Journal of Machine Learning Research 1 (2003) 1-16 Submitted 4/00; Published 10/00

ILP: A Short Look Back and a Longer Look Forward

David Page PAGE@BIOSTAT.WISC.EDU
Dept. of Biostatistics and Medical Informatics
and Dept. of Computer Sciences
University of Wisconsin
1300 University Ave., Rm 5795 Medical Sciences
Madison, WI 53706, USA

Ashwin Srinivasan ASHWIN@COMLAB.OX.AC.UK
Oxford University Computing Laboratory
Wolfson Building, Parks Road
Oxford OX1 3QD, UK

Editor: James Cussens and Alan M. Frisch

Abstract

Inductive logic programming (ILP) is built on a foundation laid by research in machine learning and computational logic. Armed with this strong foundation, ILP has been applied to important and interesting problems in the life sciences, engineering and the arts. This paper begins by briefly reviewing some example applications, in order to illustrate the benefits of ILP. In turn, the applications have brought into focus the need for more research into specific topics. We enumerate and elaborate five of these: (1) novel search methods; (2) incorporation of explicit probabilities; (3) incorporation of special-purpose reasoners; (4) parallel execution using commodity components; and (5) enhanced human interaction. It is our hypothesis that progress in each of these areas can greatly improve the contributions that can be made with ILP; and that, with assistance from research workers in other areas, significant progress in each of these areas is possible.

1. Introduction

Inductive logic programming has its foundations in computational logic, including logic programming, knowledge representation and reasoning, and automated theorem proving. These foundations go beyond the obvious basis in definite clause logic and SLD-resolution. In addition ILP has utilized such theoretical results from computational logic as Lee's Subsumption Theorem (21), Gottlob's Lemma linking implication and subsumption (13), Marcinkowski and Pacholski's result on the undecidability of implication between definite clauses (26), and many others. In addition to utilizing such theoretical results, ILP depends crucially on important advances in logic programming implementations. For example, many of the applications summarized in the next section were possible only because of fast deductive inference based on indexing, partial compilation, etc. as embodied in the best current Prolog implementations. Finally, research in computational logic has yielded numerous important lessons about knowledge representation in logic that have formed the basis for applications. Just as one example, definite clause grammars are central to several ILP applications within both natural language processing and bioinformatics.

In his famous address in 1900 to the International Congress of Mathematicians in Paris, David Hilbert commenced thus (from the English translation in (16)):

> Who of us would not be glad to lift the veil behind which the future lies hidden; to cast a glance at the next advances of our science ... We know that every age has its own problems, which the following age either solves or casts aside as profitless and replaces by new ones. If we would obtain an idea of the probable development of mathematical knowledge in the immediate future, we must let the unsettled questions pass before our minds and look over the problems which the science of today sets and whose solution we expect from the future.

In a far more humble setting, we present here what we believe to be some pressing issues that have arisen from the most challenging ILP applications of today. These are:

1. The development of novel search methods;
2. Techniques for incorporating explicit probabilities into ILP;
3. The use of special-purpose reasoners in ILP;
4. Techniques for parallel execution using commodity components; and
5. Enhancing human-computer interaction to make ILP systems true collaborators with human experts.

It is our belief that adequate solutions to the concomitant problems posed by these issues will greatly improve the quality and type of assistance that can be rendered by ILP systems. Further, we fully expect such solutions are obtainable in the future, with the assistance of research workers from machine learning, algorithm development, computational logic, and experimental psychology.

The rest of the paper is organised as follows. Section 2 gives a brief review of the some the application areas that have motivated the research issues enumerated above. Each of these issues is examined in greater detail in Sections 3–7. Section 8 concludes the paper.

