From: Proceedings of the Third International Conference on Multistrategy Learning. Copyright © 1996, AAAI (www.aaai.org). All rights reserved.

Inductive Logic Programming + Stochastic Bias = Polynomial Approximate Learning

Michèle Sebag [1,2] and Céline Rouveirol[2] and Jean-François Puget[3]
(1) LMS, CNRS-URA 317, Ecole Polytechnique, 91128 Palaiseau
(2) LRI, CNRS-URA 410, Université Paris Sud, 91405 Orsay
(3) ILOG, 9 rue de Verdun, 94250 Gentilly
FRANCE
Michele.Sebag@polytechnique.fr, Celine.Rouveirol@lri.fr, puget@ilog.ilog.fr

Abstract

A major difficulty in Inductive Logic Programming (ILP) lies in the size of the hypothesis space, and in the number of possible matchings between a candidate hypothesis and a training example.
This paper investigates the use of a stochastic bias in order to make induction a tractable problem. A goal-directed sampling mechanism of the matching space is implemented. The exhaustive exploration of the matching space is replaced by considering a fixed, user-supplied, number of samples. One thereby constructs a theory which is only approximately consistent, with a polynomial computational complexity in term of the size of the data.
This approach significantly differs from the ILP approaches based on genetic algorithms or genetic programming, where stochastic sampling is directly used to explore the hypothesis space; in our approach, the sampling mechanism is combined to induction instead of replacing it.
Experiments on the mutagenicity problem fully validates the approach in terms of both predictive accuracy and computational cost.

Introduction

The framework of Inductive Logic Programming (Muggleton & De Raedt 1994) allows induction to handle problems having relational descriptions. This very expressive formalism however rises two major questions: that of dealing with numerical values, and that of mastering the computational complexity pertaining to first-order logic.

Handling numbers in ILP has mainly been tackled in two ways: the first one consists in transforming the current ILP problem into an attribute-value problem, as done in the LINUS approach (Lavrac, Dzeroski, & Grobelnick 1991) or via moriological reformulations (Zucker & Ganascia 1994). Another possibility consists in providing the learner with some kind of "numerical knowledge" (e.g. the definition of predicate `less-than`); this knowledge can either be built in the learner, as in *FOIL* (Quinlan 1990), or added to the domain theory, as in *PROGOL* (Muggleton 1995). Yet another approach is based on the use of *Constraint Logic Programming* (CLP), which subsumes Logic Programming and allows for *interpretation* of structures, notably in domains $\mathbb{R}$ and $\mathbb{N}$ (Jaffar & Lassez 1987). Learner *ICP* (for *Inductive Constraint Programming*) was described in an earlier work (Sebag & Rouveirol 1996); it basically uses constraints to prevent negative examples from matching candidate hypotheses. Since all possible (generally numerous) matchings between a hypothesis and a negative example must be examined, *ICP* therefore fully faces with the combinatorial complexity inherent to First Order based languages.

The complexity issue is addressed in the ILP literature by using either search biases or syntactic biases; e.g., *FOIL* considers one literal at a time (Quinlan 1990); *FOCL* allows a limited amount of look-ahead (Pazzani & Kibler 1992); *GOLEM* and *PROGOL* respectively set restrictions like *ij*-determinacy or maximal number of literals on the candidate hypotheses (Muggleton & Feng 1990; Muggleton 1995); restrictions on the matching space are also possible (Zucker & Ganascia 1994).
Adjusting these biases requires a precise *a priori* knowledge, which is far from always available.

This is the reason why we preferred a versionspace-like approach, where the only bias is that of the hypothesis language (Mitchell 1982). Precisely, our induction goal is to characterize the *whole* set of hypotheses that are consistent with the training examples and that cover at least one example, denoted *Th*. In this sense, this approach is "bias free"; the only problem is that the explicit characterization of *Th* is intractable, even in a propositional language (Haussler 1988). We therefore developed an approach based on an implicit characterization of *Th*, which is of polynomial complexity in attribute-value languages (Sebag 1996). This approach was extended to ILP (Sebag 1994a; Sebag & Rouveirol 1994) and then CLP (Sebag & Rouveirol 1996) but still suffered from the exponential complexity typical of first order logic. The present paper describes how an approximate characterization of *Th* can be obtained *with polynomial complexity* in a CLP language.

From: Proceedings of the Third International Conference on Multistrategy Learning. Copyright © 1996, AAAI (www.aaai.org). All rights reserved.

This approach is illustrated on the mutagenesis problem which comes from the field of organic chemistry (King, Srinivasan, & Sternberg 1995). The shortcoming of *ICP* on this problem comes from the number of possible matchings between a hypothesis and an example of molecule, which is exponential in the number of atoms in the molecule (up to 40). This drawback is sidestepped by sampling, rather than exhaustively exploring, the set of such matchings: only a restricted number of samples is considered during induction. These matching samples are produced by a stochastic goal-driven mechanism. The hybrid learner obtained by embedding this sampling mechanism in *ICP*, termed *STILL* for *Stochastic Inductive Logic Learner*, constructs a theory that is only ensured to be approximately discriminant, since it only explores a subset of the matching space. But it does so with a polynomial computational complexity in terms of the size of the training set.

In order for this paper to be self contained, *ICP* is first briefly described. The complexity issue is examined, and we present a goal-driven mechanism that samples the matching space explored by induction. This stochastic sampling mechanism yields tractable algorithms for approximate induction and classification. An experimental validation of *STILL* on the mutagenicity testbed is last presented; our results are compared to those of *FOIL* and *PROGOL*, reported from (Srinivasan, Muggleton, & King 1995).

ICP and Mutagenicity

This section illustrates our approach of induction on the mutagenicity problem which is one well-known testbed in ILP. A detailed description of *ICP*, and a discussion about induction in Constraint Logic Programming can be found in (Sebag & Rouveirol 1996).

Data and language of examples

The mutagenicity problem consists in discriminating organic molecules (nitroaromatic compounds) with high mutagenic activity from other organic molecules. This still open problem is of utmost practical interest, for these compounds occur in car exhaust fumes, and high mutagenicity is considered carcinogenic.

The basic description of molecules, referred to as background knowledge $\mathcal{B}_1$ in (Srinivasan & Muggleton 1995), includes the description of the molecule atoms and the bonds between these atoms. As the description of atoms involves numbers (partial electric charge), "numerical knowledge" (e.g., definition of predicate `less-than`) was added to background knowledge $\mathcal{B}_1$, to form background knowledge $\mathcal{B}_2$. Additional information is given via five attributes measuring the hydrophobicity of the molecule, the energy of the molecule lowest unoccupied molecular orbital, and so on. Background knowledge $\mathcal{B}_3$ stands for $\mathcal{B}_2$ augmented by this non-structural description. There exists a still more sophisticated description of the molecules (background knowledge $\mathcal{B}_4$), that takes into account elementary structural chemical concepts (e.g. methyl or benzene groups); but it will not be considered in this paper.

