New Generation Computing, 17(1999)3-23
OHMSHA, LTD. and Springer-Verlag

Introductory Paper

Inductive Logic Programming for Relational Knowledge Discovery

Nada LAVRAČ and Sašo DŽEROSKI
J. Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
{nada.lavrac,saso.dzeroski}@ijs.si
Masayuki NUMAO
Tokyo Institute of Technology
2-12-1, Ookayama, Meguro-ku, Tokyo, 152 Japan
numao@cs.titech.ac.jp

Received 2 September 1998

Abstract Inductive logic programming (ILP) is concerned with the induction of logic programs from examples and background knowledge. In ILP, the shift of attention from program synthesis to knowledge discovery resulted in advanced techniques that are practically applicable for discovering knowledge in relational databases. This paper gives a brief introduction to ILP, presents selected ILP techniques for relational knowledge discovery and reviews selected ILP applications.

Keywords: Inductive Logic Programming, Machine Learning, Induction, Knowledge Discovery, Relational Database, Molecular Biology.

§1 Introduction

Inductive logic programming (ILP) [30, 34, 26] is a research area that has its backgrounds in inductive machine learning and logic programming. ILP research aims at a formal framework as well as practical algorithms for inductive learning of relational descriptions that typically have the form of logic programs. From logic programming, ILP has inherited its sound theoretical basis, and from machine learning, an experimental approach and orientation towards practical applications. ILP research has been strongly influenced also by computational learning theory (COLT), and recently, by knowledge discovery in databases (KDD) [14] which led to the development of new techniques for relational data mining.

In general, an ILP learner is given an initial theory B (background knowledge) and some evidence E (examples), and its aim is to induce a theory H (hypothesis) that together with B *explains* some properties of E. In most cases the hypothesis H has to satisfy certain restrictions, which we shall refer to as the *bias*. Bias includes prior expectations and assumptions, and can therefore be considered as the logically unjustified part of the background knowledge. Bias is needed to reduce the number of candidate hypotheses. It consists of the language bias L, determining the hypothesis space, and the search bias which restricts the search of the space of possible hypotheses.

The background knowledge used to construct hypotheses is a distinctive feature of ILP. It is well known that relevant background knowledge may substantially improve the results of learning in terms of accuracy, efficiency and the explanatory potential of the induced knowledge. On the other hand, irrelevant background knowledge will have just the opposite effect. Consequently, much of the art of inductive logic programming lies in the appropriate selection and formulation of background knowledge to be used by the selected ILP learner.

Let us illustrate some of the tasks that ILP can address. Some of these concern *predictive knowledge discovery* where the goal is to induce classification and prediction rules from the given data and background knowledge. Others concern *descriptive knowledge discovery* where the goal is to induce a theory that describes properties of the data, but may have a form that can not be used for prediction and classification.

A logic programming problem. Let the training set E consist of positive and negative examples for the predicate `sort/2`. A positive example $e \in E^+$ provides information known to be true and should be entailed by the induced hypothesis. A negative example $e \in E^-$ provides information that is known not to be true and should not be entailed.

$$E^+ = \{\texttt{sort([2,1,3],[1,2,3])}\}$$
$$E^- = \{\texttt{sort([2,1],[1])},\ \texttt{sort([3,1,2],[2,1,3])}\}$$

Let the background knowledge B contain correct definitions of predicates `permutation/2`, which is true if the second argument is a permuted list of the first argument, and `sorted/1`, which is true if its list-argument is sorted in ascending order. If the hypothesis language L contains all definite clauses using the predicate and functor symbols appearing in the examples and background knowledge, an ILP system can induce the following hypothesis:

```
sort(X,Y) ← permutation(X,Y), sorted(Y).
```

The above logic program can be used to answer queries about the `sort` predicate, e.g. `?- sort([4,3],[3,4])`. Hence, the illustrated program synthesis task belongs to predictive knowledge discovery.

In descriptive knowledge discovery, using E^+ and B only, an induced theory could contain the following clauses:

```
sorted(Y) ← sort(X,Y).
permutation(X,Y) ← sort(X,Y).
sorted(X) ← sort(X,X).
```

Whereas predictive knowledge discovery results in a program for sorting lists, descriptive knowledge discovery results in uncovered regularities and relations among the involved predicates.

While it is impractical at present to induce real-world logic programs from examples and background knowledge predicates only, ILP techniques have shown their potential as programming aids that can produce fully specified procedures when given examples, background predicates, and a partial specification of the target program as bias.[1)]

A knowledge discovery problem. Consider a problem of learning family relations where the predictive knowledge discovery task is to define the target relation daughter(X,Y), which states that person X is a daughter of person Y, in terms of relations defined in background knowledge B.

E^+ = {daughter(mary,ann), daughter(eve,tom)}
E^- = {daughter(tom,ann), daughter(eve,ann)}
B = {mother(ann,mary), mother(ann,tom), father(tom,eve), father(tom,ian), female(ann), female(mary), female(eve), parent(X,Y) ← mother(X,Y), parent(X,Y) ← father(X,Y), male(pat), male(tom)}

In the hypothesis language of definite clauses, given E^+, E^- and B, a predictive ILP system can induce the following clause:

```
daughter(X,Y) ← female(X), parent(Y,X).
```

Alternatively, a learner could have induced a set of clauses:

```
daughter(X,Y) ← female(X), mother(Y,X).
daughter(X,Y) ← female(X), father(Y,X).
```

In descriptive knowledge discovery, given E^+ and B only, an induced theory could contain the following clauses:

```
← daughter(X,Y), mother(X,Y).
female(X) ← daughter(X,Y).
mother(X,Y); father(X,Y) ← parent(X,Y).
```

One can see that in the predictive knowledge discovery setting classification rules are generated, whereas in the descriptive setting database regularities are derived.

The reminder of this paper gives a brief introduction to ILP and presents selected ILP techniques for relational knowledge discovery. The overview is restricted to techniques satisfying the strong criterion formulated for machine learning by Michie [27] that requires explicit symbolic form of induced descriptions. Finally, an overview of ILP applications is given, with an emphasis on applications where ILP shows its advantages over propositional learning.

§2 Problem specification

An inductive logic programming task can be formally defined as follows:

Given:

- a set of examples E
- a background theory B
- a language bias L that defines the clauses allowed in hypotheses
- a notion of *explanation*

Find: a hypothesis $H \subset L$ which explains the examples E with respect to the theory B.

