Resource Bounded Induction and Deduction in First Order Logic

Michèle Sebag
LMS, Ecole Polytechnique, F-91128 Palaiseau
LRI, Université Paris-Sud, F-91405 Orsay
Michele.Sebag@polytechnique.fr

Abstract

In Machine Learning, there is a dilemma regarding the choice of a representation language: if a language allows the description of graphs, which is what many complex domains (e.g. chemistry, natural language) are described in terms of, deductive inference must support graph matching, which is NP-complete. And inductive inference cannot be less complex than deductive inference. The complexity of inference gets tractable iff hypotheses only involve small-size patterns.

An alternative to soundly use a short "perfect" knowledge is to use "as well as possible" an imperfect knowledge with no size restrictions.

A first step in this way is being able to accommodate relational knowledge with no size restrictions and still with acceptable, or resource-bounded, complexity. In order to do so, both inductive and deductive inference must be re-designed. This way, the defects or biases of deductive inference can be anticipated and corrected to some extent by inductive inference — and *vice versa*.

This paper continues a previous work which uses stochastic heuristics to achieve resource bounded learning and reasoning in First Order Logic. Two basic learning components are considered: the generality relationship and the generalization operator. Two unsophisticated stochastic tools approximating these components with controllable complexity, are presented, and we discuss the price to pay for controllability.

Preliminary experiments on the well-studied mutagenesis problem (regression-friendly and regression-unfriendly datasets) demonstrate the potential and the limitations of this approach.

Introduction

H. Simon (1982) distinguishes two ways of dealing with complex real-world situations. In the first one, "the description of the real world is radically simplified until reduced to a degree of complication that the decision maker can handle". The second approach is based on *satisficing*: "Satisficing seeks simplification in a somewhat different direction, retaining more of the detail of the real-world situation, but settling for a satisfactory, rather than approximate-best, decision".

This paper focuses on one particular complex real-world situation: the choice of a representation language for Machine Learning. First Order Logic (FOL) Languages, say Prolog, allow one to accurately model and handle complex problem domains: natural language is adequately represented via sequences of words, and relations among them; chemical molecules are sets of atoms, connected by simple or double bonds. Learning in Logic Programming (LP), termed Inductive Logic Programming (ILP) (Muggleton & De Raedt 1994) receives an ever growing attention, as it appears a royal road for learning in complex domains (Mooney 1997; Srinivasan 1997). Nevertheless, ILP is hindered by one major weakness of LP, namely the fact that inference is exponential in the length of the knowledge[1]. For instance, consider a clause stating that a chemical molecule is toxic if it contains a given pattern of atoms: checking whether this clause matches an actual molecule amounts to graph matching, which is exponential in the size of the pattern.

To sum up, Machine Learning faces the following dilemma: if a programming framework allows the description of graphs, which is what many complex domain problems are described in terms of, deductive inference must support graph matching, which is NP-complete in the general case. And inductive inference cannot be less complex than deductive inference. The complexity of inference gets tractable iff clauses only involve small-size patterns. Accordingly, human programmers tend to write short clauses, and ILP learners uses learning biases to restrict the length of the clauses they explore and produce. The complexity of knowledge is thus "reduced to a degree of complication that the decision maker can handle".

[1] Another weakness, not discussed hereafter, is that LP poorly handles numerical information. This limitation can be addressed by using a Constraint Logic Programming formalism, which is a superset of LP supporting interpreted predicates (Jaffar & Lassez 1987).

Figure 1: Generalization in First Order Logic: C generalizes D

$C:$	$poison(X):-$	$atom(X,Y',carbon),\ atom(X,Y'',hydrogen),\ bond(Y',Y'',1)$
$D:$	$poison(m):-$	$atom(m,m_1,hydrogen),\ atom(m,m_2,carbon),\ atom(m,m_3,carbon),\ bond(m_2,m_1,1)$

Figure 2: Rewriting $C = \mathcal{C}\gamma$ and $D = \mathcal{D}\delta$

$$\begin{array}{lcl}
\mathcal{C} & : & poison(X):-\ atom(X',Y',Z'),\ atom(X'',Y'',Z''),\ bond(U,V,W) \\
\gamma & = & (X=X'=X''),(Y'=U),(Y''=V),(Z'=carbon),(Z''=hydrogen),(W=1),(Z'\neq Z''),(Z'\neq W),(W\neq Z'') \\
\mathcal{D} & : & poison(R):-\ atom(R',S',T'),\ atom(R'',S'',T''),\ atom(R''',S''',T'''),\ bond(O,P,Q) \\
\delta & = & (R=R'=R''=R'''),(S'=m_1),(T'=hydrogen),(S''=m_2),(T''=carbon),(S'''=m_3,T'''=carbon),...
\end{array}$$

Figure 3: Tabular representation of FOL examples (excerpt). Non informed attributes are indicated by $\star$

	R	R'	S'	T'	R''	S''	T''	R'''	S'''	T'''	O	P	Q	$S''=O$	$S'''=O$
δ	m	m	m_1	*hy*	m	m_2	*ca*	m	m_3	*ca*	m_2	m_1	1	*true*	*false*
$\gamma\sigma_1$	m	m	m_1	*hy*	m	m_2	*ca*	$\star$	$\star$	$\star$	m_2	m_1	1	*true*	$\star$
$\gamma\sigma_2$	m	m	m_1	*hy*	$\star$	$\star$	$\star$	m	m_2	*ca*	m_2	m_1	1	$\star$	*true*

(For instance, FOIL (Quinlan 1990), FOCL (Pazzani & Kibler 1992), TILDE (Blockeel & Raedt 1997) use search biases; PROGOL (Muggleton 1995), REGAL (Giordana & Neri 1994) and REMO (Zucker & Ganascia 1996) use syntax biases).

This paper rather follows the other branch of the alternative discussed by Simon, and investigates what a "satisficing" framework for Machine Learning could be. Our approach is inspired from resource bounded reasoning (see (Zilberstein 1996) for a survey). It relies on two components: a representation change, meant to simplify the pattern-matching task (Kodratoff & Ganascia 1986); and a resource-bounded inference, replacing LP standard inference. Resource-bounded inference involves a randomized heuristic first used in (Sebag & Rouveirol 1997), which is recalled and discussed in section 2. In section 3, this heuristic is combined to the least general generalization (lgg) operator defined by Plotkin (Plotkin 1970), to derive a resource-bounded generalization operator and a similarity function. The approach is experimentally validated on the mutagenesis dataset, which has been extensively studied (Srinivasan *et al.* 1996; Sebag & Rouveirol 1997); the results obtained on the easy "regression-friendly" and on the difficult "regression-unfriendly" datasets, are compared. Last section finally presents some perspectives for further research.