2. Challenging Application Areas for ILP

One of the most important application domains for machine learning in general is bioinformatics, broadly interpreted. This domain is particularly attractive for (1) its obvious importance to society, and (2) the plethora of large and growing data sets. Data sets obviously include the newly completed and available DNA sequences for *C. elegans* (nematode), *Drosophila* (fruitfly), and (depending on one's definitions of "completed" and "available") man. But other data sets include gene expression data (recording the degree to which various genes are expressed as protein in a tissue sample), bio-activity data on potential drug molecules, x-ray crystallography and NMR data on protein structure, and data from novel techniques in proteomics. Applications within bioinformatics include protein struc-

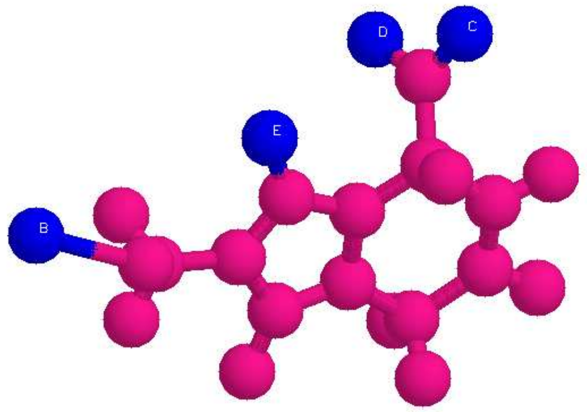

Figure 1: ACE inhibitor number 1 with highlighted 4-point pharmacophore.

ture prediction (33, 42), mutagenicity prediction (20), and pharmacophore discovery (25, 8) (discovery of a 3D substructure responsible for drug activity that can be used to guide the search for new drugs with similar activity). ILP is particularly well-suited for bioinformatics tasks because of its abilities to take into account background knowledge and work directly with structured data. For example, the following is a potential pharmacophore for ACE inhibition (a form of hypertension medication), where the spatial relationships are described through pairwise distances.[1]

```
Molecule A is an ACE inhibitor if:
   molecule A contains a zinc binding site B, and
   molecule A contains a hydrogen acceptor C, and
   the distance between B and C is 7.9 +/- .75 Angstroms, and
   molecule A contains a hydrogen acceptor D, and
   the distance between B and D is 8.5 +/- .75 Angstroms, and
   the distance between C and D is 2.1 +/- .75 Angstroms, and
   molecule A contains a hydrogen acceptor E, and
   the distance between B and E is 4.9 +/- .75 Angstroms, and
   the distance between C and E is 3.1 +/- .75 Angstroms, and
   the distance between D and E is 3.8 +/- .75 Angstroms.
```

Figures 1 and 2 show two different ACE inhibitors with the parts of the pharmacophore highlighted and labeled. The preceding rule was automatically translated directly from

1. Hydrogen acceptors are atoms with a weak negative charge. Ordinarily, zinc-binding would be irrelevant; it is relevant here because ACE is one of several proteins in the body that typically contains an associated zinc ion. The error tolerance on distances was fixed to 0.75 Angstroms based on the recommendation of a domain expert; multiple possible error tolerances can be incorporated into the search if that is desired instead. 1.0 and 0.75 are typical tolerances preferred by chemists.

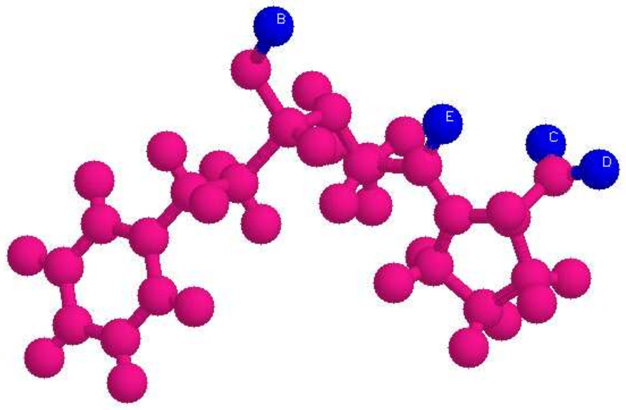

Figure 2: ACE inhibitor number 2 with highlighted 4-point pharmacophore.

logic. It illustrates another strength of ILP, in that logical rules are easily translated into English. If the vocabulary for the rules is meaningful to the domain experts, as in this case, then ILP discoveries are directly comprehensible to humans, at least to domain experts.