This definition needs to be instantiated for different types of ILP tasks.[34] The instantiation will concern the representation of training examples, the choice of a hypothesis language and an appropriate notion of explanation. By explanation we here refer to an acceptance criterion for hypotheses: a hypothesis explains the data if it satisfies a certain user-defined criterion w.r.t. the data. We will discuss some formal acceptance criteria used in different ILP settings, but we also need to bear in mind that ILP aims at the induction of hypotheses that are expressed in an explicit symbolic form, that can be easily interpreted by the user/expert and may contribute to the better understanding of the problem addressed, ideally forming a piece of new knowledge discovered from the data.

Predictive ILP is the most common ILP setting, often referred to as *normal ILP*, *explanatory induction*, *discriminatory induction*, or *strong ILP*. Predictive ILP is aimed at learning of classification and prediction rules. This ILP setting typically restricts E to ground facts, and H and B to sets of definite clauses. The strict notion of explanation in this setting usually denotes coverage, defined by logical entailment, and requires global completeness and consistency.

Global completeness and consistency implicitly assume the notion of *intensional coverage* defined as follows. Given background theory B, hypothesis H and example set E, an example $e \in E$ is (intensionally) covered by H if $B \cup H \models e$. Hypothesis H is (globally) complete if $\forall e \in E^+ : B \cup H \models e$. Hypothesis H is (globally) consistent if $\forall e \in E^- : B \cup H \not\models e$.

Given the restriction to definite theories $T = B \cup H$, for which there exists a unique least Herbrand model $M(T)$,[*1] and to ground atoms as examples, this is equivalent to requiring that all examples in E^+ are true in $M(B \cup H)$.[34]

By relaxing the notion of explanation to allow incomplete and inconsistent theories that satisfy some other acceptance criteria (high predictive accuracy, significance, compression), the predictive ILP setting can be extended to include learning of classification and prediction rules from imperfect data. In a broader sense, predictive ILP incorporates also *learning of logical decision trees,*[2] *first-order regression*[20] and *constraint inductive logic programming*[39] for which different acceptance criteria apply.

Descriptive ILP is sometimes referred to as *confirmatory induction, non-monotonic ILP, description learning,* or *weak ILP.* Descriptive ILP is usually aimed at learning of clausal theories.[7] This ILP setting typically restricts B to a set of definite clauses, H to a set of (general) clauses, and E to positive examples. The strict notion of explanation used in this setting requires that all clauses c in H are true in some preferred model of $T = B \cup E$, where the preferred model of T may be, for instance, the least Herbrand model $M(T)$. (One may also require the completeness and minimality of H, where completeness means that a maximally general hypothesis H is found, and minimality means that the hypothesis does not contain redundant clauses.)

By relaxing the strict notion of explanation used in clausal discovery[7] to allow for theories that satisfy some other acceptance criteria (similarity, associativity, interestingness), descriptive ILP can be extended to incorporate *learning of association rules,*[5] *first-order clustering,*[6, 12] *database restructuring*[15, 40] *subgroup discovery,*[44] *learning qualitative models*[19] and *equation discovery.*[11]

Other ILP settings have also been investigated, the most important being *relational instance-based learning.*[13] Excellent predictive results have been achieved by the relational instance-based learner RIBL[13] in numerous classification and prediction tasks. Recently, *first-order reinforcement learning* has also been studied by De Raedt and Džeroski. Since these two ILP settings do not involve hypothesis formation in explicit symbolic form, the developed techniques do not qualify as techniques for relational knowledge discovery.

§3 Dimensions of the problem

The computational difficulty of an ILP learning problem depends on how its primary components are instantiated, i.e., which languages are allowed for the specification of examples, background knowledge, and hypothesis, and which notion of explanation is used. In the most general case, full first-order logic could

[*1] The least Herbrand model $M(T)$ of a definite theory T is a set of ground facts $M(T) = \{a \mid a \in$ Herbrand base of T and $T \models a\}$, where the Herbrand base of T is the set of all ground atoms which can be formed using the predicate, functor and constant symbols occuring in T. From a practical point of view, logical entailment of fact a by theory T can be verified using a Prolog interpreter with knowledge base T and query ?-a. Thus, roughly speaking, the least Herbrand model of T is the set of all facts that are logically entailed by theory T.

be used as a language, and as stated above for predictive ILP, explanation would be identified with logical entailment. Due to the undecidability of first-order logic, in this case it is not even decidable whether a hypothesis is a solution to the learning problem. One of the central goals of ILP is thus to find restrictions of the various components of the learning problem that on the one hand make the problem easier while on the other still allow interesting concepts to be learned.

3.1 Choice of the learning setting and acceptance criteria

The nature of the problem indicates what is the appropriate ILP setting and the corresponding acceptance criteria for hypotheses.

When learning from real-life data which is imperfect, the strict notions of explanation need to be relaxed in order to deal with noise (random errors) in the examples E and background knowledge B, sparseness of the training set E, and inappropriateness of B that may contain predicates that are not relevant or are insufficient for reliably solving a learning task.

Due to these reasons, and also due to the difference in the nature of various learning tasks, the strict notion of explanation needs to be replaced by a less strict, and typically different acceptance criteria for hypotheses. Section 4 gives an indication of the variety of tasks and criteria used in relational knowledge discovery.

3.2 Choice of the hypothesis language

All ILP systems use a language bias L to restrict the set of clauses allowed in the hypotheses, thus limiting the search space of hypotheses. Of course, an inappriopriate bias can prevent the learner from finding a hypothesis explaining the given examples.

Most ILP systems in use today employ clausal first-order logic as their representation, usually concentrating on Horn clauses, i.e., clauses with one positive literal (head). Entailment in these languages is still undecidable, so many systems in addition assume *function-free* languages which are decidable and have turned out to be adequate for many practical problems. Furthermore, it is possible to automatically transform a clausal representation with function symbols into a function-free representation using a technique known as *flattening*.[38] The answer set of the transformed function-free program is, however, equivalent to the original program with functions only under certain conditions.