A molecule a is thus described by a clause, an excerpt of which is:

$$tc(a) :- \; atom(a, a_1, carbon, 22, -0.138), \ldots, atom(a, a_{26}, oxygen, 40, -0.388), \ldots bond(a, a_1, a_2, 7), \ldots, bond(a, a_{24}, a_{26}, 2), logp(a, 4.23), lumo(a, -1.246), \ldots$$

where tc stands for the target concept satisfied by a, here *active* or *inactive*.

Literal $atom(a, a_1, carbon, 22, -0.138)$ states that in compound a, atom a_1 is a carbon, of type 22, with partial charge -0.138. Literal $bond(a, a_1, a_2, 7)$ expresses that there exists a (unique) bond between atoms a_1 and a_2 in a, the type of which is 7.

This problem thus typically involves numerical and relational features.

Overview of *ICP*

ICP is a divide-and-conquer algorithm in the line of the *AQ* algorithms (Michalski 1983; Michalski *et al.* 1986): it successively generalizes one example Ex, termed *seed*, against all examples not satisfying the same target concept as Ex, termed *counter-examples* to Ex, noted $Ce_1, \ldots, Ce_n$.

ICP differs from other divide-and-conquer algorithms in two respects. First, most authors (Michalski 1983; Muggleton 1995) restrict themselves to learning one target concept (e.g. the *activity*), while *ICP* learns both the target concept and its negation. This hopefully allows the effects of noisy positive and negative examples to counterbalance each other.

Second, *ICP* considers *all* training examples instead of pruning the examples covered by previous hypotheses. The problem with pruning is that it induces an additional bias (the eventual theory depends on the choice of seeds) while increasing the overall complexity of induction (this will be detailed further).

Another key difference between *ICP* and all other learners, as far as we know, is that *ICP* does involve neither search biases nor syntactic biases: it aims at characterizing the set *Th(Ex)* of all hypotheses consistent with $Ex.Ce_1, \ldots Ce_n$, i.e. the Version Space discriminating Ex from any Ce_i (Mitchell 1982).

In opposition, *FOIL* (Quinlan 1990), *FOCL* (Pazzani & Kibler 1992) and *PROGOL*, among others, aim at finding "the" best hypothesis covering a training example, according to the more or less greedy optimization of a numerical criterion (quantity of information for *FOIL* and *FOCL*, MDL principle for *PROGOL*). To a lesser extent, *ML-Smart* (Bergadano & Giordana 1990; Botta & Giordana 1993) and *REGAL* (Giordana & Saitta 1993) also look for concise theories.

From: Proceedings of the Third International Conference on Multistrategy Learning. Copyright © 1996, AAAI (www.aaai.org). All rights reserved.

Formally, *ICP* builds a theory *Th* which is the disjunction of the versionspaces *Th(Ex)* for Ex ranging over the training set. *Th(Ex)* includes all hypotheses that cover Ex and discriminate counter-examples to Ex; it is given by the conjunction of the set of hypotheses $D(Ex, Ce)$ that cover Ex and discriminate Ce, for Ce ranging over the counter-examples to Ex:

Disjunctive Version Space Algorithm

$Th = false.$
For each Ex **training example**
  $Th(Ex) = True$
  For each Ce **counter-example to** Ex
    Build $D(Ex, Ce)$ *(see below)*
    $Th(Ex) = Th(Ex) \wedge D(Ex, Ce)$
  $Th = Th \vee Th(Ex).$

Language of hypotheses

The elementary step of *ICP* consists in characterizing the set of hypotheses $D(Ex, Ce)$ that cover the current seed Ex and discriminate a counter-example Ce to the seed.

Consider two examples of (simplified and arbitrary) molecules:

Ex : active(ex) :- *atom(ex, a, oxygen, -3.38), atom(ex, b, carbon, 1.24).*
Ce : inactive(ce) :- *atom(ce, c, carbon, -2.75), atom(ce, d, oxygen, 0.33).*

Example Ex is decomposed as $C\theta$, where C stores the structural information of Ex, i.e., that Ex is an *active* molecule having two atoms:

$C : active(X) : -atom(X', Y, Z, T), atom(X'', U, V, W)$

and θ carries all other information in Ex:

$$\theta = \{ X/ex,\ X'/ex,\ X''/ex,\ Y/a,\ Z/oxygen,\ T/-3.38,\ U/b,\ V/carbon,\ W/1.24\}$$

This decomposition allows induction to simultaneously explore two search spaces:

- The space of logical clauses generalizing C. Exploring this space is fairly simple since all variables in C are distinct: the only way to generalize C is by dropping literals.
- In what regards θ, we take advantage of the fact that θ, and more generally any substitution on C, are special cases of constraints on C. Variable grounding $\{X/v\}$ amounts to domain constraint $(X = v)$, which is analogous to a selector (Michalski 1983). And variable linking $\{X/Y\}$ amounts to binary constraint $(X = Y)$. *ICP* then explores the set of constraints on C generalizing θ. (See (Jaffar & Lassez 1987) for a comprehensive presentation of CLP).

Constructing $D(Ex, Ce)$ amounts to finding out all pairs (C, ρ), where C generalizes C (i.e. describes a molecule satisfying the same target concept as Ex and including at most the same number of atoms and bonds) and ρ is a constraint on C that generalizes θ: furthermore, C and ρ must be such that $C\rho$ discriminates Ce.

In the general case, discrimination can be based on predicates: if C involves a predicate that does not appear in Ce, C does not generalize Ce. Predicate-based discrimination amounts to simple boolean discrimination: presence/absence of a predicate (Sebag & Rouveirol 1996). It is not further considered in this paper since all molecules, whatever the target concept they satisfy, are described by the same predicates (*atom*, *bond*,..): predicate-based discrimination thus does not apply here.

Constraint-based discrimination takes place when the body of C (or of the current hypothesis) generalizes that of Ce. Then there exists at least one substitution σ matching the body of C with the body of Ce, called ***negative substitution.***
For instance, such a substitution σ respectively maps the first and second atoms in C onto the first and second atoms in Ce:

$$\sigma = \{ X/ce,\ X'/ce,\ X''/ce,\ Y/c,\ Z/carbon,\ T/-2.75,\ U/d,\ V/oxygen,\ W/0.33\}$$

But if such a σ exists, C is inconsistent: its body generalizes the bodies of both Ex and Ce, which yet satisfy opposite target concepts by definition.