Note that the features (acceptors and zinc-binding) are related to one another by distances, and a typical molecule may have many atoms or groups that potentially could play the role of a given feature in the rule. Hence just testing whether a molecule satisfies a rule is itself a constraint-satisfaction problem, where features are variables and distances are constraints. For this reason, ordinary feature based learners (e.g., decision tree algorithms), cannot learn rules of this form unless each possible rule is itself encoded as a single feature. But the problem with that approach is that it leads to millions of features for a typical set of molecules.

A very different type of domain for machine learning is natural language processing (NLP). This domain also includes a wide variety of tasks such as part-of-speech tagging, grammar learning, information retrieval, and information extraction. Arguably, natural language translation (at least, very rough-cut translation) is now a reality—witness for example the widespread use of Altavista's Babelfish (http://babel.altavista.com/). Machine learning techniques are aiding in the construction of information extraction engines that fill database entries from document abstracts or web pages (e.g., (3)). NLP became a major application focus for ILP in particular with the ESPRIT project ILP2. A strength of ILP for NLP is that grammars can be represented as logic programs, so the same algorithms used to learn pharmacophores can be applied to learning grammars or portions of grammars.

A third popular and challenging application area for machine learning is knowledge discovery from large databases with rich data formats, which might contain for example satellite images, audio recordings, movie files, etc. While Dzeroski has shown how ILP applies very naturally to knowledge discovery from ordinary relational databases (7), advances are needed to deal with multimedia databases.

ILP has advantages over other machine learning techniques for all of the preceding application areas. Nevertheless, these applications also highlight the following shortcomings of present ILP technology:

- Techniques such as bigrams and trigrams, or the more complex hidden Markov models, Bayes nets and dynamic Bayes nets, can expressly represent the probabilities inherent in tasks such as part-of-speech tagging, alignment of proteins, robot maneuvering, etc. Few ILP systems need to have such capabilities.

- ILP systems have higher time and space requirements than other machine learning systems, making it difficult to apply them to large data sets. Novel search algorithms and parallel processing need to be explored.

- ILP works well when data and background knowledge are cleanly expressible in first-order logic. But what can be done when databases contain images, audio, movies, etc.? ILP needs to learn lessons from constraint logic programming regarding the incorporation of special-purpose techniques for handling special data formats.

- In scientific knowledge discovery, for example in the domain of bioinformatics, it would be beneficial if ILP systems could collaborate with scientists rather than merely running in batch mode. If ILP does not take this step, other forms of collaborative scientific assistants will be developed, supplanting ILP's position within these domains.

The directions for further research that are discussed in the following sections address these shortcomings, in the same order.

3. Improved/Novel Search Methods

Most ILP algorithms search a lattice of clauses ordered by subsumption. They seek a clause that maximizes some function of the size of the clause and coverage of the clause, i.e. the numbers of positive and negative examples entailed by the clause together with the background theory. Depending upon how they search this lattice, these ILP algorithms are classified as either specific-to-general (based on least general generalization) or general-to-specific (based on refinement). Algorithms are further classified by whether they perform a greedy search, beam search, admissible search, etc. But a large space of possible algorithms still remains unexplored—algorithms that are neither top-down nor bottom-up, nor even necessarily deterministic. For other challenging logic or artificial intelligence tasks outside ILP, great progress has been made in the development of novel search stategies. The best-known case is satisfiability, where GSAT (38) made a substantial improvement over Davis-Putnam, then WalkSAT (37) improved upon GSAT, and where more recently novel versions of Davis-Putnam with rapid random restarts have outperformed WalkSAT (12). Consequently, a promising research direction is to apply novel search strategies such as these to ILP.