Resource-Bounded Inference

One cornerstone of induction is the generality relation used to check whether a hypothesis covers an example and whether some hypothesis is more general than some other (Mitchell 1982). Most ILP learners use as generality relation the θ-subsumption defined by Plotkin (Plotkin 1970), for the sake of convenience[2] (Muggleton & De Raedt 1994). This section briefly recalls the definition of subsumption and discusses its limitations, trying to keep the presentation as intuitive as possible. We further present our working hypotheses and define a resource-bounded subsumption operator.

Standard subsumption

A formula C generalizes or *subsumes* a formula D, noted $D \prec C$, iff there exists a mapping σ from the variables in C onto the terms in D such that $C\sigma$ is included in D ($C\sigma \subseteq D$). C is said to generalize D according to the mapping, or substitution, σ.
For instance, C generalizes D (Figure 1): the mapping σ associating the hydrogen atom in C to the hydrogen atom in D, and the carbon atom in C to the first carbon atom in D actually satisfies $C\sigma \subseteq D$.

As shown in Figure 2, a FOL formula C can be rewritten as[3] $\mathcal{C}\gamma$, where $\mathcal{C}$ denotes the formula obtained from C by replacing all terms t_i in C by *distinct* variables X_j, and γ is the most specific formula deduced from the set of equalities ($X_j = t_i$).

This rewriting makes clear what can be done at an attribute-value level and what must be handled at a FOL level. Let σ now denote a mapping from the variables in $\mathcal{C}$ onto the variables in $\mathcal{D}$. It can be shown that formula $C = \mathcal{C}\gamma$ generalizes $D = \mathcal{D}\delta$ according to mapping σ iff both conditions A.1 and A.2 are satisfied:

A.1 : $\mathcal{C}$ generalizes $\mathcal{D}$ according to σ;

A.2 : δ implies the formula $\gamma\sigma$ obtained by replacing any variable X in γ by $\sigma(X)$.

Condition A.1 compares the FOL parts of C and D and is amenable to a boolean test: it fails iff C involves a predicate absent from D (remind that all variables in $\mathcal{C}$ and $\mathcal{D}$ are distinct). Assume thereafter that condition A.1 holds. Then let σ be defined by repeatedly associating any literal $p(X_1, X_2, ..)$ in $\mathcal{C}$ to

[2] θ-subsumption is weaker than logical implication as far as function symbols, recursive clauses, and/or background knowledge are concerned. However, ILP examples are often pre-processed so as to get rid of function symbols or intensional background knowledge, e.g. via flattening or saturation (Rouveirol 1994). In the case of recursive clauses, logical implication can also be brought back to the θ-subsumption of clauses derived from the ones at hand (Muggleton 1995). See (Lloyd 1987) and (Muggleton & De Raedt 1994) for a general presentation of LP and ILP.

[3] This rewriting relies on the fact that Logic Programming is embedded into Constraint Logic Programming. See (Jaffar & Lassez 1987) for a general presentation of CLP.

a literal $p(Y_1, Y_2, ..)$ in $\mathcal{D}$ built on the same predicate symbol p, and by setting $\sigma(X_i) = Y_i$. (Two examples of such mappings are given below).
Let Σ be the set of all such σ. By construction, σ satisfies condition A.1 iff it belongs to Σ.

Condition A.2 actually compares attribute-value information. Consider the attribute-value language defined by attributes Y_i and boolean attributes $(Y_i = Y_j)$, for all variables Y_i and Y_j in $\mathcal{D}$. Any set of constraints δ derived from writing $D = \mathcal{D}\delta$ can be represented in this attribute-value language (Figure 3). And it can be shown without difficulty that δ implies $\gamma\sigma$ iff the attribute-value representation of $\gamma\sigma$ is more general than that of δ.

In our example, let σ_1 be defined by mapping the second literal *atom* in $\mathcal{C}$ onto the first *atom* in $\mathcal{D}$ (which both correspond to *hydrogen* atoms in C and D) and the first literal *atom* in $\mathcal{C}$ onto the second *atom* in $\mathcal{D}$ (which both correspond to *carbon* atoms in C and D), and the only literal *bon* in $\mathcal{C}$ onto the only *bond* literal in $\mathcal{D}$. Let σ_2 be as σ_1 except for the fact that it maps the first literal *atom* in $\mathcal{C}$ onto the *third* literal *atom* in $\mathcal{D}$ (which both correspond to *carbon* atoms in C and D). Figure 3 shows an excerpt of the attribute-value representation of δ, $\gamma\sigma_1$ and $\gamma\sigma_2$. One sees that σ_1 satisfies A.2, and σ_2 does not.

The subsumption test then explores the space Σ, searching for a substitution σ satisfying condition A.2. This can be viewed as a constraint satisfaction problem (CSP) on the search space Σ. Indeed many heuristics devised to optimize the subsumption test closely resemble actual CSP heuristics[4]. Still, the subsumption test is NP-complete. If $p_1, \ldots, p_K$ denote the predicate symbols in $\mathcal{C}$, and if n_i and m_i respectively denote the number of literals built on p_i in $\mathcal{C}$ and $\mathcal{D}$, then the size of Σ is ${m_1}^{n_1} \times \ldots \times m_K^{n_K}$.

Limitations of induction and deduction

What precedes implies that the use of Logic Programming in weakly structured domains is severely limited. Consider for instance the chemistry domain, and let C be a clause involving 10 literals *atom* and 5 literals *bond*. Let D describe a medium-size molecule, involving 100 atoms and 80 bonds. Checking whether C generalizes D is in $100^{10} \times 80^5 \approx 3.10^{29}$. This implies that, even if such long clauses were available, it would be unrealistic to use them. Of course, common sense suggests not to extract hypotheses which cannot be exploited. Accordingly, ILP learners restrict themselves to explore usable clauses, via syntactic biases (Muggleton 1995; Zucker & Ganascia 1996; Giordana & Neri 1994) or search biases (Quinlan 1990; Pazzani & Kibler 1992; Blockeel & Raedt 1997). In the mutagenesis (King, Srinivasan, & Sternberg 1995) or in the PTE problem (Srinivasan 1997), they produce short clauses, including up to three or four literals *atom*.