Constraint-based discrimination prevents such inconsistencies by specializing C: it adds "conditions" (that is, constraints) to the body of C such that negative substitution σ does not satisfy these "conditions". For instance, constraint

$$\rho = (V = carbon)$$

is incompatible with σ, since $V.\sigma = oxygen$. By the way, ρ must also generalize the substitution θ derived from Ex, in order for $C\rho$ to still generalize Ex. For instance, constraint

$$\rho' = (V = hydrogen)$$

is also incompatible with σ, but $C\rho'$ does not generalize Ex any more. A formal presentation of constraint entailment and generalization order will be found in (Jaffar & Lassez 1987); roughly, constraint ρ_1 generalizes ρ_2 (equivalently, ρ_2 *entails* ρ_1) iff all substitutions satisfying ρ_2 also satisfy ρ_1.

Note that building constraints that generalize θ and are incompatible with a negative substitution σ amounts to an attribute value discrimination problem. This is particularly clear if we restrict our language of constraints to domain constraints (of the form

From: Proceedings of the Third International Conference on Multistrategy Learning. Copyright © 1996, AAAI (www.aaai.org). All rights reserved.

$(X = \mathcal{V})$, where $\mathcal{V}$ is a subset of the domain of X). This is also true when binary logical and arithmetic constraints are considered (e.g. $(X \neq Y)$. $(Z < T + 10)$, $(S > U - 20)$), by introducing auxiliary variables (this point is detailed in (Sebag & Rouveirol 1996)). However, binary constraints will not be further considered here, for two reasons. First of all, introducing binary constraints does not significantly modify the complexity of induction (it only affects its polynomial part), which is our primary concern in this paper. Second, unary constraints turned out to be sufficient to reach a good level of predictive accuracy on the mutagenesis problem.

Finally, our language of constraints is restricted to unary constraints of the form $(X = \mathcal{V})$, where

- $\mathcal{V}$ is an interval if X is a real or integer-valued variable;
- $\mathcal{V}$ is a value if X is a nominal variable.

Characterizing $D(Ex, Ce)$

Let us assume first that there exists a single negative substitution σ derived from Ce.

For any variable X, let $\rho_{X,\sigma}$ denote the maximally general constraint on variable X that generalizes θ and is incompatible with σ. Of evidence, $\rho_{X,\sigma}$ exists iff $X.\theta$ and $X.\sigma$ are distinct constants. When it is the case, and when X is an integer or real-valued variable, $\rho_{X,\sigma}$ is $(X = \mathcal{V})$ where $\mathcal{V}$ is the maximum interval including $X.\theta$ and excluding $X.\sigma$. And in case X is a nominal variable, $\rho_{X,\sigma}$ is $(X = X.\theta)$.
In our toy example, $(T =]-\infty, -2.75))$ (or for the sake of simplicity $(T < -2.75)$), is the most general constraint on T that generalizes θ and is incompatible with σ; similarly, $(Z = oxygen)$ is the most general constraint on Z (in our constraint language) that generalizes θ and is incompatible with σ.

Let ρ_σ denote the disjunction of $\rho_{X,\sigma}$ for X ranging over the variables in $\mathcal{C}$ such that $X.\theta \neq X.\sigma$. One easily shows that any constraint generalizing θ discriminates σ iff it entails (is generalized by) ρ_σ. A clause $\mathcal{C}\rho$ therefore belongs to $D(Ex, Ce)$ iff two conditions hold: $\mathcal{C}\rho$ must generalize Ex and ρ must entail ρ_σ.
If σ were the only negative substitution on $\mathcal{C}$ derived from Ce, any clause $\mathcal{C}\rho$ generalizing Ex such that ρ entails the disjunction

$$(Z = oxygen) \vee (T < -2.75) \vee (V = carbon) \vee (W > .33)$$

would discriminate Ce.
But consider the negative substitution σ' mapping the first atom in $\mathcal{C}$ onto the second atom in Ce, and the second atom in $\mathcal{C}$ onto the first atom in Ce:

$$\sigma' = \{ \begin{array}{l} X/ce,\ X'/ce,\ X''/ce, \\ Y/d,\ Z/oxygen,\ T/0.33, \\ U/c,\ V/carbon,\ W/-2.75, \} \end{array}$$

Of evidence, constraint $\rho = (Z = oxygen)$ is quite compatible with σ'; hence $\mathcal{C}\rho$ is inconsistent. This shows that a discriminant constraint ρ must be incompatible with *all* negative substitutions derived from Ce, in order for $\mathcal{C}\rho$ to be consistent with Ce.

In the general case, let $\Sigma_{Ex,\ Ce}$ be the set of negative substitutions on $\mathcal{C}$ derived from Ce. The previous result extends as (Sebag & Rouveirol 1996):

Proposition 1. $\mathcal{C}\rho$ belongs to $D(Ex, Ce)$ iff $\mathcal{C}\rho$ generalizes Ex and ρ entails ρ_σ for all σ in $\Sigma_{Ex,\ Ce}$.

To sum up, $D(Ex, Ce)$ is computationally described by $\mathcal{C}$ and the set of constraints $\{\rho_\sigma\ s.t.\ \sigma \in \Sigma_{Ex,\ Ce}\}$. The question of explicitly characterizing $D(Ex, Ce)$ and then the whole theory *Th* from this computational characterization was addressed in (Sebag & Rouveirol 1996) and will not be considered in this paper. Rather, we focus on using the computational description of $D(Ex, Ce)$ in order to classify further instances of the problem domain.

As a matter of fact, this computational description enables one to check whether an unseen instance E is covered by a hypothesis in $D(Ex, Ce)$, or, for short, *belongs to* $D(Ex, Ce)$. It is shown (Sebag & Rouveirol 1996) that:

Proposition 2. E belongs to $D(Ex, Ce)$ iff E can be expressed as $C\tau$, where C generalizes $\mathcal{C}$ and τ entails ρ_σ for all σ in $\Sigma_{Ex,\ Ce}$.

Classifying further examples

The important point in the above result, is that it allows one to classify unseen instances of the problem domain:

- From Proposition 2, one can compute whether any given instance E is covered by a hypothesis in *Th(Ex)*, or, for short, *belongs to Th(Ex)*: E belongs to *Th(Ex)* iff E belongs to $D(Ex, Ce)$, for all Ce counter-example to Ex.
- Knowing whether E belongs to *Th(Ex)* for all training examples Ex gives means to classify E, via a nearest neighbor-like process. Let E be termed *neighbor* of Ex if E belongs to *Th(Ex)* ; the class of E can thereafter be determined by a majority vote among its neighbors in the training set. (see (Sebag 1996) for a discussion about the advantages of this nearest neighbor-like classification process).

Neighbor (E, Ex) : (E *belongs to* Th(Ex))

```
For each Ce counter-example to Ex
  if NOT Belongs(E, D(Ex,Ce))
    return false
return true
```

From: Proceedings of the Third International Conference on Multistrategy Learning. Copyright © 1996, AAAI (www.aaai.org). All rights reserved.