3.3 Structuring the hypothesis space

ILP can be regarded as a search problem, where the space of possible solutions is determined by the syntactic (language) bias L. Searching the whole hypothesis space is clearly inefficient, therefore structuring the search space is necessary. Nearly all ILP learners structure the search by means of the dual notions of generalization and specialization.

There is also a decision criterion, e.g., global completeness and consistency in predictive ILP, to check whether a candidate hypothesis H is a solution to a problem. However, even for a Horn clause program consisting of one ground fact, one ground query and two Horn clauses with two literals, the ILP problem defined above remains undecidable. Therefore, almost all ILP approaches have chosen to replace (semantic) logical entailment ($\models$) by (syntactic) ϑ-subsumption ($\geq_\vartheta$), first used for learning by Plotkin.

Semantic generalization. A hypothesis H_1 is semantically more general than a hypothesis H_2 with respect to theory B if and only if $B \cup H_1 \models H_2$.

Syntactic generalization or ϑ-subsumption. A clause c_1 (a set of literals) is syntactically more general than a clause c_2 if and only if there exists a substitution ϑ such that $c_1\vartheta \subseteq c_2$. In this case we say that clause c_1 ϑ-subsumes c_2 ($c_1 \geq_\vartheta c_2$). A hypothesis H_1 is syntactically more general than a hypothesis H_2 if and only if for each clause c_2 in H_2 there exists a clause c_1 in H_1 such that c_1 is syntactically more general than c_2.

Plotkin has proved that the syntactic *is more general than* relation induces a lattice on the set of all clauses, up to equivalence classes and variable renamings, i.e., a lattice on the set of *reduced* clauses. Equivalence classes consist of clauses that are ϑ-subsumption equivalent ($\equiv_\vartheta$): $c \equiv_\vartheta d$ if and only if $c \geq_\vartheta d$ and $d \geq_\vartheta c$. A clause is reduced if it is not ϑ-subsumption equivalent to any proper subset of itself. In a lattice, any two clauses have a least upper bound (*lub*) and a greatest lower bound (*glb*) which are unique up to renaming of variables.

Notice that when a clause c_1 (resp. hypothesis) is syntactically more general than a clause c_2 it is also semantically more general: i.e., if $c_1 \geq_\vartheta c_2$, then $c_1 \models c_2$.

The converse is not true, as shown by the examples in Fig. 1. As illustrated by the last example in Fig. 1, the incompleteness of ϑ-subsumption is due to clauses that can resolve with themselves. If these self-resolving clauses are not allowed, ϑ-subsumption is complete with respect to $\models$. Checking ϑ-subsumption is an NP-complete problem, but in many cases it is possible to exploit structural properties of the data (determinacy and locality) to greatly reduce the combinatorial matching problem and runtime when compared to a blind matching approach.

Both *is more general than* relations are useful for induction because:

- when generalizing a hypothesis H_1 to H_2, all formulae f entailed by the hypothesis H_1 and background theory B will also be entailed by hypothesis H_2 and theory B, i.e. $(B \cup H_1 \models f) \rightarrow (B \cup H_2 \models f)$.
- when specializing a hypothesis H_1 to H_2, all formulae f logically not entailed by hypothesis H_1 and background theory B will not be entailed by hypothesis H_2 and theory B either, i.e. $(B \cup H_1 \not\models f) \rightarrow (B \cup H_2 \not\models f)$.

The two properties can be used to prune large parts of the search space.

3.4 Searching the hypothesis space

In the ϑ-subsumption lattice of reduced clauses, an ILP learner can search the hypothesis space systematically, either top-down or bottom-up.

c_1:=daughter(X,Y) $\leftarrow$ parent(Y,X)
c_2:=daughter(mary,ann) $\leftarrow$ parent(ann,mary)
c_3:=daughter(mary,ann) $\leftarrow$ female(mary), parent(ann,mary), parent(tom,mary)
$c_1 \geq_\vartheta c_2$ and $c_1 \geq_\vartheta c_3$, both with $\vartheta = \{X/mary, Y/ann\}$.

c_4:=daughter(X,Y) $\leftarrow$ parent(Y,X), parent(U,V)
$c_1 \equiv_\vartheta c_4$; c_1 is reduced, c_4 is not reduced.

c_5:=q(Y,Z,X) $\leftarrow$ q(X,Y,Z), c_6:=q(Z,X,Y) $\leftarrow$ q(X,Y,Z)
$c_5 \not\geq_\vartheta c_6$ and $c_6 \not\geq_\vartheta c_5$, but $c_5 \models c_6$ and $c_6 \models c_5$.

Fig. 1 ϑ-subsumption examples

Specialization techniques search the hypothesis space top-down, from the most general to specific hypotheses, using a specialization/refinement operator, which ideally computes the set of all minimal (most general) specializations of the currently considered clause.

Generalization techniques search the hypothesis space in a bottom-up manner: they start from the training examples (most specific hypotheses) and search the hypothesis space by employing a generalization operator. Ideally, generalizations are the *least general generalizations* of the given examples ("cautious generalization").

In predictive ILP, both techniques repeat their procedure on a reduced example set if the found clause by itself does not entail all positive examples. They use thus an iterative process to compute disjunctive hypotheses consisting of more than one clause (often called the "covering" approach).

Specialization techniques are better suited for empirical learning in the presence of noise since top-down search can easily be guided by heuristics. On the other hand, generalization techniques are best suited for interactive and incremental learning from few examples.

3.5 Advanced search techniques

One of the key problems in ILP is how to search the space of logic programs efficiently. Existing ILP algorithms use variants of greedy search to explore the space of logic programs. Due to myopic behavior, a greedy search algorithm is easily trapped in a local minimum. To alleviate the problem of myopic greedy search, several advanced search techniques have been developed, such as beam search, bi-directional search, stochastic search and genetic search. The simplest is beam search in which a beam of literals is searched in order to find the best literals or conjunctions of literals needed in clause construction.

Bi-directional search is used in the Progol algorithm,[31] combining the advantages of top-down and bottom-up search. The algorithm (1) randomly selects an example e_i, (2) uses (depth-bounded) inverse resolution to construct the most specific clause $c(e_i, B)$ that explains e_i (such that $c(e_i, B) \models e_i$), (3) by top-down search of the ϑ-subsumption lattice finds a clause c_i which subsumes $c(e_i, B)$, is consistent w.r.t. the current example set E and maximally compresses a set of examples subsumed by c_i, (4) adds c_i to the current hypothesis H. It repeats the process with the examples that are still not covered until no more compression is possible. Bi-directional search results in a substantial reduction of the search space.