This limitation can be addressed by better structuring the problem domain: e.g. the size of Σ is dramatically decreased if carbon atoms can be grouped into benzenic rings.

The alternative is to redesign a generality or subsumption relation, to accommodate weakly structured domains. Let us first reconsider the representation of such domains.

A new relational representation

The remainder of the paper relies on the following remark: the arguments of predicates can be divided into two categories depending on the nature of their domain of instantiation.
Some domains could be modified without affecting in any way the information contained in the examples. For instance, the second argument of predicate *atom*, and the first and second arguments of predicate *bond* have $\{m_1, m_2, m_3, ...\}$ for domain of instantiation. These constants stand for the "names" of the atoms in the molecules. However, even a chemistry-illiterate knows that the semantics of a molecule is invariant up to a consistent renaming of its atoms: actually, any constant in domain $\{m_1, m_2, m_3, ...\}$ could be replaced by any new constant without affecting the information of the examples. In other words, the grounding of arguments with such domains of instantiation does not convey any intrinsic information. These arguments are called *relational*.

Other arguments are called *valued*: e.g. the third argument of predicate *atom* indicates the element of the atom and has $\{carbon, hydrogen, oxygen, ..\}$ for domain of instantiation.

We assume that the category of the predicate arguments is explicitly declared by the expert, e.g.:

$$category(atom(Relational, Relational, Valued))$$

These declarations are quite similar to the *mode* declarations used in PROGOL (Muggleton 1995), that describe which arguments can be computed from which others, and serve to exclude non-usable hypotheses. In contrast, the *category* declarations are meant to support the way hypotheses are used, which is detailed further.

Two more assumptions are made.
- If a predicate does not involve any relational argument, there exists at most one literal built on this predicate in each example (e.g. the description of a

[4] For instance, (Geibel & Wysotzki 1997) takes advantage of the partitions on variables induced by γ and γ (two variables X and Y are in the same subset if the instantiation of X sets a constraint on the instantiation of Y), and notices that a substitution σ satisfying A.2 must preserve the partitions respectively induced by γ and γ. Similar decompositions are used in CSP (Weigel & Faltings 1997).

molecule involves at most one literal built on the *hydrophobicity* predicate). In other words, this predicate encodes one or several attributes.

• If a predicate involves relational argument(s), there do not exist two literals built on this predicate, with same instantiation of the relational arguments, and different instantiations of the valued arguments (e.g. one does not have $atom(m, m_1, carbon)$ and $atom(m, m_1, oxygen)$). In other words, valued arguments in a predicate encode functions of the relational arguments of this predicate.

These assumptions altogether imply that any example E involves a set of objects $o_1, \ldots o_k$, and that E is described by a set of attributes, the properties of objects o_i (functions of o_i), and more generally the properties of some tuples of these o_i.

These assumptions do not hold for all ILP problems; typically, the case of recursive clauses has not been considered. Still, these assumptions fit one most famous domain of application of ILP, that is chemistry. Actually, the "objects" of chemistry are the atoms; any molecule is described by some attributes (*lumo* or *logp*), functions of its atoms (*element*, *element-type*, and *electrical-charge*) and functions of pairs or tuples of atoms (*bond-type*, *methanol-radical*).
These assumptions are also satisfied in the Finite Element Methods problem (Dolsak & Muggleton 1991): the objects here are the vertices of the finite elements. Last, these assumptions are satisfied in all "family" problems (discovery of the "grand-parent" or "ancestor" relations).

These assumptions altogether allow one to reformulate any constrained or definite clause C as:

- a set of objects $\mathcal{O}(C) = \{c_1, \ldots c_K\}$,
- a set of constraints of the type:

$$[g(c_{i_1}, \ldots, c_{i_L}) = V]$$

where $(c_{i_1}, \ldots, c_{i_L})$ is a L-tuple of objects in $\mathcal{O}(C)$, g is a function of arity L, and V is a subset of the domain of value of g. In the particular case $L = 0$, the constraint just is a selector (Michalski 1983).

Within this relational representation, the subsumption test described in the previous paragraph slightly simplifies. Let C be a logical formula described as above, and let D be similarly described by a set of objects $\mathcal{O}(D) = \{d_1, \ldots d_M\}$ and a set of constraints $[g'(d_{j_1}, \ldots, d_{j_M}) = W]$.

Let Σ now denote the set of mappings from $\mathcal{O}(C)$ onto $\mathcal{O}(D)$. Then, C generalizes D iff both conditions B.1 and B.2 are satisfied:

B.1: The attribute-value part of C is more general than the attribute-value part of D, that is for each selector $[attribute = V]$ in C, there exists a selector $[attribute = W]$ in D such that V includes W;

B.2: There exists a mapping σ in Σ such that for each constraint $[g(c_{i_1}, \ldots, c_{i_L}) = V]$ in C, there exists a constraint $[g'(d_{j_1}, \ldots, d_{j_M}) = W]$ in D, with $g = g'$ and $\sigma(c_{i_k}) = d_{j_k}$ for $k = 1 \ldots L = M$ and V includes W.

Condition B.1 is tested with (negligible) linear complexity in the number of attributes. Only cases where $B.1$ is satisfied are thus examined thereafter.

Condition B.2 is tested with linear complexity in the size of Σ and in the numbers of constraints in C and D. But the size of Σ is of order[5] $|\mathcal{O}(D)|^{|\mathcal{O}(C)|}$. As before, the only way for the subsumption test to get tractable is to restrict C.

Resource-bounded subsumption

An alternative to the exhaustive exploration of Σ, first investigated in (Sebag & Rouveirol 1997), is to sample Σ by Monte-Carlo like methods. Practically, one decides whether condition B.2 above is satisfied on the only basis of some K mappings $\sigma_1, \ldots \sigma_K$ drawn in Σ with uniform probability. This defines a randomized estimate of condition B.2, which answers *true* if at least one σ_i for $i = 1 \ldots K$ satisfies B.2. This leads to a randomized estimate of the subsumption test, called K-subsumption. By definition, C ***K-subsumes*** D, noted $D \prec_K C$, if condition B.1 holds, and condition B.2 holds for at least one σ_i among K mappings independently selected in Σ with uniform probability.

