP program, Tech. Rep. PRG-TR-08-99, Oxford University Computing Laboratory, Oxford, 1999.
[32] H. Lodhi, S. Muggleton, Is mutagenesis still challenging, in: ILP-05 Late-Breaking Papers, 2005, pp. 35–40.
[33] S. Džeroski, S. Schulze-Kremer, K. Heidtke, K. Siems, D. Wettschereck, Applying ILP to diterpene structure elucidation from ^{13}C NMR spectra, in: S. Muggleton (Ed.), Proceedings of the 6th International Workshop on Inductive Logic Programming, in: Lecture Notes in Artificial Intelligence, vol. 1314, Springer, 1996, pp. 41–54.
[34] J.U. Kietz, Learnability of description logic programs, in: S. Matwin, C. Sammut (Eds.), Proceedings of the 12th Inter. Conference of Inductive Logic Programming, ILP 2002, Springer, 2002, pp. 117–132.