Checking whether E belongs to $D(Ex, Ce)$ is computed as follows. Let $\Sigma_{Ex,\ E}$ denote the set of substitutions on $\mathcal{C}$ matching E. Then:

Belongs(E, $D(Ex, Ce)$)

```
For each τ in Σ_{Ex, E}
  If τ entails ρ_σ for all σ in Σ_{Ex, Ce},
    return true.
return false.
```

Simply put, *ICP* rather constructs an oracle than an explicit theory. This oracle achieves the classification of further examples; it is made of theory Th, stored as the list of

$$D(Ex_i, Ex_j) = (\mathcal{C}_i, \{\rho_\sigma \ s.t. \ \sigma \in \Sigma_{Ex_i, Ex_j}\})$$

for Ex_i in the training set and Ex_j counter-example to Ex_i; and this theory is interpreted according to Proposition 2. Actually, the classifier constructed by *ICP* consists of Th and of the standard nearest neighbor algorithm, calling the above *Neighbor* boolean function.

Complexity

Under the standard assumption that the domain of any variable is explored with a bounded cost, the complexity of building ρ_σ is linear in the number of variables in $\mathcal{C}$. Let $\mathcal{X}$ and $\mathcal{S}$ respectively denote upper-bounds on the number of variables in $\mathcal{C}$ and on the number of substitutions in Σ_{Ex_i, Ex_j}. The characterization of $D(Ex_i, Ex_j)$ is then in $\mathcal{O}(\mathcal{X} \times \mathcal{S})$.

Let N be the number of training examples. Since all $D(Ex_i, Ex_j)$ must be characterized, the complexity of learning in *ICP* is

$$\mathcal{O}(N^2 \times \mathcal{X} \times \mathcal{S})$$

And, since checking whether an instance E belongs to $Th(Ex)$ requires to consider all substitutions τ on $\mathcal{C}$ (with $Ex = \mathcal{C}.\theta$) matching E, the number of which is upper-bounded by $\mathcal{S}$, the complexity of classification in *ICP* is

$$\mathcal{O}(N^2 \times \mathcal{X} \times \mathcal{S}^2)$$

In particular, since pruning a training example requires to check whether it is covered by a previous theory $Th(Ex)$, the complexity of learning increases up to $\mathcal{O}(N^3 \times \mathcal{X} \times \mathcal{S}^2)$ if *ICP* follows a standard divide-and-conquer strategy.

The crux of complexity lies in factor $\mathcal{S}$: in the simple case of molecules with two atoms, $\mathcal{S}$ is 2^2 (any atom in $\mathcal{C}$ can match any atom in Ce). This makes the *ICP* algorithm intractable in the mutagenesis problem, where molecules involve up to 40 atoms: $\mathcal{S}$ then amounts to 40^{40}...

Polynomial Approximate Learning

ICP suffers from two major drawbacks: first, it is intractable for medium-sized truly relational problems, as shown above. Second, it basically stems from the Version Space framework, and therefore ill-prepared to deal with noisy and sparse data.

The tractability limitation is first addressed via a stochastic bias: the idea consists in sampling, rather than exhaustively exploring, the set of substitutions $\Sigma_{Ex,Ce}$. We again illustrate the stochastic sampling mechanism on the mutagenesis problem.

Second, two heuristics, taken from the propositional version of *ICP* (Sebag 1996), are used to relax the standard consistency and generality requirements of Version Spaces, and cope with noise and sparseness.

Stochastic Bias

Let us examine the structure of examples in the mutagenesis problem. Note that the semantics of a molecule is not modified by changing the identifiers of the atoms (nominal constants), provided the change is consistent.

These identifiers can thus be arbitrarily set to $1, 2, \ldots, n$, if n denotes the number of atoms in Ex. A negative substitution σ on $\mathcal{C}$ is completely defined by associating each atom i in $\mathcal{C}$ to an atom in Ce, noted $\sigma(i)$ by abuse of notations. The intractability comes from the fact that, if Ce involves n' atoms, the number of such negative substitutions is in $n'^{\,n}$.

Basically, discriminating σ from θ requires to

- discriminate at least one atom i in Ex from atom $\sigma(i)$ in Ce; or
- discriminate at least one bond in Ex, linking atoms i and j, from the bond in Ce linking atoms $\sigma(i)$ and $\sigma(j)$, if such a bond exists.

The more "similar" atoms i in Ex and $\sigma(i)$ in Ce, the more difficult it is to discriminate them, and the more informative the negative substitution σ is: this notion parallels that of near-misses in attribute-value languages. Formally, a partial order can be defined on the substitutions in $\Sigma_{Ex,Ce}$, and it is shown that non-minimal substitutions can soundly be pruned by discriminant induction (Sebag 1994a; Sebag & Rouveirol 1994): this pruning is analogous to the pruning of non near-misses examples in the propositional case (Smith & Rosenbloom 1990; Sebag 1994b). Unfortunately, building the set of such minimal substitutions turns out to be intractable too.

Another possibility is to consider *only one* substitution σ, defined as minimizing some distance to θ, in the line of the structural similarity developed in (Bisson 1992). The "optimal" substitution in $\Sigma_{Ex,Ce}$ would thus minimize the sum of the distances between atom i in Ex and atom $\sigma(i)$ in Ce, plus the sum of the distances between bonds $i-j$ in Ex and bonds $\sigma(i)-\sigma(j)$ (if such a bond exists) in Ce. The distance between any two atoms makes no problem: as noted by (Srinivasan & Muggleton 1995), the description of an atom can be

From: Proceedings of the Third International Conference on Multistrategy Learning. Copyright © 1996, AAAI (www.aaai.org). All rights reserved.

handled as a single tree-structured feature since the element of an atom commands its atom type and its atom type commands its charge.

However, using such an optimization approach to determine which substitution to consider in $\Sigma_{Ex,Ce}$ rises two problems: first of all, we feel that a single substitution, even optimal, cannot be representative of the whole set $\Sigma_{Ex,Ce}$; second, the optimization routine in itself can be computationally expensive.

Finally, we decided to consider several substitutions, the number of which to be supplied by the user. These substitutions are not purely random: as stated above, substitutions nearer to θ should be preferred. When constructing a substitution σ, one thus tries to associate to any atom i in Ex the atom j in Ce which is most similar to i, provided that j is not already associated to another atom in Ex. A substitution σ constructed this way constitutes a local optimum with respect to the above minimization problem.
Currently, the sampling mechanism of the substitutions in $\Sigma_{Ex,\ Ce}$, where Ex and Ce respectively includes n and n' atoms, is implemented as follows:

Select σ **in** $\Sigma_{Ex,\ Ce}$

while possible
  Select i **in** $\{1,\ldots,n\}$ **not yet selected**
    Select j **in** $\{1,\ldots,n'\}$ **not yet selected**
      such that atom j **in** Ce **is as close as possible to atom** i **in** Ex,
      Do $\sigma(i) = j$.