The complexity of K-subsumption is in $\mathcal{O}(K \times |C| \times |D|)$, where $|C|$ and $|D|$ denote the number of constraints in C and D: as desired, the resources are controlled by the user through parameter K.

Let us examine the price to pay for this controllability, that is, the probability for K-subsumption to differ from standard subsumption. K-subsumption is obviously sound; if C K-subsumes D, then C subsumes D. But the converse is not true, and K-subsumption is not complete. The chance of error thus is the probability of having $D \not\prec_K C$ conditioned by $D \prec C$.

Let p denote the fraction of mappings in Σ satisfying condition B.2; p is strictly positive as $D \prec C$. The probability of selecting K mappings *not* satisfying B.2 is $(1-p)^K$; therefore

$$Proba(D \not\prec_K C \mid D \prec C) \;=\; (1-p)^K$$

so the probability of error goes to 0 as K goes to infinity. Unfortunately, this does not mean that the convergence is fast. In our toy example, Σ contains 9 mappings (from $\{1, 2\}$ onto $\{1, 2, 3\}$) and only one of them satisfies B.2. Then, in order to detect that $D \prec_K C$ with confidence 95%, K must be greater than $log(.05)/\ log(8/9) \approx 26$...

More generally, K-subsumption provides an overly specific estimate of standard subsumption. The deductive inference based on K-subsumption then tends to be overly cautious and deliver much fewer results than expected.

[5] The gain compared to the standard logical formalism is that the number of objects is usually less than the number of literals built on a same predicate symbol.

Figure 4: Least general generalization

$E:$	$atom(e, e_1, carbon), atom(e, e_2, hydrogen), bond(e_1, e_2, 1)$
$F:$	$atom(f, f_1, hydrogen), atom(f, f_2, carbon), atom(f, f_3, carbon)$
$lgg(E, F):$	$atom(S, X', X"), atom(S, Y', carbon), atom(S, Z', carbon), atom(S, U', hydrogen), atom(S, V', V"), atom(S, W', W")$
$rlgg(E, F):$	$atom(S, X', X"), atom(S, Y', carbon), atom(S, Z', carbon), atom(S, U', hydrogen)$

Figure 5: Redescription of E and F in our simple relational representation

$E:$	$\mathcal{O}(E) =$	(e_1, e_2)	$[elem(e_1) = carbon]$, $[elem(e_2) = hydrogen]$, $[bond\_type(e_1, e_2) = 1]$
$F:$	$\mathcal{O}(F) =$	(f_1, f_2, f_3)	$[elem(f_1) = hydrogen]$, $[elem(f_2) = carbon]$, $[elem(f_3) = carbon]$

When induction anticipates a deduction bias

Let us re-examine the elementary question stated as *does C generalizes D*, and focus on the context of this question. In both contexts of induction or classification, which we are mostly concerned with[6], D is given: D is a training example in an induction context, and an instance to classify in a classification context. In contrast, C stands for the hypothesis extracted by induction and exploited by deduction.

If deductive inference is under control and relies on K-subsumption, it tends to be over-specific. But inductive inference is also under our control: we might design an induction bias, correcting by anticipation the deduction bias toward specificity. This leads to a new problem integrating induction and deduction:

Find a hypothesis C_K such that for all example D,
there exists a "good" hypothesis C with
"C_K K-subsumes D" $\approx$ "C subsumes D".

where '$\approx$' stands for "correctly approximates".

As K-subsumption is overly specific, C_K must be overly general. Further, as parameter K allows us to tune the degree of specificity of K-subsumption, we symmetrically search for a family of nested overly general hypotheses, and a means of controlling their generality. It has been experimentally shown (Sebag & Rouveirol 1997) that the over-specificity of K-subsumption can be effectively resisted by an overly general, Version space-based, learning approach (Sebag 1996). We examine in the next section another use of K-subsumption.

Resource Bounded Generalization and Similarity

Another cornerstone of induction is the generalization operator, which takes as input two examples or hypotheses E and F and produces a hypothesis $G(E,F)$ more general than these. As the maximally general hypothesis of the language obviously is an uninteresting solution, one must somehow restrict the generality of $G(E,F)$. This restriction is either absolute (e.g. $G(E,F)$ being of minimal generality such that it generalizes E and F) or relative to other examples (e.g. $G(E,F)$ being of maximal generality such that it does *not* generalize some other examples).

[6]Another concern is that of *explanation*, which will be considered in a further work.

This section describes one generalization operator most studied in ILP, the least general generalization (Plotkin 1970), and discusses its limitations. A stochastic approach is used to derive a generalization operator and a similarity index, both with resource-bounded complexity.

Least General Generalization

The least general generalization (lgg) operator has been defined by Plotkin (Plotkin 1970) (see (Muggleton 1992) for a detailed presentation). Given two examples E and F, lgg constructs what would be the most specific hyper-rectangle containing E and F, if E and F were described in an attribute-value language.

Let E and F be two formulas; the lgg of E and F is the conjunction of the lgg of l_E and l_F, for all literals l_E and l_F respectively in E and F such that they are built on the same predicate symbol. The lgg of two literals $p(s_1, s_2, \ldots)$ and $p(t_1, t_2, \ldots)$ built on the same predicate p is defined from the lgg of their arguments s_i and t_i:

$$lgg(\ p(\ s_1,\ s_2, \ldots),\ p(\ t_1,\ t_2, \ldots)) = p\ (\ lgg(s_1, t_1),\ lgg(s_2, t_2), \ldots)$$

where the lgg of two arguments is usually defined depending on their domain of instantiation. The lgg is then reduced (rlgg) to remove redundant literals, i.e. literals that can be deduced from the rest of the formula (Figure 4).

The limitations of lgg are twofold. First, lgg produces long formulas (the size of the output being the product of the sizes of the input). Second, $lgg(E, F)$ usually covers very few examples besides E and F. One then iteratively constructs $lgg(lgg(E, F), G)$, then $lgg(lgg(lgg(E, F), G), H)$, and so forth. The size of the result is exponential in the number of examples considered.