ILP algorithms face not one but two difficult search problems. In addition to the search of the lattice of clauses, already described, simply testing the coverage of a clause involves repeated searches for proofs—"if I assume this clause is true, does a proof exist for that example?" Some work on stochastic search in ILP already has been done, and it addressed this latter search problem. Sebag and Rouveirol (36) employed stochastic matching, or theorem proving, and obtained efficiency improvements over Progol in the prediction of mutagenicity, without sacrificing predictive accuracy or comprehensibility. More recently, Botta, Giordana, Saitta, and Sebag have pursued this approach further, continuing to show the benefits of replacing deterministic matching with stochastic matching (11, Botta et al.). But at the center of ILP is the search of the clause lattice. Genetic algorithms have been employed for searching this lattice, but more work on novel search strategies is needed. The remainder of this section briefly outlines several directions for such research.

First, one can easily imagine variants of GSAT and WalkSAT tailored to search a lattice of clauses instead of truth assignments, trying to maximize consistency with a data set rather than clauses satisfied in a Boolean CNF formula. A natural ILP variant of GSAT performs as follows. It first draws a random first-order definite clause, rather than a random truth assignment. Instead of "flipping" the truth assignments of individual variables, its moves involve adding or deleting literals in the clause. The ILP variant of WalkSAT is a very similar algorithm. The difference is that with some probability p the algorithm makes a random move—it randomly selects an efficacious literal to add or delete. An efficacious addition is a literal that, when added, will cause the clause not to cover some negative example; an efficacious deletion is a literal that, when deleted, will permit the clause to cover a previously-uncovered positive example.

We have conducted preliminary experiments using an implementation of these algorithms within the Aleph system. On an artificial domain consisting of random graphs, we find that ILP-WalkSAT outperforms ILP-GSAT. Both algorithms perform better than a routine greedy search, and find useful clauses in cases where it is intractable to use a complete search.

These results are promising, but much more more research can be done. First, the procedures described still search for one clause at a time. To learn multiple clauses, they employ the standard greedy-covering heuristic. Can stochastic searches be formulated that search the space of entire *theories* rather than clauses? Second, in GSAT or WalkSAT scoring a given truth assignment is very fast. In contrast, scoring a clause can be much more time consuming because it involves repeated theorem proving. Therefore, it might be beneficial to combine the ILP GSAT and WalkSAT algorithms with the stochastic theorem proving mentioned earlier. Third, the number of literals that can be built from a language often is infinite, so we cannot test all possible additions of a literal. Our approach has been to base any given iteration of the algorithm on a "bottom clause" built from a "seed example," based on the manner in which the ILP system Progol (30) constrains its search space. Fourth, other types of stochastic search could be tried, such as simulated annealing.

We have noted already that Davis-Putnam has been improved substantially through alternative settings of parameters and *rapid random restarts* (RRR) (12). This success suggests that RRR might also be used in ILP to improve refinement-based searches. A start in this direction has been made very recently, and the results indicate that refinement-based search with RRR is indeed a promising approach (44). Much more research is needed to determine appropriate parameters for such a search, including how rapidly the restarts should occur.

4. Probabilistic Inference: ILP and Bayes Nets

Bayesian networks have largely supplanted traditional rule-based expert systems. Why? Because in task after task artificial intelligence practitioners have realized that probabilities are central. For example, in medical diagnosis few universally true rules exist and few entirely accurate laboratory experiments are available. Instead, probabilities are needed to model the task's inherent uncertainty. Bayes nets are designed specifically to model probability distributions and to reason about these distributions accurately and (in some cases) efficiently. Consequently, in many tasks including medical diagnosis (15), Bayes nets have been found to be superior to rule-based systems. Interestingly, inductive inference, or machine learning, has turned out to be a very significant component of Bayes net reasoning. Inductive inference from data is particularly important for developing or adjusting the conditional probability tables for various network nodes, but also is used in some cases even for developing or modifying the structure of the network itself.