Index j is deterministically selected depending on i: atom j in Ce has same electric charge as atom i in Ex, if possible; otherwise, it has same atom type; otherwise, it is of same element.

Index i is stochastically selected with uniform probability in $\{1,\ldots,n\}$. This way, any atom i in Ex will in average be associated to a similar atom in Ce, provided the sampling mechanism is run a sufficient number of times.

More precisely, the above stochastic sampling mechanism ensures that a set of samples captures an arbitrarily precise representation of $\Sigma_{Ex,\ Ce}$ with high probability, provided the number of samples allowed is "sufficient". Further work is concerned with formalizing this intuition, as well as improving the selection mechanism via taking into account also the bonds between atoms.

Overview of *STILL*

The *STILL* algorithm combines the general approach of *ICP* and the above sampling mechanism. This stochastic bias is used to make both induction and classification tractable.

Approximate Learning. Consider the building of the set of hypotheses $Th(Ex)$ that cover Ex and are consistent. Instead of exploring the whole sets of substitutions $\Sigma_{Ex,\ Ce}$ for Ce ranging over the counter-examples to Ex, *STILL* only processes η substitutions, where η is a positive integer supplied by the user. To give an order of idea, η was set to 300 in our experiments on the mutagenesis problem.
This way, it constructs a set of hypotheses $Th_\eta(Ex)$ that cover Ex and are only partially ensured to be consistent, since only sampled substitutions are surely discriminated.

Concretely, the set of hypotheses $Th_\eta(Ex)$ is characterized as follows. Let n be the number of counter-examples to Ex. For each counter-example Ce, $\frac{\eta}{n}$ samples of substitutions are selected in $\Sigma_{Ex,\ Ce}$. The computational description of $Th_\eta(Ex)$ consists of clause C and the set $\mathcal{R}$ of discriminant constraints ρ_σ, corresponding to the η sampled substitutions σ derived from all counter-examples. This way, the number of constraints in $\mathcal{R}$ does not depend on whether Ex belongs to the majority or the minority class. Otherwise, the number of constraints in $\mathcal{R}$ would be much higher, and hence $Th_\eta(Ex)$ much more specific for the minority class than for the majority class. This heuristics was adopted for reasons of empirical accuracy: experimentally, this makes a difference in the mutagenesis problem, where examples are typically two active to one inactive.

Characterize $Th_\eta(Ex)$:

$\mathcal{R} = \phi$.
n = **Number of counter-examples to** Ex
For Ce **counter-example to** Ex
  For $j = 1 \ldots \frac{\eta}{n}$,
    Select σ in $\Sigma_{Ex,\ Ce}$,
    Build ρ_σ
    Do $\mathcal{R} = \mathcal{R} \bigcup \{ \rho_\sigma \}$
return $(C, \mathcal{R})$.

The disjunction Th_η of theories $Th_\eta(Ex)$ for Ex ranging over the training set, is termed *approximate theory*: the rate of approximation is the number η of allowed samples. Note that Th_η is more general than Th; it tends toward Th as η increases.

Approximate classification. The classification process in *ICP* is based on checking whether the instance E to classify is neighbor of Ex, i.e. belongs to $Th(Ex)$, for all training examples Ex. In order to do so, it still explores the set $\Sigma_{Ex,\ E}$ of substitutions on C (where $Ex = C\theta$), matching E. The size of this set similarly makes classification intractable.

This limitation is addressed in *STILL* via the sampling mechanism too: instead of exhaustively exploring $\Sigma_{Ex,\ E}$, *STILL* only considers a fixed number K of substitutions in this set, where K is a positive inte-

From: Proceedings of the Third International Conference on Multistrategy Learning. Copyright © 1996, AAAI (www.aaai.org). All rights reserved.

ger supplied by the user. To give an order of idea, K was set to 3 in our experiments on the mutagenesis problem.

The *Neighbor* function is therefore modified as:

Approx_Neighbor (E, Ex) :

$(\mathcal{C}, \mathcal{R})$ = Characterize $Th_\eta(Ex)$
For i = 1...K
  Select τ in $\Sigma_{Ex,E}$
    If τ entails all ρ in $\mathcal{R}$
      return *true*
return *false*

As soon as a substitution τ among K samples of substitutions in $\Sigma_{Ex,E}$ entails all discriminant constraints in the characterization of $Th_\eta(Ex)$, E is considered an approximate neighbor of Ex.

Note the above function corresponds to an "interpretation" of $Th_\eta(Ex)$ that is more specific than $Th_\eta(Ex)$ itself; this overspecificity decreases as K increases.

Parameter K controls the number of trials allowed to get an answer from theory Th_η; metaphorically speaking, K corresponds to the "patience" of the constructed expert.

Coping with noisy and sparse examples

$Th(Ex)$ (which is the theory $Th_\eta(Ex)$ tends toward as η increases) includes consistent hypotheses only, and maximally general consistent hypotheses in particular. No doubt this approach is ill-suited to real-world datasets: when erroneous examples are encountered, strictly consistent hypotheses have few predictive accuracy (Clark & Niblett 1987). And when examples are sparse, maximally general consistent hypotheses are too general: most instances come to be covered by a hypothesis in most $Th_\eta(Ex_i)$, and therefore get unclassified, or classified in the majority class.

These limitations were already encountered in the attribute-value version of *ICP*, and have been addressed by two heuristics (Sebag 1996), which simply extend to first-order logic owing to the computational characterization of the constructed theory.

The first one addresses the presence of noise in the data, by allowing one to relax the consistency requirement. The originality consists in relaxing this requirement and allowing a limited number of inconsistencies ***during the classification of an instance rather than during induction.*** In opposition, the number of acceptable inconsistencies in *PROGOL* for instance is set before starting induction; and if one wants to modify this bias, s/he must restart induction from scratch.

This is done as follows. By definition, E belongs to $Th(Ex)$ and is considered as neighbor of Ex iff it belongs to $D(Ex, Ce)$ for all Ce counter-example to Ex. This definition is simply relaxed as: E is from now on considered as neighbor of Ex iff it belongs to $D(Ex, Ce)$ for all Ce counter-example to Ex, but at most ε of them, where ε is a positive integer supplied by the user. Of evidence, the greater ε, the more general the interpretation of $Th(Ex)$ is.

The second heuristics addresses the sparseness of the data, by allowing one to increase the specificity of the produced theory. This is done at the level of the discriminant constraints ρ_σ. By construction, ρ_σ is the maximally general constraint that discriminates σ and generalizes θ; it is the disjunction over the variables X in $\mathcal{C}$, of domain constraints $\rho_{X,\sigma}$. A given substitution τ entails ρ_σ iff there exists at least one variable X such that $X.\tau$ satisfies $\rho_{X,\sigma}$.

The specificity of the theory is simply modified by interpreting now ρ_σ as a M-of-N concept: from now on, substitution τ is considered to entail ρ_σ iff there exists at least M variables X such that $X.\tau$ satisfies $\rho_{X,\sigma}$, where M is a positive integer supplied by the user.