Resource Bounded Generalization

Let E and F be two logical formulas in the relational representation defined above (Figure 5). Let Σ be the set of mappings from $\mathcal{O}(E)$ onto $\mathcal{O}(F)$. Let $d_E = [g(e_{i_1}, \ldots e_{i_M}) = V]$ and $d_F = [g'(f_{i_1}, \ldots f_{i_N}) = W]$ respectively denote two constraints in E and F. For any σ in Σ, the σ-generalization of d_E and d_F, noted $gen_\sigma(d_E, d_F)$ is set to d_E iff $g = g'$, $\sigma(e_{i_j}) = f_{i_j}$ for $j = 1 \ldots L$ and $V = W$; and $gen_\sigma(d_E, d_F)$ is undefined otherwise.

The σ-generalization of examples E and F, noted

$gen_\sigma(E,F)$ is defined as the conjunction of the σ-generalizations of their constraints whenever these are defined. For instance, if σ maps e_1 onto f_2 and e_2 onto f_1, the σ-generalization of E and F is:

$$gen_\sigma(E,F) : (e_1, e_2)$$
$$[elem(e_1) = carbon], [elem(e_2) = hydrogen]$$

Note that, up to a renaming of the objects, $gen_\sigma(E,F) = gen_\sigma(F,E)$. For each mapping σ in Σ, let $n(\sigma)$ denote the number of constraints in $gen_\sigma(E,F)$: the greater $n(\sigma)$, the more specific $gen_\sigma(E,F)$.

A resource-bounded generalization operator called K-generalization, is defined as follows. Let $\sigma_1, \ldots \sigma_K$ be K mappings drawn in Σ with uniform probability, and let σ_i be such that $n(\sigma_i)$ is the maximum of $n(\sigma_j)$ for $j = 1 \ldots K$ (selected at random in case there is a tie). Then, the K-generalization of E and F, noted $gen_K(E,F)$, is set to $gen_{\sigma_i}(E,F)$. A resource-bounded similarity function noted $sim_K(E,F)$ is defined as the number of constraints in $gen_K(E,F)$.

The complexity of K-generalization and sim_K is in $\mathcal{O}(K \times |E| \times |F|)$, where $|E|$ and $|F|$ respectively denote the number of constraints in E and F.

Discussion

Let us first briefly recall some related work based on FOL similarities. (Similarities based on rewriting the initial representation (Sebag 1997; Schadler & Wysotzki 1997) are not considered). KBG (Bisson 1990) computes the similarity of all pairs of examples/hypotheses, and uses this similarity function to guide the generalization. RIBL (Emde & Wettscherek 1996) computes the similarity of the instance to classify with all training examples, and classifies the instance at hand by means of a standard k-nearest neighbor approach.

Both KBG and RIBL require elementary distances to be defined on the domains of instantiation of the arguments. These distances are used to compute the elementary similarity of non-valued constants (called "objects" within our framework) that respectively appear in examples E and F. For instance, let example E describe "objects" $John$ and Ann, who are cousins, with $John$ being 24 year old and Ann living in $New-York$. Let example F describe "objects" $Peter$, $Marc$ and $Jane$, such that $Marc$ and $Jane$ are cousins, $Jane$ lives in $San\text{-}Francisco$ and $Peter$ is 20-year old. The similarity between $John$ and $Marc$ depends on the similarity between their respective cousins Ann and $Jane$, which itself depends on the similarity between $John$ and $Marc$ and on the distance between $New\text{-}York$ and $San\text{-}Francisco$.

KBG and RIBL compute the similarity between objects in different ways. KBG involves user-supplied weights; e.g. depending on the relative weight of predicates *age* and *cousin*, one finds that $John$ is most similar to $Peter$ or $Marc$. The similarity of all pairs of objects is then determined by resolving a set of linear equations. In opposition, RIBL automatically determines the predicate weights[7], and computes the similarity of any two objects by an iterative method.

Both KBG and RIBL use these elementary similarities to define a mapping σ from E on F, by repeatedly mapping each object e_i in E onto the object f_j in F most similar to e_i when possible. Such a σ induces an attribute-value "alignment" of E and F. Thereby the generalization step in KBG is brought back to attribute-value generalization, and the global similarity of E and F in RIBL proceeds like a weighted Euclidean-like distance.

In other words, KBG and RIBL attempt to find a global optimum (the best σ) by combining locally optimal choices; the elementary similarity function must then capture information as global as possible. For instance, RIBL sets an upper-bound on the number ℓ of links examined: if ℓ is set to 3, cousins Ann and $Jane$ are considered when computing the similarity of $John$ and $Marc$, as they are directly linked to $John$ and $Marc$; the similarity of Ann's and $Jane$'s respective daughters, (and there can be many of them), would also be considered; but the similarity of their daughters' dogs would be discarded.

The ability of the elementary similarity function to evaluate whether $John$ resembles $Marc$ from a global point of view certainly increases with ℓ ; unfortunately the complexity of elementary similarity is exponential in ℓ.

On the opposite, K-generalization attempts to find a "rather good" σ: it uniformly selects a bunch of mappings in Σ and retains the mapping σ optimizing the *very poor, but global* criterion $n(\sigma)$. Indeed, this criterion could be refined by using more elaborate elementary similarities (see further). In summary, all approaches are myopic in some sense: KBG and RIBL approaches rely on detailed local points of view; K-generalization uses a global but very approximative point of view.

K-generalization is to be more general than KBG-like generalization. In the same line, the similarity function sim_K is to be poorer than that used in RIBL: it gives the maximum size of the set of constraints shared by E and F, estimated from a Monte-Carlo process. Still, K-generalization is more and more specific as K goes to infinity; for large enough values of K, the K-generalization should be equal or more specific than KBG-generalization; and the neighborhoods based on the RIBL similarity should be close to those based on sim_K.

This raises the question of whether K-generalization or sim_K could achieve reasonable results, *for reasonable values of* K.

[7] The accuracy of weights is evaluated from the rate of test examples correctly classified; finding a good set of weights then reformulates as an optimization problem.