In spite of these advantages, a Bayes net is less expressive than first-order logic, on a par with propositional logic instead. Consequently, while a Bayes net is a graphical representation, it cannot represent relational structures. The only relationships captured by the graphs are conditional dependencies among variables. This failure to capture other relational information is particularly troublesome when using the Bayes net representation in learning. For a concrete illustration, consider the task of pharmacophore discovery. It would be desirable to learn probabilistic predictors, e.g., what is the probability that a given structural change to the molecule fluoxetine (Prozac) will yield an equally effective anti-depressant (specifically, serotonin reuptake inhibitor)? To build such a probabilistic predictor, we might choose to learn a Bayes net from data on serotonin reuptake inhibitors. Unfortunately, while a Bayes net can capture the probabilistic information, it cannot capture the structural properties of a molecule that are predictive of biological activity.

The inability of Bayes nets to capture relational structure is well known and has led to the recent extension to *probabilistic relational models* (PRMs) and the study of learning algorithms for such models (10). Probabilistic relational models are an extension of Bayes nets to multiple relational tables, as in a relational database. Because so many real-world data mining tasks are relational in nature, and hence require multiple relational tables, the power of PRMs has immediately been widely recognized. It is worth bearing in mind, nevertheless, that PRMs fall short of the expressivity of first-order logic, or even of Datalog, and that the learning algorithms are very different from those employed within ILP. An interesting alternative for ILP researchers to examine is learning clauses with probabilities attached. It will be important in particular to examine how such representations and learning algorithms compare with PRMs and PRM learning algorithms. It may well be the case that these closely related research directions can benefit greatly from one another. Several candidate probabilistic logic representations have been proposed and include probabilistic logic programs, Bayesian logic programs, stochastic logic programs, and probabilistic constraint logic programs; Cussens provides a nice survey of these representations (5). Study already has begun into algorithms and applications for learning stochastic logic programs (31) and Bayesian logic programs (18), and these are exciting areas for further work. The first-order representations closest to Bayes nets are the representation of Ngo and Haddawy (35, 34) and the logic programs of Kersting and De Raedt (19). The remain-

```
drug(Molecule,Activity_Level):-
    contains_hydrophobe(Molecule,Hydrophobe),
    contains_basic_nitrogen(Molecule,Nitrogen),
    contains_hydrogen_acceptor(Molecule,Acceptor),
    distance(Molecule,Hydrophobe,Nitrogen,D1),
    distance(Molecule,Hydrophobe,Acceptor,D2),
    distance(Molecule,Nitrogen,Acceptor,D3).
```

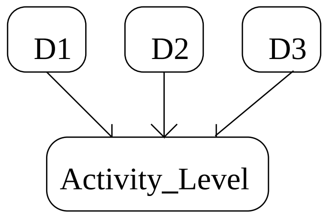

Figure 3: A clause with a Bayes net fragment attached (*conditional probability table* not included). The random variables are *Activity_Level*, *D1*, *D2*, and *D3*. Rather than using a hard range in which the values of *D1*, *D2*, and *D3* must fall, as the pharmacophores described earlier, this new representation allows us to describe a probability distribution over *Activity_Level* in terms of the values of *D1*, *D2*, and *D3*. For example, we might assign higher probabilities to high *Activity_Level* as *D1* gets closer to 3 Angstroms from either above or below. The conditional probability table itself might be a linear regression model, i.e. a linear function of *D1*, *D2*, and *D3* with some fixed variance assumed, or it might be a discretized model, or other.

der of this section points to approaches for, and potential benefits of, learning clauses in the representation of Ngo and Haddawy or a similar representation.

Clauses in the representation of Ngo and Haddawy may contain random variables as well as ordinary logical variables. A clause may contain at most one random variable in any one literal, and random variables may appear in body literals only if a random variable appears in the head. Finally, such a clause also has a Bayes net fragment attached, which may be thought of as a constraint. This fragment has a very specific form. It is a directed graph of node depth two (edge depth one), with all the random variables from the clause body as parents of the random variable from the clause head.[2] Figure 3 provides an example of such a clause as might be learned in pharmacophore discovery (conditional probability table not shown). This clause enables us to specify, through a conditional probability table, how the probability of a molecule being active depends on the particular values assigned to the distance variables $D1$, $D2$, and $D3$. In general, the role of the added constraint in the form of a Bayes net fragment is to define a conditional probability distribution over the random variable in the head, conditional on the values of the random variables in the body. When multiple such clauses are chained together during inference, a larger Bayes net is formed that defines a joint probability distribution over the random variables.