Stochastic search in the state space of logic programs can be used to alleviate the problem of myopic greedy search. In a stochastic ILP system MILP,[25] based on ideas from simulated annealing, transitions in the state space are probabilistic; the probability of a transition to a state is proportional to the (heuristically evaluated) quality of the state.

Genetic search is performed on a population of hypotheses. In each iteration, hypotheses are mutated and recombined, and the best n hypotheses are kept. To use such a strategy for ILP, a representation of hypotheses must be found that allows effective mutation and recombination operators to be formulated. In the GA-Smart system,[17] a restricted hypothesis space based on clause templates was selected to allow for an appropriate bitstring representation. In the system Gilp by Varšek, a genetic algorithm operating on a population of Prolog clauses was employed in conjunction with the covering principle.

3.6 Other dimensions

Empirical versus incremental learning. This dimension describes the way the examples E are obtained. In empirical ILP, the evidence is given at the start and not changed afterwards. In incremental ILP, the user supplies the examples one by one, in a piecewise fashion.

Interactive versus non-interactive learning. In interactive ILP, the learner is allowed to pose questions to the user about the intended interpretation. Usually these questions query for the intended interpretation of examples or clauses.

Theory revision. In addition to examples E and background knowledge B, an existing theory may be given to the learner as a starting point for hypothesis construction, often requiring that it be minimally modified to arrive at a hypothesis. Incremental and interactive learning approaches require theory revision.

Predicate invention. Predicate invention denotes the process whereby entirely new predicates (not present in E and B) are induced. Predicate invention results in extending the vocabulary of the learner and may therefore facilitate the learning task. Generally speaking, search space becomes very large in predicate invention. In spite of this difficulty, some systems invent a predicate without any user interaction.[21]

Single versus multiple predicate learning. In single predicate learning, the evidence contains examples for only one predicate and the aim is to induce a definition for this predicate; in multiple predicate learning, the aim is to learn a set of possibly interacting predicate definitions or properties that hold among various predicates. Descriptive ILP tasks involving multiple predicates are handled easier and more naturally, due to overcoming the order dependency problem of multiple predicate learning in predictive ILP.

§4 Techniques for relational knowledge discovery

4.1 Predictive ILP

Learning of classification rules. This is the standard ILP setting that has been used in numerous successful predictive knowledge discovery applications. The well-known systems for classification rule induction include Foil,[37)][*2] Golem[33)] and Progol.[31)] Foil is efficient and best understood due to its similarity to Clark and Niblett's CN2. On the other hand, Golem and Progol are champions concerning successful ILP applications, despite the fact that they are substantially less efficient. Foil is a top-down learner, Golem is a bottom-up learner, and Progol uses a combined search strategy. All are mainly concerned with single predicate learning from positive and negative examples and background knowledge; in addition, Progol can also be used to learn from positive examples only. They use different acceptance criteria: compression, coverage/accuracy and minimal description length, respectively.

Induction of logical decision trees. The system Tilde[2)] belongs to Top-down induction of decision trees algorithms. It can be viewed as a first-order upgrade of Quinlan's C4.5, employing logical queries in tree nodes which involves appropriate handling of variables. The main advantage of Tilde is its efficiency and capability of dealing with large numbers of training examples, which are the well-known properties of Tilde's propositional ancestors. Hence Tilde currently represents one of the most appropriate systems for predictive knowledge discovery. Besides the language bias, Tilde allows for lookahead and prepruning (according to the minimal number of examples covered) defined by parameter setting.

First-order regression. The relational regression task can be defined as follows: Given training examples as positive ground facts for the target predicate $r(Y, X_1, ..., X_n)$, where the variable Y has real values, and background knowledge predicate definitions, find a definition for $r(Y, X_1, ..., X_n)$, such that each clause has a literal binding Y (assuming that $X_1, ..., X_n$ are bound). Typical background knowledge predicates include less-or-equal tests, addition, subtraction and multiplication. An approach to relational regression is implemented in the system FORS (First Order Regression System)[20)] which performs top-down search of a refinement graph. In each clause, FORS can predict a value for the target variable Y as the output value of a background knowledge literal, as a constant, or as a linear combination of variables appearing in the clause (using linear regression).

Inductive Constraint Logic Programming. It is well known that Constraint Logic Programming (CLP) can successfully deal with numerical constraints. The idea of Inductive Constraint Logic Programming (ICLP)[39)] is to

[*2] The successor of Foil, the system Ffoil, can successfully be used for inducing relational definitions of functions.

benefit from the number-handling capabilities of CLP, and to use the constraint solver of CLP to do part of the search involved in inductive learning. To this end a maximally discriminant generalization problem in ILP is transformed to an equivalent constraint satisfaction problem (CSP). The solutions of the original ILP problem can be constructed from the solutions of CSP, which can be obtained by running a constraint solver on CSP.

4.2 Descriptive ILP

Learning of clausal theories and association rules. In discovering full clausal theories, as done in the system Claudien, [7] each example is a Herbrand model, and the system searches for the most general clauses that are true in all the models. Clauses are discovered independently from each other, which is a substantial advantage for data mining, as compared to the learning of classification rules (particularly learning of mutually dependent predicates in multiple predicate learning). In Claudien, search of clauses is limited by the language bias. Its acceptance criterion can be modified by setting two parameters: the requested minimal accuracy and minimal number of examples covered. In another clausal discovery system, Primus, [16] the best-first search for clauses is guided by heuristics measuring the "confirmation" of clauses. The Claudien system was further extended to Warmr [5] that enables learning of association rules from multiple relations.

First-order clustering. Top-down induction of decision trees can be viewed as a clustering method since nodes in the tree correspond to sets of examples with similar properties, thus forming concept hierarchies. This view was adopted in C0.5, [6] an upgrade of the Tilde logical decision tree learner. An approach combining learning and conceptual clustering techniques was implemented in the system Cola. [12] Given a small (sparse) set of classified training instances and a set of unclassified instances, Cola uses Bisson's conceptual clustering algorithm KBG on the entire set of instances, climbs the hierarchy tree and uses the classified instances to identify (single or disjunctive) class descriptions.