And the greater M, the more specific the interpretation of $Th(Ex)$ is.

Note that theory Th_η does not depend in any way on the values of parameter ε or M. In particular, *STILL* requires no *a priori* knowledge regarding the rate of noise and representativity of the data. Parameters M and ε can be adjusted from the experimental classification results — but with no need to restart induction. See (Sebag 1996) for a discussion about the advantages of such *a posteriori* biases.

Complexity

As expected, the complexity of *STILL* is much more affordable than that of *ICP*.

Let $\mathcal{X}$ still denote an upper-bound on the number of variables in $\mathcal{C}$. The complexity of building ρ_σ is still linear in $\mathcal{X}$. The cost of selecting σ is quadratic in $\mathcal{X}$ (this is a large over-estimation). Hence, the complexity of learning $Th_\eta(Ex)$ is in $\mathcal{O}(\mathcal{X}^3 \times \eta)$. Finally, the computational complexity of induction in *STILL* is linear in the rate of approximation and in the number of training examples, and cubic in the number of variables in one example:

$$\mathcal{O}(N \times \mathcal{X}^3 \times \eta)$$

In the mutagenicity problem, N is 188, $\mathcal{X}$ is less than 200. The rate of approximation η was set to 300, to be compared with the typical size of a set $\Sigma_{Ex,\ Ce}$, that is 30^{30}.

The complexity of classification is that of learning, increased by factor K (which was set to 3 in our experiments):

$$\mathcal{O}(N \times \mathcal{X}^3 \times \eta \times K)$$

Note that the heuristics designed to cope with noise and sparseness do not modify the computational complexity of classification.

From: Proceedings of the Third International Conference on Multistrategy Learning. Copyright © 1996, AAAI (www.aaai.org). All rights reserved.

Experimentation

This section presents an experimental validation of our approximate learning and classification scheme on the mutagenicity problem, which is presented in details in (Srinivasan & Muggleton 1995).

The data

The experimentations reported in this section consider the data set composed of 188 compounds, described via the background knowledge $\mathcal{B}_1$ (including the description of atoms and bonds in the molecules), $\mathcal{B}_2$ ($\mathcal{B}_1$ augmented with definitions of numerical inequalities), and $\mathcal{B}_3$ ($\mathcal{B}_2$ augmented with five non structural attributes).

For all experiments that follow, the atoms in relevant background theories are partitioned in 188 ground clauses, each clause describing all information relevant to a given compound.

The reference results obtained by *PROGOL* and *FOIL* on this problem (reported from (Srinivasan & Muggleton 1995) and personal communication from A. Srinivasan), are:

Background knowledge	Accuracy	
	FOIL	*PROGOL*
$\mathcal{B}_1$	60 ± 4	76 ± 3
$\mathcal{B}_2$	81 ± 3	81 ± 3
$\mathcal{B}_3$	83 ± 3	83 ± 3

Table 1: Results of FOIL *and* PROGOL *on the 188-compound problem: Average predictive accuracy on the test set*

Experimental Settings

The parameter η used to ensure the tractability of induction (rate of approximation) is set to 300. The parameter K used to ensure the tractability of classification is set to 3.

Parameter M used to control the specificity of the theory varies from 1 to 10. Parameter ε, used to control the consistency of the theory, varies from 0 (strictly consistent hypotheses only) to 15%. The consistency of a hypothesis is from now on defined in term of *percentage* of inconsistencies, rather than in term of *number* of inconsistencies.

All results are averaged over 15 independent runs, where each run consists in learning from 90% of the 188 compounds (randomly selected such that the ratio of active/inactive compounds in the training set is same as in the global data, i.e. about two to one) and classifying the remaining 10% of the data. This protocol of validation is similar to the ten-fold cross validation used in (Srinivasan & Muggleton 1995): the number of runs is only slightly increased (from 10 to 15), as suggested for stochastic approaches (Kinnear 1994).

STILL is written in C^{++}. The run-time on HP-735 workstations (including the construction of the theory and the classification of the test examples), varies from 60 to 120 seconds. Despite the difference in the language of implementation (Language C for *FOIL* and Prolog for *PROGOL*), this demonstrates that *STILL* is two or three orders of magnitude faster than *FOIL* and *PROGOL* (respectively 5 000 and 117 000 seconds on HP-735 when handling background knowledge $\mathcal{B}_1$).

Experiments with $\mathcal{B}_1$

STILL is first experimented in the context of background knowledge $\mathcal{B}_1$; this means that variables describing the atom type and the partial electric charge of atoms are handled as nominal variables instead of respectively integer and real-valued variables. With those settings, *STILL* can only learn discriminant *instantiations* of those variables (e.g., $(Charge_i = .84)$). Results of these experiments are therefore to be compared to results of *FOIL* and *PROGOL* in the context of background knowledge $\mathcal{B}_1$.

Table 2 shows how the average predictive accuracy on the test set varies with ε and M (label *Accur*). It also gives the standard deviation of the accuracy, as well as the average percentages of unclassified and misclassified examples (label *Unclass* and *Misclass*). Examples happen to be unclassified either when they have no neighbor in the training set, or when the nearest-neighbor process ends up in a tie.

ε	M	$\mathcal{B}_1$			
		Accur.	Unclass.	Misclass.	Time
0	1	79 ± 2	1.19	19.8	69
	2	**84.5** ± 2	3.17	12.3	73
	3	77.8 ± 3	7.94	14.3	77
	4	76.2 ± 3	11.9	11.9	78
	5	68.3 ± 2	21	10.7	79
5	1	80.2 ± 2	0	19.8	64
	2	81 ± 2	0	19	68
	3	82.5 ± 1	1.98	15.5	72
	4	79.4 ± 2	1.59	19	75
	5	80.6 ± 3	2.78	16.7	77
10	1	73 ± 2	0	27	60
	2	79.4 ± 3	0.397	20.2	64
	3	83.3 ± 3	0	16.7	69
	4	78.2 ± 2	0.794	21	73
	5	75.8 ± 2	3.57	20.6	76
15	1	69.8 ± 0.7	0	30.2	59
	2	80.6 ± 2	0.794	18.7	63
	3	81 ± 2	0.397	18.7	68
	4	83.7 ± 2	0.794	15.5	72
	5	76.6 ± 2	0.794	22.6	74

Table 2: STILL, *Average predictive accuracy on the test set, with background knowledge* $\mathcal{B}_1$, $\eta = 300$, $K = 3$

Note that, as M increases, theories $Th_\eta(Ex)$ get more specific, hence any instance has less and less neighbors in the training set. This is shown as more and more

From: Proceedings of the Third International Conference on Multistrategy Learning. Copyright © 1996, AAAI (www.aaai.org). All rights reserved.

examples shift from *Accur* and *Misclass* to *Unclass* (especially for $\varepsilon = 0$).