Table 1: *Reference results on the mutagenesis dataset (10fold cross-validation on 188; leave-one-out on 42)*

		188 data set (default: 66 ±3)			42 data set (default: 69 ±7)	
	Description	$\mathcal{B}_0$	$\mathcal{B}_1$	$\mathcal{B}_4$	$\mathcal{B}_0$	$\mathcal{B}_4$
Attribute-value	REG	89 ±2			67 ±7	
	DT	85 ±3			67 ±7	
	NN	89 ±2			64 ±7	
First Order Logic	Progol (Muggleton 1995)		76 ±3	88 ±2		83 ±6
	Foil (Quinlan 1990)		61 ±6	86 ±3		—
	Still (Sebag & Rouveirol 1997)		83 ±4	93 ±4		—

Goal of the experiments

This paper only studies the relevance of the similarity function defined by sim_K. This relevance is evaluated from the predictive accuracy of two long-known similarity-based data analysis algorithms (Duda & Hart 1973).

The first one is the well-known k-nearest neighbor classification (kNN) process (see (Wettscherek & Dietterich 1995) for a comprehensive discussion). It computes the similarity between the case C at hand and any training example, and classifies C according to the majority vote of the k training examples most similar to C. The complexity of classification is in $\mathcal{O}(N \times |sim|)$, if N denotes the number of training examples and $|sim|$ the complexity of one similarity calculation.

Apparently, kNN does not involve any learning stage; however, the adjustment of parameter k should be viewed as a hidden learning stage. Anyway, it should be taken into account for fair algorithmic comparisons (Scheffer & Herbrich 1997); as this adjustment relies in any case on the classification of the training examples themselves (Kohavi & John 1995), its complexity is in $\mathcal{O}(N^2 \times |sim|)$.

A variant of the k-nearest neighbor algorithm, termed *consistent nearest neighbor* (cNN) and not involving any parameter, is defined as follows. For each training example E, let F_E be a "near-miss" of E, i.e. a training example most similar to E such that it does not belong to the same class as E. One computes the similarity between the case C at hand and any training example; then, one classifies C according to the majority vote of all training examples E such that C is closer to E than its near-miss ($sim_K(C,E) > sim_K(E,F_E)$). The complexity of classification is also in $\mathcal{O}(N \times |sim|)$. The learning stage includes the computation of the near-misses of the training examples, in $\mathcal{O}(N^2 \times |sim|)$.

Experimental validation

To ease comparisons, experiments consider one well-studied ILP benchmark, the mutagenesis problem (King, Srinivasan, & Sternberg 1995).

The domain

The mutagenesis problem concerns the classification of chemical organic molecules into two classes, *active* and *inactive* (King, Srinivasan, & Sternberg 1995)(see (Srinivasan *et al.* 1996; Srinivasan 1997) for a detailed discussion about using ILP in chemistry domains). Several descriptions of the molecules are considered. The first one, thereafter noted $\mathcal{B}_0$, is an attribute-value description which was used by propositional classifiers, e.g. linear regression (REG), decision tree (DT), neural nets with 3 hidden units (NN). A basic FOL description, noted $\mathcal{B}_1$, only contains the description of the atoms and the bonds in the molecules. The most complete FOL description, noted $\mathcal{B}_4$, is $\mathcal{B}_1$ completed with expert attributes (e.g. hydrophobicity) and chemical concepts (e.g. methyl groups, benzenic rings).

The total dataset includes 230 molecules, which are decomposed into a 188 molecule dataset called "regression-friendly" and a 42 molecule dataset called "regression-unfriendly". Propositional learners, including linear regression (REG), decision tree (DT), neural nets with 3 hidden units (NN) have been experimented on both datasets (with description $\mathcal{B}_0$). Most ILP learners have been experimented on this domain, among which are FOIL (Quinlan 1990), PROGOL (Muggleton 1995), FORS (Karalic 1995), and STILL (Sebag & Rouveirol 1997). The percentage of test examples correctly classified together with the standard deviation (measured by a 10-fold cross-validation in the 188-molecule case and a leave-one-out process in the 42-molecule case) are reported from (Srinivasan *et al.* 1996; Srinivasan & Muggleton 1995; Sebag & Rouveirol 1997) in Table 1.

Experimental setting

It is recommended to evaluate a stochastic algorithm from its results averaged over no less than 15 independent runs (with different random seeds) (Jr 1994). We therefore average the results of our approach over 15 independent runs; to permit fair comparisons, each run is a 10-fold cross-validation for the 188 dataset, and a leave-one-out validation for the 42 dataset.

We used two stochastic mechanisms to sample the matching space Σ. The unbiased sampling mechanism (US) uses a uniform distribution: while possible, one selects a non yet selected object e_i with uniform probability in $\mathcal{O}(E)$, and then selects a non yet selected object f_j with uniform probability in $\mathcal{O}(F)$; the sample σ is iteratively defined by $\sigma(e_i) = f_j$.

The biased sampling mechanism (BS) differs from the previous one in the selection of object f_j, which depends on e_i: e_i is mapped onto the atom f_j most similar to e_i and non yet selected. The description of the dataset makes this bias rather straightforward: two atoms can have same description (same atom-element, same atom-type and same electric-charge); otherwise, they can be very similar (same atom-element and same atom-type); otherwise, they are similar (same atom-element); otherwise, they are different.

We further restrict ourselves to considering the $\mathcal{B}_1$ description of the examples: the attributes (*logp*, *lumo*, etc) and chemical concepts are not taken into account. The reason for this restriction is the following: As such, similarity sim_K would not take much advantage from the attributes; even if two examples had same attribute-value description, this would only increase their similarity by 5 (the number of attributes), which is nothing compared to the number of constraints in a molecule (3 times the number of atoms[8] plus the number of bonds).

The fact that having same hydrophobicity is more important than having two atoms with same element should be acknowledged via weights. However, our intention is to study the general behavior of sim_K, rather than how this behavior could be improved by weights.

We thus focus on how the behavior of sim_K depends on both the number K of samples selected in Σ, and the sampling mechanism, biased or unbiased.

Results

Tables 2 and 3 summarize the results obtained by k-nearest neighbors (kNN with $k = 1, 3$) and consistent nearest neighbors (cNN), based on the similarity sim_K. For each experiment, we indicate the predictive accuracy and the standard deviation, averaged over 15 independent runs.