Database restructuring. The system Fender [40] searches for common parts of rules describing a concept, thus forming subconcept definitions to be used in the refurmulation of original rules. The result is a knowledge base with new intermediate concepts and deeper inferential structure than the initial "flat" rulebase. The system Index[15] is concerned with the problem of determining which attribute dependencies (functional or multivalued) hold in the given relational database. The induced attribute dependencies can be used to obtain a more structured database. Both approaches can be viewed as doing predicate invention, where (user selected) invented predicates are used for theory restructuring.

Subgroup discovery. The subgroup discovery task is defined as follows: given a population of individuals and a property of those individuals we are interested in, find the subgroups of the population that are statistically "most interesting," i.e., are as large as possible and have the most unusual statistical (distributional) characteristics with respect to the property of interest. The system Midos[44] guides the top-down search of potentially interesting subgroups using numerous user-defined parameters.

Learning qualitative models of dynamic systems. The automated construction of models of dynamic system may be aimed at qualitative model discovery. A recent qualitative model discovery system, [19] using a Qsim-like representation, is based on Coiera's Genmodel to which signal processing capabilities have been added.

Equation discovery. The system LAGRANGE [11] discovers a set of differential equations from an example behavior of a dynamic system. Example behaviors are specified by lists of measurements of a set of system variables, and background knowledge predicates enable the introduction of new variables as time derivatives, sines or cosines of system variables. New variables can be further introduced by multiplication.

§5 Selected ILP applications

The number and diversity of ILP applications has been increasing at a steady pace. Application areas where ILP has been used for relational knowledge discovery range from ecology through mechanical engineering to traffic control. Even human feelings are modeled in a relational representation framework. [36] Applications in the area of molecular biology have come closest to practical relevance. Among the first was predicting protein secondary structure,[32] followed by predicting drug activity through modeling structure-activity relations[22, 23] and predicting the mutagenicity of aromatic and heteroaromatic nitro-compounds.[41] On these problems, which are of immediate practical interest, results that are better or equal to the best previously known results have been obtained, in addition to understandable and relevant new knowledge. Recent ILP applications in the area of molecular biology include prediction of rodent carcinogenicity bioassays,[43] modeling structure-activity relations for modulating transmembrane calcium movement,[42] pharmacophore discovery for ACE inhibition[35] and diterpene structure elucidation. [10] In the remainder of this section we briefly summarize the abovementioned ILP applications in molecular biology domains.

5.1 Predicting protein secondary structure

A protein is basically a string of amino acids (or *residues*). Predicting the three-dimensional shape of proteins from their amino acid sequence is widely believed to be one of the hardest unsolved problems in molecular biology. It is also of interest to the pharmaceutical industry since the shape of a protein determines its function.

The sequence of amino acids is called the *primary structure* of the protein. Spatially, the amino acids are arranged in different patterns (spirals, turns, flat sections, etc.). The three-dimensional spatial shape of a protein is called the *secondary structure*. A limited version of the problem of predicting the shape (*secondary structure*), involving only the α-helix shape, was considered by Muggleton et al. [32] The ILP system GOLEM used information on the sequence of residues and properties of the individual residues to predict whether a given residue will belong to an α-helix. The induced rules achieved an accuracies of 81% on unseen cases, which at the time was better than the best previously reported result of 76%.[24]

5.2 Modeling structure-activity relations

A central concern of chemistry is understanding the relationships between chemical structure and activity. In most cases, these relationships cannot be derived solely from physical theory and experimental evidence is essential. Such empirically derived relationships are called Structure Activity Relationships (SARs). In a typical SAR problem, a set of chemicals of known structure and activity are given, and the task is to find a predictive theory relating the structure of a compound to its activity. This relationship can then be used to select structures with high or low activity. Typically, knowledge of such relationships is used for devising clinically effective, non-toxic drugs.

The ILP system GOLEM[33] has been applied to several problems of this kind, including the problem of inhibition of E. Coli Dihydrofolate Reductase by two different groups of drugs (pyrimidines and triazines)[22, 23] and the properties (toxicity, acetocholinesteraze inhibition, etc.) of Tacrine (a drug for treating Alzheimers disease) derivatives.[23] The ILP formulation of the problems compares the properties of pairs of compounds with known activity. The background knowledge contains predicates specifying the chemical structure of the drugs with known activity, as well as properties of some groups of atoms (substituents or radicals). For the problem of inhibition of E. Coli Dihydrofolate Reductase, GOLEM induced nine rules comparing drug activities. The Spearman rank correlation of the drug activity order predicted by GOLEM with the actual order was 0.46 for a testing set of drugs[22] as compared to a correlation of 0.42 by a classical SAR approach, taken by Hansch et al.[18] Besides achieving better accuracy than traditional methods, the induced rules also provide a description of the chemical laws governing the problem.

5.3 Predicting mutagenicity

Srinivasan et al.[41] applied the ILP system Progol[31] to induce theories for predicting the mutagenicity of a set of 230 aromatic and heteroaromatic nitrocompounds. The prediction of mutagenicity is important as it is relevant to the understanding and prediction of carcinogenicity. The compounds used in this study are more heterogeneous structurally than those used in the drug design domains and can only be fully represented in a first-order setting.

Of the 230 compounds, 138 have positive levels of log mutagenicity, these are labeled "active" and constitute the positive examples: the remaining 92 compounds are labeled "inactive" and constitute the negative examples. The target relation is in this case $active(C)$, stating that compound C has positive log mutagenicity. The background knowledge contains the structure of the compounds represented as a list of atoms and bonds that can be found in each compound. The predicate $atm(C, A, E, T, Charge)$ states that atom A in compound C is an atom of element E (e.g., carbon), of type T (e.g., aromatic carbon) with charge *Charge*. The predicate $bond(C, A1, A2, BT)$ states that there is a bond of type BT (e.g., aromatic bond) between atom $A1$ and atom $A2$ of compound C. The facts for these predicates were generated by the molecular modeling program QUANTA, where the compounds were entered through a chemical editing facility.

In addition, four attributes are provided for analysis of the compounds. These can be used directly by both propositional and ILP learners. They are: (1) the hydrophobicity of the compound (termed logP); (2) the energy level of the lowest unoccupied molecular orbital (termed LUMO); (3) a boolean attribute identifying compounds with 3 or more benzyl rings (termed indicator variable I1); and (4) a boolean attribute identifying a sub-class of compounds termed acenthryles (termed indicator variable Ia). The last two are pre-selected structural features that incorporate chemical expert knowledge.