Note also that for increasing values of ε, theories $Th_\eta(Ex)$ get more and more general, and the best predictive accuracy moves toward larger values of M: the overspecificity due to large values of M resists the overgenerality due to large values of ε.

The predictive accuracy reached for the best adjustment of parameters ε and M ($\varepsilon = 0$ and $M = 2$) is significantly better than the results achieved by *PROGOL* and *FOIL* for background knowledge $\mathcal{B}_1$. The question of automatically tuning the learning parameters nevertheless remains open.

Of course, a fair comparison would require to see how the predictive accuracy of *PROGOL* similarly varies depending on the maximum number of inconsistencies and the maximum number of literals in the clauses, which are respectively set to 5 and 4 in (Srinivasan & Muggleton 1995).

A tentative explanation of the differences between the performances of *PROGOL* and that of *STILL* on this particular problem is the following. *PROGOL* and *STILL* operate in similar search spaces and use different heuristics to the same end, namely constructing partially complete and partially consistent clauses. This suggests that the better performances of *STILL* are due to its inherent redundancy: *PROGOL* starts from some training examples and constructs the "best" hypothesis covering these examples, while *STILL* considers all training examples and constructs all approximately consistent hypotheses covering these examples. The redundancy of the constructed theory has frequently been viewed as a factor of robustness and reliability of the classifier: see (Gams 1989; Nok & Gascuel 1995) among others.

Experiments with $\mathcal{B}_2$

We then check the added value of using a CLP formalism: variables describing the atom type and electric charge of atoms are from now on handled as integer and real-valued variables. The only difference is that *STILL* can now learn discriminant domain constraints such as $(Charge_i > .143)$ or $(AtomType_i < 22)$, instead of simple discriminant instantiations $(Charge_i = .84)$ as before.

The results obtained here (Table 3) must thus be compared to those of *FOIL* and *PROGOL* in the context of background knowledge $\mathcal{B}_2$.

Unexpectedly, learning constrained clauses only results in a slight improvement of the overall predictive accuracy (from 84.5 to 86.5): allowing to set numerical inequalities on variables $Charge_i$ and $AtomType_i$ makes few difference. In retrospect, $\mathcal{B}_1$ mainly involves structural information and few distinct numerical values : as noted in (Kohavi 1995), the presence of numerical variables does not mean that a problem is basically numerical.

One side effect of allowing numerical inequalities, is that the best predictive accuracy is obtained for higher values of ε and M. This can be explained as follows: an inequality is more often satisfied than an equality. This implies that the theories constructed with constrained clauses containing inequalities are more general than those built with pure Horn clauses. Increasing the value of parameter M allows to resist this overgenerality, while increasing the value of ε aims at keeping a desirable level of generality.

ε	M	$\mathcal{B}_2$			
		Accur.	Unclass.	Misclass.	Time
0	4	81.5 ± 2	0	18.5	100
	5	83.3 ± 2	0	16.7	107
	6	**85.6** ± 2	0.37	14.1	113
	7	70.7 ± 2	0.741	28.5	122
	8	69.6 ± 3	2.59	27.8	128
5	4	73.7 ± 2	0	26.3	89
	5	81.1 ± 2	0.37	18.5	95
	6	81.9 ± 2	0.37	17.8	100
	7	83.7 ± 2	0	16.3	106
	8	76.3 ± 3	0.741	23	115
10	4	68.1 ± 0.6	0	31.9	88
	5	74.8 ± 2	0	25.2	93
	6	77.8 ± 2	0.741	21.5	98
	7	86.3 ± 2	0	13.7	105
	8	80.4 ± 2	0.37	19.3	109
15	5	70.8 ± 1	0.694	28.5	92
	6	77.4 ± 2	0.347	22.2	96
	7	79.9 ± 2	0.694	19.4	101
	8	**86.5** ± 2	0	13.5	108
	9	83.3 ± 2	0.741	15.9	115

Table 3: STILL, *Average predictive accuracy on the test set, with background knowledge* $\mathcal{B}_2$, $\eta = 300$, $K = 3$

Additional experiments show that the predictive accuracy only slightly increases with η: e.g. increasing η from 300 to 700 improves the best predictive accuracy by only one point; in counterpart, the computational cost is nearly twice as much as for $\eta = 300$.

Experiments with $\mathcal{B}_3$

As noted earlier, the relatively disappointing results of *STILL* with background knowledge $\mathcal{B}_2$ may be explained by the little number of distinct numerical constants involved in the description of atoms.

A third set of experiments has thus considered background knowledge $\mathcal{B}_3$, that is $\mathcal{B}_2$ enriched with (truly) numerical attributes.

The best results, obtained for $\varepsilon = 0$ and $M = 6$, are impressive (Table 4); as far as we know, they are significantly better than the results obtained by various learners using a superset of background knowledge $\mathcal{B}_3$, and in particular, by ILP learners having numerical skills (Karalic 1995).

From: Proceedings of the Third International Conference on Multistrategy Learning. Copyright © 1996, AAAI (www.aaai.org). All rights reserved.

Experiments with $\mathcal{B}_3$ also confirm the trend observed when experimenting with $\mathcal{B}_2$: the optimal accuracy moves toward higher values of M; the produced theory needs some extra specialization when it is allowed to include inequality constraints.

Again, the main question here appears that of adjusting automatically parameters ϵ and M.

ϵ	M	$\mathcal{B}_3$		
		Accur.	Unclass.	Misclass.
0	6	**93.4** ± 1	0.347	6.25
	7	88.5 ± 2	1.04	10.4
	8	89.9 ± 1	1.74	8.33
	9	88.9 ± 2	3.47	7.64
	10	88.9 ± 2	2.78	8.33
5	6	89.6 ± 2	0	10.4
	7	91 ± 1	0	9.03
	8	83 ± 2	0.694	16.3
	9	83.7 ± 2	0.694	15.6
	10	79.5 ± 2	0.694	19.8
10	6	85.8 ± 3	0	14.2
	7	84.7 ± 1	1.04	14.2
	8	90.3± 2	0.694	9.03
	9	85.8 ± 2	0.347	13.9
	10	77.8 ± 3	1.39	20.8
15	6	85.1 ± 2	0.347	14.6
	7	86.8 ± 2	0.347	12.8
	8	89.2 ± 2	0	10.8
	9	87.5 ± 2	1.04	11.5
	10	83.7 ± 2	0.694	15.6

Table 4: STILL, Average predictive accuracy on the test set, with background knowledge $\mathcal{B}_3$, $\eta = 300$, $K = 3$

Conclusion

We have experimentally demonstrated the potentialities of the stochastic approximate learner *STILL* for classification purposes.