Learning is linear in K and quadratic in the number of training examples. On a Pentium 100, the learning time for $K = 100$ is 31 minutes for biased sampling and 21 minutes for unbiased sampling on the 188 dataset, and 80 seconds for biased sampling on the 42 dataset. The classification time is negligible.

	K	Pred. acc
cNN	2	87 ± 7
1NN		89 ± 6
3NN		85 ± 8
cNN	10	89 ± 6
1NN		89 ± 6
3NN		86 ± 8
cNN	50	89 ± 5
1NN		89 ± 7
3NN		86 ± 8

(a) Biased sampling

	K	Pred. acc
cNN	100	80 ± 9
1NN		78 ± 8
3NN		78 ± 10
cNN	200	79 ± 8
1NN		79 ± 8
3NN		79 ± 10
cNN	300	79 ± 9
1NN		80 ± 8
3NN		80 ± 10

(b) Unbiased sampling

Table 2: Predictive accuracy of sim_K on the 188-dataset (averaged on 15 runs; 10fold cross validation)

The results obtained on the 188 dataset when sim_K is constructed with biased and unbiased sampling are respectively shown in Table 2(a) and Table 2(b). K varies in $\{2, 10, 50\}$ for the biased sampling, and in $\{100, 200, 300\}$ for the unbiased sampling.

	K	Pred. acc
cNN	100	74 ± 14
1NN		73 ± 17
3NN		79 ± 12
cNN	500	75 ± 14
1NN		74 ± 18
3NN		79 ± 11
cNN	1000	75 ± 14
1NN		74 ± 18
3NN		80 ± 9

Table 3: Predictive accuracy of sim_K on the 42-dataset (averaged on 15 runs; leave-one-out validation)

The results obtained on the 42 dataset when sim_K is constructed with biased sampling are shown in Table 3. K varies in $\{100, 500, 1000\}$ for the biased sampling. The results of unbiased sampling are not shown as they are comparable to the default rate for reasonable values of K.

Discussion

On both datasets, the biased sampling leads to results significantly better than those of unbiased sampling. This could have been expected, as biased sampling achieves a partial alignment of the examples, and sim_K just estimates the maximum number of parts in the examples which can be aligned. In particular, biased sampling strongly decreases the standard deviation of the results obtained for a given number K of samples.

On the 188 dataset with biased sampling, competitive results are obtained for low values of K, and no improvement is obtained by increasing K. Further, there exist almost no variation (not shown here) between the runs. With respect to computational cost, this approach needs less than 900 seconds on Pentium-120, which is much faster than PROGOL (46 000 seconds on HP-735) and much slower than STILL (120 seconds on HP-735).

What is surprising is the fact that despite sim_K uses the basic $\mathcal{B}_1$ description of the data, it obtains same results as FOIL or PROGOL using the much richer $\mathcal{B}_4$ description (circa 89%). As a matter of fact, FOIL or PROGOL obtain *significantly worse* results than sim_K when they consider description $\mathcal{B}_1$ (respectively 61% or 76%).

On the whole, the 188 dataset seems to be rather clean and regular; the absence of noise is witnessed by the fact that better results are obtained when fewer neighbors are considered to classify the examples.

The 42 data set is clearly more difficult. Similarity sim_K with unbiased sampling is not significantly different from the default classifier. With biased sampling, the predictive accuracy slowly increases as K increases, but the trend is not statistically significant

[8] Since each atom is described by its element (e.g. carbon), its element-type (e.g. 22) and its electric charge (e.g. -.33).

given the high standard deviation. On this dataset, sim_K using description $\mathcal{B}_1$ stays behind PROGOL using description $\mathcal{B}_4$. It would be interesting to see how PROGOL performs with description $\mathcal{B}_1$.

Note that the information captured by sim_K reflects the FOL nature of examples, but not that much: the bonds between atoms are responsible for about 30% of $sim_K(E, F)$, the remaining 70% coming from the comparison of the distribution of atom properties (elements, types and electric charges) in E and F, which could be accommodated within an attribute-value description.

Conclusion and Perspectives

In some relational domains, the background knowledge is sufficiently structured for deductive inference to be tractable; and, provided one only wants to extract short pieces of knowledge, inductive inference remains tractable as well.

Focusing on weakly structured relational domains, we have investigated whether acceptable, resource-bounded, inference can be achieved in such domains with no restrictions on the knowledge size. Both deductive and inductive inference need be reconsidered. This way, the biases of deductive inference will hopefully be anticipated and corrected to some extent at the level of inductive inference — and *vice versa*.

In this line, this paper first describes a new representation change, based on objects, functions, and constraints. This change can to some extent be viewed as opposite to the flattening operator[9] (Rouveirol 1994). Its goal is to provide a simpler description of the pattern-matching task (Kodratoff & Ganascia 1986), amenable to exploring a set of mappings Σ.

In this framework, several resource-bounded operators, all based on the stochastic sampling of Σ, are defined. This paper mostly focuses on an unsophisticated similarity function, estimating the maximum number of parts of two examples which can be "aligned" by means of a Monte-Carlo like method.

It is experimentally shown that the k-nearest neighbor process based on this similarity gives results comparable to the state of the art on an "easy" ILP problem. What is unexpected is that our unsophisticated approach, provided with the poorest FOL description of the problem, obtains results comparable to those of prominent ILP learners provided with the most complete FOL description available (these learners actually give far worse results when they are provided with the poor description). This seems to confirm that different learning strategies should be adopted depending on the quality of the background knowledge.

[9]The flattening operator removes the function symbols arising in FOL formulas, by "promoting" them to predicate symbols. This allows one to sidestep the decidability problems caused by function symbols, though a judicious choice of the background knowledge.

On-going experiments evaluate how the generalization operator itself interacts with the resource-bounded deductive inference. Further work will take a better account of the numerical information[10], to refine both the similarity function and the generalization operator.

References

Bisson, G. 1990. KBG: A knowledge base generalizer. In Porter, B., and Mooney, R., eds., *Proceedings of the 7th International Conference on Machine Learning*, 9–15. Morgan Kaufmann.

Blockeel, H., and Raedt, L. D. 1997. Experiments with top-down induction of decision trees. In *ESPRIT Project LTR 20237 ILP²*. Partial Progress Report 1.