Generic structural knowledge was used as background knowledge in some experiments with Progol. It includes definitions of the concepts of methyl groups, nitro groups, aromatic rings, heteroaromatic rings, connected rings, ring length, and the three distinct topological ways to connect three benzene rings. These definitions are generic to the field of organic chemistry.

The 230 compounds are divided into two sets: 188 compounds that could be fitted using linear regression (regression-friendly set), and 42 compounds that could not (regression-unfriendly set). In the Progol experiments, accuracies of theories constructed for the 188 compounds were estimated from a 10-fold cross-validation and the accuracy of theories for the 42 compounds were estimated by a leave-one-out procedure.

In summary, Progol produced theories that perform as well as linear regression or neural networks (88% vs. 89% accuracy) on the regression-friendly set and much better (88% vs. 69% accuracy) on the regression-unfriendly set. In comparison with CART,[3] no significant differences in performance exist when the pre-selected structural features are available. However, when these features are not available, Progol performs much better on the regression-friendly set (88 % vs 83 %). Progol also performed better than FOIL.[37]

It is worth noting that Progol generated a single rule for the regression-unfriendly set. The structure indicated by this rule is a new structural alert for high mutagenicity in chemical compounds.

5.4 Prediction of rodent carcinogenicity bioassays

The problem addressed by King and Srinivasan[43] was to predict the carcinogenicity of a diverse set of chemical compounds, learning from a dataset that originates from the US National Toxicology Program. The data on carcinogencity was obtained by testing the chemicals on rodents, each trial taking several years and hundreds of animals (long term rodent bioassays). The training set consisted of 291 compounds, of which 161 carcinogens, while the testing set consisted of 39 compounds, of which 22 carcinogens.

An accuracy of 64% on unseen cases was achieved by Progol. No SAR models (of either human or machine origin) were significantly more accurate than this. Progol was also the most accurate method that did not use data from biological tests on rodents (these data were not available). A set of structural alerts for carcinogenicity was generated automatically and the chemical rationale for them investigated. Unlike other SAR methods, the Progol alerts are statistically independent of those available from an existing carcinogenesis test based on Salmonella mutagenicity.

5.5 SAR models for for modulating transmembrane calcium movement

The compounds considered in this study are a class of calcium-channel activators. Their activity is measured as the logarithm of the potency of the compound relative to an accepted standard calcium-channel activator. Initial experiments in the chemical literature used the hydrophobicity and molar reflectivity of the compounds to derive linear models of the SARs in this domain.

Progol was applied to this dataset yielding structural concepts that were translated into boolean-valued attributes.[42] These were added to the abovementioned attributes and significanly improved the linear model predicting activity. The final model is comparable in accuracy to a much more complex model derived using computational chemistry methods.

5.6 Pharmacophore discovery for ACE inhibition

The task in this domain[35] was to identify the structure (pharmacophore) responsible for the activity of ACE (Angiotensin-converting enzyme) inhibitors. Given were the structures of 28 molecules that display the activity of ACE inhibition, described by atom and bond information, including the 3D positions of the atoms. Background knowledge about atom groups and the distances between pairs of groups was available.

Pharmacophores are typically described in terms of types of atoms (e.g., hydrogen donors) or functional groups and the pairwise distances among them. The pairwise distances define the geometric arrangement of the atoms or groups that is necessary for them to lock into a particular site of a protein such as ACE. P-Progol discovered a four-piece pharmacophore with one zinc-binder and three hydrogen acceptors present in all molecules. According to expert opinion, the discovered pharmacophore is equivalent to the generally accepted pharmacophore for ACE inhibition.

5.7 Diterpene structure elucidation

Diterpenes are organic compounds of low molecular weight with a skeleton of 20 carbon atoms. They are of significant chemical and commercial interest because of their use as lead compounds in the search for new pharmaceutical effectors. The interpretation of diterpene ^{13}C NMR-spectra normally requires specialists with detailed spectroscopic knowledge and substantial experience in natural products chemistry, more specifically knowledge on peak patterns and chemical structures.

Given a database of peak patterns for diterpenes with known structure, several ILP approaches were applied to discover correlations between peak patterns and chemical structure.[10] The approaches used include relational instance based learning, induction of logical decision trees and inductive constraint logic. Performance close to the one of domain experts was achieved, which suffices for practical use.

More specifically, the task addressed was to identify the skeleton (type) of diterpenoid compounds, given their ^{13}C-NMR-Spectra that include the multiplicities and the frequencies of the skeleton atoms. The multiplicity of a carbon atom is the number of hydrogen atoms connected to it. Pre-processed (so-called reduced) multiplicities were used which result from eliminating measurement side-effects. Each molecule was thus represented by a set of facts of the form `red(MoleculeID,Multiplicity,Frequency)`, yielding a nondeterminate representation. Twenty-three different skeleton types are represented in the whole set of 1503 compounds: there are thus 23 possible class values (target predicates).

Several propositional versions of the problem were considered. The simplest represented a molecule by four numbers: the numbers of atoms of each of the four possible multiplicities. The most complex represented each molecule by 860 attributes. All of these representations yield an accuracy of approx. 80% on unseen cases as estimated by cross-validation.

Using the relational representation in combination with the four features mentioned above, the ILP systems RIBL[13] and TILDE[2] yield accuracies of over 90%. This is in the range of the accuracy with which experts classify diterpenes into skeleton types given ^{13}C NMR spectra only. That number can actually only be estimated since it is expensive to have an expert carry out a statistically significant number of structure predictions without using other additional information that often becomes available from heterogeneous sources (such as literature, and ^{1}H NMR spectra).

While RIBL is instance-based and does not yield understandable theories, TILDE is able to do so, at least in principle. In practice, understandability of the theories is diminished when only complex theories are accurate. Results with TILDE show, however, that it may be advantageous to find a slightly less accurate theory that is more understandable.

§6 Conclusions

This paper gives a short overview of the field of ILP, its goals and main methods. It presents two main ILP settings, predictive and descriptive ILP, and outlines recently developed techniques for relational knowledge discovery. ILP applications are illustrated by a representative set of ILP applications in molecular biology.