The main interest of this work in our sense, is to show how stochastic processes can be engineered to cut into the combinatorial complexity pertaining to ILP. Note that this sampling mechanism is combined with, rather than replaces, induction. This is a strong difference with the genetic side of machine learning and ILP (Kovacic 1994; Wong & Leung 1995)[1]. More precisely, what is sampled here is related to examples rather than to solutions.

Another interest lies in the non-standard use of the Version Space framework: the computational representation of the constructed theory sidesteps the intrinsic combinatorial complexity of Version Spaces. Further, it allows one to relax at no additional cost the consistency and generality requirements, whenever this is required by the defects, noise and sparseness of the data. Last, the degrees of consistency and generality can be tuned with no need to modify the constructed theory, and hence without restarting induction.

The main weakness of our learning approach is that it constructs nothing like an intelligible theory[2]. Further work is concerned with pruning and compacting the inarticulate theory underlying the oracle; the challenge lies in providing a readable version of this theory *having same predictive accuracy*. The key question is that of the long debated trade-off between intelligibility and predictive accuracy.

This approach will also be given a learnability model, be it based on PAC-learnability (Valiant 1984) or U-learnability (Muggleton. 1994). In particular, in the *Probably Approximately Correct* (PAC) framework, parameter η used to sample the substitutions naively corresponds to the probability of getting the desired theory, the approximation of which is given by ϵ.

Acknowledgments. We gratefully thank S. Muggleton, A. Srinivasan and R. King, who formalized, studied and made available the mutagenicity problem: this nice problem was determinant for the orientation of the presented work. The work of the authors has been partially supported by the ESPRIT BRA 6020 Inductive Logic Programming and by the ESPRIT LTR 20237 *ILP*[2].

References

Bergadano. F., and Giordana, A. 1990. Guiding induction with domain theories. In Kodratoff, Y., and Michalski, R., eds., *Machine Learning : an artificial intelligence approach*, volume 3. Morgan Kaufmann. 474-492.

Bisson, G. 1992. Learning in FOL with a similarity measure. In *Proceedings of* 10^{th} *AAAI.*

Botta, M., and Giordana, A. 1993. Smart+ : A multistrategy learning tool. In *Proceedings of IJCAI-93.* 937 943. Morgan Kaufmann.

Clark, P., and Niblett, T. 1987. Induction in noisy domains. In Bratko, I., and Lavrac, N., eds., *Proceedings of EWSL-87*, 11-30. Sigma Press.

Gams, M. 1989. New measurements highlight the importance of redundant knowledge. In Morik, K., ed., *Proceedings of EWSL-89*, 71-80. Pitman, London.

Giordana, A., and Saitta, L. 1993. REGAL: An integrated system for learning relations using genetic algorithms. In *Proceedings of* 2^{nd} *International Workshop on Multistrategy Learning*, 234-249. Harpers Ferry.

Giordana, A.; Saitta. L.; and Zini, F. 1994. Learning disjunctive concepts by means of genetic algorithms.

[1] In most GA-based learners, solutions are sampled and evaluated from their predictive accuracy, i.e. through deduction only. An exception is the system described in (Giordana, Saitta, & Zini 1994), where the selection process is driven by inductive heuristics.

[2] Though the constructed oracle is *definitely not* a black box: for any classified instance, we can extract a set of clauses justifying this classification (Sebag 1995).

From: Proceedings of the Third International Conference on Multistrategy Learning. Copyright © 1996, AAAI (www.aaai.org). All rights reserved.
In Cohen, W., and Hirsh, H., eds., *Proceedings of ICML-94*, 96–104. Morgan Kaufmann.

Haussler. D. 1988. Quantifying inductive bias : AI learning algorithms and Valiant's learning framework. *Artificial Intelligence* 36:177–221.

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.

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

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

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

Kohavi, R. 1995. The power of decision tables. In Lavrac, N., and Wrobel, S., eds., *Proceedings of ECML-95*, 174–189. Springer-Verlag.

Kovacic, M. 1994. MILP: A stochastic approach to ILP. In Wrobel, S., ed., *Proceedings of ILP-94*.

Lavrac, N.; Dzeroski, S.; and Grobelnick, M. 1991. Learning non recursive definitions of relations with LINUS. In *Proceedings of EWSL'91*.

Michalski, R.; Mozetic, I.; Hong, J.; and Lavrac, N. 1986. The AQ15 inductive learning system: an overview and experiment. In *Proceedings of IMAL*.

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.

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

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

Muggleton, S., and Feng, C. 1990. Efficient induction of logic programs. In *Proceedings of the 1st conference on algorithmic learning theory*. Ohmsha, Tokyo, Japan.

Muggleton., S. 1994. Bayesian inductive logic programming. In Warmuth, M., ed., *Proceedings of COLT-94, ACM Conference on Computational Learning*, 3–11. ACM Press.

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

Nok, R., and Gascuel, O. 1995. On learning decision committees. In Prieditis, A., and Russell, S., eds., *Proceedings of ICML-95*, 413–420. Morgan Kaufmann.

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

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

Sebag, M., and Rouveirol, C. 1994. Induction of maximally general clauses compatible with integrity constraints. In Wrobel, S., ed., *Proceedings of ILP-94*.

Sebag, M., and Rouveirol, C. 1996. Constraint inductive logic programming. In de Raedt, L., ed.. *Advances in ILP*. IOS Press.

Sebag, M. 1994a. A constraint-based induction algorithm in FOL. In Cohen, W., and Hirsh, H., eds., *Proceedings of ICML-94*. M. Morgan Kaufmann.

Sebag, M. 1994b. Using constraints to building version spaces. In Raedt, L. D., and Bergadano, F., eds., *Proceedings of ECML-94*. Springer Verlag.

Sebag, M. 1995. 2^{nd} order understandability of disjunctive version spaces. In *Workshop on Machine Learning and Comprehensibility, IJCAI-95*. Report LRI, Université Paris-Sud.

Sebag, M. 1996. Delaying the choice of bias: A disjunctive version space approach. In Saitta, L., ed., *Proceedings of ICML-96*. Morgan Kaufmann.

Smith, B., and Rosenbloom, P. 1990. Incremental non-backtracking focusing: A polynomially bounded generalization algorithm for version space. In *Proceedings of AAAI-90*, 848–853. Morgan Kaufmann.

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.; and King, R. 1995. Comparing the use of background knowledge by two ILP systems. In *Technical Report, Oxford, UK*.

Valiant, L. 1984. A theory of the learnable. *Communication of the ACM* 27:1134–1142.

Wong, M., and Leung, K. 1995. Combining genetic programming and inductive logic programming using logic grammars. In Fogel, D. B., ed., *Second IEEE International Conference on Evolutionary Computation*, 733–736. IEEE Press.

Zucker, J.-D., and Ganascia, J.-G. 1994. Selective reformulation of examples in concept learning. In Cohen, W., and Hirsh, H., eds., *Proceedings of ICML-94*, 352–360. Morgan Kaufmann.