Dolsak, D., and Muggleton, S. 1991. The application of ILP to finite element mesh design. In Muggleton, S., ed., *Proceedings of the first International Workshop on Inductive Logic Programming*, 453–472. Academic Press.

Duda, R., and Hart, P. 1973. *Pattern Classification and scene analysis*. John Wiley and sons, Menlo Park, CA.

Emde, W., and Wettscherek, D. 1996. Relational instance based learning. In Saitta, L., ed., *Proceedings of the 13th International Conference on Machine Learning*, 122–130. Morgan Kaufmann.

Geibel, P., and Wysotzki, F. 1997. Alogical framework for graph theoretical decision tree learning. In *Proceedings of ILP-97*, 173–180. Springer Verlag, LNCS 1297.

Giordana, A., and Neri, F. 1994. Search internsive concept induction. *Evolutionary Computation* 3(4):375–416.

Jaffar, J., and Lassez, J. L. 1987. Constraint logic programming. In *Proc. of the fourteenth ACM Symposium on the Principles of Programming Languages*, 111–119.

Karalic, A. 1995. *First Order Regression*. Ph.D. Dissertation, Institut Josef Stefan, Ljubljana, Slovenia.

Kinnear, E. K. 1994. A perspective on GP. In K. E. Kinnear, J., ed., *Advances in Genetic Programming*. Cambridge, MA: MIT Press. 3–19.

King, R.; Srinivasan, A.; and Sternberg, M. 1995. Relating chemical activity to structure: an examination of ILP successes. *New Gen. Comput.* 13.

Kodratoff, Y., and Ganascia, J.-G. 1986. Improving the generalization step in learning. In Michalski, R.; Carbonell, J.; and Mitchell, T., eds., *Ma-*

[10]K-generalization so far involves boolean elementary similarities: with same notation as in the second section, $n(\sigma)$ is incremented iff $V = V'$. This could be refined for instance by using a threshold T, and incrementing $n(\sigma)$ iff the increase of $gen(V, V')$ compared to V is less than T% of the domain of value of f.

chine Learning: an artificial intelligence approach, volume 2. Morgan Kaufmann. 215–244.

Kohavi, R., and John, G. 1995. Automatic parameter selection by minimizing estimated error. In Prieditis, A., and Russell, S., eds., *Proceedings of the 12th International Conference on Machine Learning*, 304–312. Morgan Kaufmann.

Lloyd, J. 1987. *Foundations of logic programming.* Springer-Verlag, 2nd edition.

Michalski, R. 1983. A theory and methodology of inductive learning. In Michalski, R.; Carbonell, J.; and Mitchell, T., eds., *Machine Learning: an artificial intelligence approach*, volume 1. Morgan Kaufmann. 83–134.

Mitchell, T. 1982. Generalization as search. *Artificial Intelligence* 18:203–226.

Mooney, R. 1997. ILP for natural language processing. In Muggleton, S., ed., *Inductive Logic Programming, Selected papers of ILP'96.* Springer-Verlag. 3–22.

Muggleton, S., and De Raedt, L. 1994. Inductive logic programming: Theory and methods. *Journal of Logic Programming* 19:629–679.

Muggleton, S. 1992. Inductive logic programming. In Muggelton, S., ed., *Inductive Logic Programming.* Academic Press.

Muggleton, S. 1995. Inverse entailment and PROGOL. *New Gen. Comput.* 13:245–286.

Pazzani, M., and Kibler, D. 1992. The role of prior knowledge in inductive learning. *Machine Learning* 9:54–97.

Plotkin, G. 1970. A note on inductive generalization. In *Machine Intelligence*, volume 5. Edinburgh University Press.

Quinlan, J. 1990. Learning logical definition from relations. *Machine Learning* 5:239–266.

Rouveirol, C. 1994. Flattening and saturation: Two representation changes for generalisation. *Machine Learning* 14:219–232.

Schadler, K., and Wysotzki, F. 1997. A connectionnist approach to structural similarity detection as a basis of clustering, classification and feature detection. In Kolski, J., and Zytkow, J., eds., *Proceedings of PKDD'97*, 254–264. Springer Verlag.

Scheffer, T., and Herbrich, R. 1997. Unbiased assessment of learning algorithms. In *Proceedings of IJCAI-97*, 798–803. Morgan Kaufmann.

Sebag, M., and Rouveirol, C. 1997. Tractable induction and classification in FOL. In *Proceedings of IJCAI-97*, 888–892. Morgan Kaufmann.

Sebag, M. 1996. Delaying the choice of bias: A disjunctive version space approach. In Saitta, L., ed., *Proceedings of the 13th International Conference on Machine Learning*, 444–452. Morgan Kaufmann.

Sebag, M. 1997. Distance induction in FOL. In *Proceedings of ILP-97*, 264–272. Springer Verlag, LNCS 1297.

Simon, H. 1982. *Models of Bounded Rationality.* Cambridge, MIT Press.

Srinivasan, A., and Muggleton, S. 1995. Comparing the use of background knowledge by two ILP systems. In de Raedt, L., ed., *Proceedings of ILP-95.* Katholieke Universiteit Leuven.

Srinivasan, A.; Muggleton, S.; Sternberg, M.; and King, R. 1996. Theories for mutagenicity: a study in first order and feature-based induction. *Artificial Intelligence* 85:277–299.

Srinivasan, A. 1997. The predictive toxicology evaluation challenge. In *Proceedings of IJCAI-97*, 4–8. Morgan Kaufmann.

Weigel, R., and Faltings, B. 1997. Structuring techniques for constraint satisfaction problems. In *Proceedings of IJCAI-97*, 418–423. Morgan Kaufmann.

Wettscherek, D., and Dietterich, T. 1995. An experimental comparison of the nearest-neighbor and nearest hyperrectangle algorithms. *Machine Learning* 19:5–27.

Zilberstein, S. 1996. Resource-bounded reasoning in intelligent systems. *Computing Surveys* 28(4).

Zucker, J.-D., and Ganascia, J.-G. 1996. Representation changes for efficient learning in structural domains. In Saitta, L., ed., *Proceedings of the 13th International Conference on Machine Learning*, 543–551. Morgan Kaufmann.