Acknowledgment

This work reflects some of the research results achieved in the EU funded projects no. 6020 ILP (1992–95) and no. 20237 ILP2 (1996–98), ILP related projects funded by the Slovenian Ministry of Science and Technology and the Information-Technology Promotion Agency of Japan through the Advaned Software Enrichment Project. The authors are grateful to Luc De Raedt, Peter Flach and Stefan Wrobel for their collaboration and previous joint overviews of ILP that strongly influenced this work and its presentation.

References

1) Bergadano, F. and Gunetti, D., *Inductive Logic Programming: From Machine Learning to Software Engineering*, The MIT Press, 1995.

2) Blockeel, H. and De Raedt, L., "Experiments with top-down induction of logical decision trees," to appear in *Artificial Intelligence*, 1998.

3) Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J., *Classification and Regression Trees*, Wadsworth, Belmont, 1984.

4) Cohen, W., "Recovering software specifications with inductive logic programming," in *Proc. Twelfth National Confeernce on Artificial Intelligence*, The MIT Press, 1994.

5) Dehaspe, L. and De Raedt, L., "Mining association rules in multiple relations," in *Proc. Seventh Int. Workshop on Inductive Logic Programming*, LNAI 1297, Springer, pp. 125–132, 1997.

6) De Raedt, L. and Blockeel, H., "Using logical decision trees for clustering," in *Proc. Seventh Int. Workshop on Inductive Logic Programming*, LNAI 1297, Springer, pp. 133–140, 1997.

7) De Raedt, L. and Dehaspe, L., "Clausal discovery," *Machine Learning*, 26(2/3), pp. 99–146, 1997.

8) Dolšak, B. and Muggleton, S., "The application of inductive logic programming to finite-element mesh design," in *Inductive Logic Programming* (S. Muggleton, ed.), Academic Press, pp. 453–472, 1992.

9) Džeroski, S., De Haspe, L., Ruck, B. and Walley, W., "Classification of river water quality data using machine learning," in *Proc. Fifth Int. Conference on the Development and Application of Computer Technologies to Environmental Studies, Vol. I: Pollution Modelling*, Computational Mechanics Publications, Southampton, pp. 129–137, 1994.

10) Džeroski, S., Schulze-Kremer, S., Heidtke, K., Siems, K., Wettschereck, D. and Blockeel, H., "Diterpene Structure Elucidation from ^{13}C NMR Spectra with Inductive Logic Programming," to appear in *Applied Artificial Intelligence*, 1998.

11) Džeroski, S. and Todorovski, L., "Discovering dynamics: From inductive logic programming to machine discovery," in *Proc. Tenth Int. Conference on Machine Learning*, Morgan Kaufmann, pp. 97–103, 1993.

12) Emde, W., "Learning of characteristic concept descriptions from small sets to classified examples," in *Proc. Seventh European Conference on Machine Learning*, LNAI 784, Springer, pp. 103–121, 1994.

13) Emde, W. and Wettschereck, D., "Relational instance-based learning," in *Proc. Thirteenth Int. Conference on Machine Learning*, Morgan Kaufmann, pp. 122–130, 1996.

14) Fayyad, U., Piatetsky-Shapiro, G., Smyth, P. and Uthurusamy, R., eds., *Advances in Knowledge Discovery and Data Mining*, The MIT Press, 1995.

15) Flach, P.A., "Predicate invention in inductive data engineering," in *Proc. Sixth European Conference on Machine Learning*, LNAI 667, Springer, pp. 83-94, 1993.

16) Flach, P. and Lachiche, N., "Cooking up integrity constraints with Primus," *Technical report*, University of Bristol, 1998.

17) Giordana, A. and Sale, C., "Learning structured concepts using genetic algorithms," in *Proc. Ninth Int. Workshop on Machine Learning*, pp. 169–178, 1992.

18) Hansch, C., Li, R., Blaney, J. and Langridge, R., "Comparison of the inhibition of escherichia coli and lactobacillus casei dihydrofolate reductase by 2,4-diamino-5-(substituted-benzyl) pyrimidines: Quantitative structure-activity relationships, x-ray crystallography, and computer graphics in structure-activity analysis," *J. Med. Chem.*, 25, pp. 777–784, 1992.

19) Hau, D.T. and Coiera, E.W., "Learning qualitative models of dynamic systems," *Machine Learning*, 26(2/3), pp. 177–212, 1997.

20) Karalič, A. and Bratko, I., "First order regression," *Machine Learning*, 26(2/3), pp. 147–176, 1997.

21) Kijsirikul, B., Numao, M. and Shimura, M., "Discrimination-based constructive induction of logic programs," in *AAAI92*, AAAI Press/The MIT Press, pp. 44–49, 1992.

22) King, R.D., Muggleton, S., Lewis, R. and Sternberg, M.J.E., "Drug design by machine learning: The use of inductive logic programming to model the structure-activity relationships of trimethoprim analogues binding to dihydrofolate reductase," in *Proc. of the National Academy of Sciences of the USA*, 89(23), pp. 11322–11326, 1992.

23) King, R.D., Srinivasan, A. and Sternberg, M.J.E., "Relating chemical activity to structure: an examination of ILP successes," *New Generation Computing*, 13, pp. 411–433, 1995.

24) Kneller, D., Cohen, F. and Langridge, R., "Improvements in protein secondary structure prediction by an enhanced neural network," *J. Mol. Biol.*, 214, pp. 171–182, 1990.

25) Kovačič, M., "MILP - A stochastic approach to inductive logic programming," in *Proc. Fourth Int. Workshop on Inductive Logic Programming*, GMD-Studien 237, pp. 123–138, 1994.

26) Lavrač, N. and Džeroski, S., *Inductive Logic Programming: Techniques and Applications*, Ellis Horwood, 1994.

27) Michie, D., "Machine learning in the next five years," in *Proc. Third European Working Session on Learning*, Pitman, pp. 107–122, 1988.

28) Mizoguchi, F., Ohwada, H., Daidoji, M. and Shirato, S., "Using inductive logic programming to learn classification rules that identify glaucomatous eyes," in *Intelligent Data Analysis in Medicine and Pharmacology* (N. Lavrač, E. Keravnou, B. Zupan, eds.), Kluwer, pp. 227–242, 1997.

29) Mooney, R.J. and Califf, M.E., "Induction of first-order decision lists: Results on learning the past tense of English verbs," *Journal of Artificial Intelligence Research* 3, pp. 1–24, 1995.

30) Muggleton, S., ed., *Inductive Logic Programming*, Academic Press, 1992.

31) Muggleton, S., "Inverse Entailment and Progol," *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4), 1995.

32) Muggleton, S., King, R.D. and Sternberg, M.J.E., "Protein secondary structure prediction using logic," *Protein Engineering* 7, pp. 647–657, 1992.

33) Muggleton, S. and Feng, C., "Efficient induction of logic programs," in *Proc. First Conference on Algorithmic Learning Theory*, Ohmsha, Tokyo, pp. 368–381, 1990.

34) Muggleton, S. and De Raedt, L., "Inductive logic programming: Theory and methods," *Journal of Logic Programming* 19/20, pp. 629–679, 1994.

35) Muggleton, S., Page, C.D. and Srinivasan, A., "An initial experiment into stereochemistry-based drug design using ILP," in *Proc. 6th Int. Workshop on Inductive Logic Programming* (S. Muggleton, ed.), Stockholm University, Royal Institute of Technology, pp. 245–261, 1996.

36) Numao, M., Kobayashi, M and Sakaniwa, K., "Acquisition of human feelings in music arrangement," in *Proc. IJCAI 97*, Morgan Kaufmann, 1997.

37) Quinlan, J.R., "Learning logical definitions from Relations," *Machine Learning* 5, pp. 239–266, 1990.

38) Rouveirol, C. and Puget, J-F., "A simple solution for inverting resolution," in *Proc. Fourth European Working Session on Learning*, Pitman, pp. 201–210, 1989.

39) Sebag, M. and Rouveirolz, C., "Constraint Inductive Logic Programming," in *Advances in Inductive Logic Programming* (L. De Raedt, ed.), IOS Press, pp. 277–294, 1996.

40) Sommer, E., "Rulebase stratifications: An approach to theory restructuring," in *Proc. Fourth Int. Workshop on Inductive Logic Programming*, GMD-Studien 237, pp. 377-390, 1994.

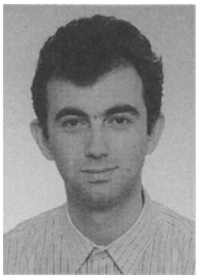
41) Srinivasan, A., Muggleton, S., King, R.D. and Sternberg, M.J.E., "Mutagenesis: ILP experiments in a non-determinate biological domain," in *Proc. Fourth Int. Workshop on Inductive Logic Programming*, GMD-Studien 237, pp. 217–232, 1994.

42) Srinivasan, A. and King, R.D., "Feature construction with inductive logic programming: A study of quantitative predictions of biological activity aided by structural attributes," in *Proc. 6th Int. Workshop on Inductive Logic Programming* (S. Muggleton, ed.), Stockholm University, Royal Institute of Technology, pp. 352–367, 1996.

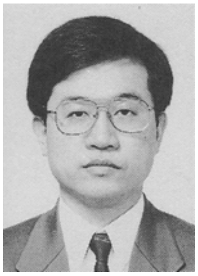
43) Srinivasan, A., King, R.D., Muggleton, S. and Sternberg, M.J.E., "Carcinogenesis prediction using inductive logic programming," in *Intelligent Data Analysis in Medicine and Pharmacology* (N. Lavrač, E. Keravnou, B. Zupan, eds.), Kluwer, pp. 243–260, 1997.

44) Wrobel, S., "An algorithm for multi-relational discovery of subgroups," in *Proc. First European Symposium on Principles of Data Mining and Knowledge Discovery*, Springer, pp. 78–87, 1997.

Nada Lavrač, Ph.D.: She is a senior research associate at the Department of Intelligent Systems, J. Stefan Institute, Ljubljana, Slovenia (since 1978) and a visiting professor at the Klagenfurt University, Austria (since 1987). Her main research interest is in machine learning, in particular inductive logic programming and intelligent data analysis in medicine. She received a BSc in Technical Mathematics and MSc in Computer Science from Ljubljana University, and a PhD in Technical Sciences from Maribor University, Slovenia. She is coauthor of KARDIO: A Study in Deep and Qualitative Knowledge for Expert Systems, The MIT Press 1989, and Inductive Logic Programming: Techniques and Applications, Ellis Horwood 1994, and coeditor of Intelligent Data Analysis in Medicine and Pharmacology, Kluwer 1997. She was the coordinator of the European Scientific Network in Inductive Logic Programming ILPNET (1993–1996) and program cochair of the 8th European Machine Learning Conference ECML'95, and 7th International Workshop on Inductive Logic Programming ILP'97.



Sašo Džeroski, Ph.D.: He is a research associate at the Department of Intelligent Systems, J. Stefan Institute, Ljubljana, Slovenia (since 1989). He has held visiting researcher positions at the Turing Institute, Glasgow (UK), Katholieke Universiteit Leuven (Belgium), German National Research Center for Computer Science (GMD), Sankt Augustin (Germany) and the Foundation for Research and Technology-Hellas (FORTH), Heraklion (Greece). His research interest is in machine learning and knowledge discovery in databases, in particular inductive logic programming and its applications and knowledge discovery in environmental databases. He is co-author of Inductive Logic Programming: Techniques and Applications, Ellis Horwood 1994. He is the scientific coordinator of ILPnet2, The Network of Excellence in Inductive Logic Programming. He was program co-chair of the 7th International Workshop on Inductive Logic Programming ILP'97 and will be program co-chair of the 16th International Conference on Machine Learning ICML'99.



Masayuki Numao, Ph.D.: He is an associate professor at the Department of Computer Science, Tokyo Institute of Technology. He received a bachelor of engineering in electrical and electronics engineering in 1982 and his Ph.D. in computer science in 1987 from Tokyo Institute of Technology. He was a visiting scholar at CSLI, Stanford University from 1989 to 1990. His research interests include Artificial Intelligence, Global Intelligence and Machine Learning. Numao is a member of Information Processing Society of Japan, Japanese Society for Artificial Intelligence, Japanese Cognitive Science Society, Japan Society for Software Science and Technology and AAAI.