We conjecture that existing ILP algorithms can effectively learn clauses of this form with the following modification. For each clause constructed by the ILP algorithm, collect the positive examples covered by the clause. Each positive example provides a value for the random variable in the head of the clause, and because the example is covered, the

2. This is not exactly the definition provided by Ngo and Haddawy, but it is an equivalent one. Readers interested in deductive inference with this representation are encouraged to see (35, 34).

example together with the background knowledge provides values for the random variables in the body. These values, over all the covered positive examples, can be used as the data for constructing the conditional probability table (conditional probability table) that accompanies the attached Bayes net fragment. When all the random variables are discrete, a simple, standard method exists for constructing conditional probability tables from such data and is described nicely in (14). If some or all of the random variables are continuous, then under certain assumptions again simple, standard methods exist. For example, under one set of assumptions linear regression can be used, and under another naive Bayes can be used. In fact, the work by Srinivasan and Camacho (40) on predicting levels of mutagenicity and the work by Craven and colleagues (4, 3) on information extraction can be seen as special cases of this proposed approach, employing linear regression and naive Bayes, respectively.

While the approach just outlined appears promising, of course it is not the only possible approach and may not turn out to be the best. More generally, ILP and Bayes net learning are largely orthogonal. The former handles relational domains well, while the latter handles probabilities well. And both Bayes nets and ILP have been applied successfully to a variety of tasks. Therefore, it is reasonable to hypothesize the existence and utility of a representation and learning algorithms that effectively capture the advantages of both Bayes net learning and ILP. The space of such representations and algorithms is large, so combining Bayes net learning and ILP is an area of research that is not only promising but also wide open for further work.

5. Special-purpose Reasoning Mechanisms

One of the well-documented success stories of computational logic is constraint logic programming. And one of the reasons for this success is the ability to integrate logic and special purpose reasoners or constraint solvers. Many ILP applications could benefit from the incorporation of special-purpose reasoning mechanisms. Indeed, the approach advocated in Section 3.1 to incorporating probabilities in ILP can be thought of as invoking special purpose reasoners to construct constraints in the form of Bayes net fragments. The work by Srinivasan and Camacho mentioned there uses linear regression to construct a constraint, while the work by Craven and Slattery uses naive Bayes techniques to construct a constraint. The point that is crucial to notice is that ILP requires a "constraint constructor," such as linear regression, in addition to the constraint solver required during deduction. Let's now turn to consideration of tasks where other types of constraint generators might be useful.

Consider the general area of knowledge discovery from databases. Suppose we take the standard logical interpretation of a database, where each relation is a predicate, and each tuple in the relation is a ground atomic formula built from that predicate. Džeroski and Lavrač show how ILP techniques are very naturally suited to this task, if we have an "ordinary" relational database (6). But now suppose the database contains some form of complex objects, such as images. Simple logical similarities may not capture the important common features across a set of images. Instead, special-purpose image processing techniques may be required, such as those described by Leung and colleagues (23, 22). In addition to simple images, special-purpose constraint constructors might be required when applying ILP to movie (e.g. MPEG) or audio (e.g. MP3) data, or other data forms that are becoming ever more commonplace with the growth of multimedia. For example, a fan of the Bach, Mozart, and Brian Wilson would love to be able to enter her/his favorite pieces, have ILP with a constraint generator build rules to describe these favorites, and have the rules suggest other pieces or composers s/he should access. As multimedia data becomes more commonplace, ILP can remain applicable only if it is able to incorporate special- purpose constraint generators.

Frisch and Page (9) have shown that the ordinary subsumption ordering over formulas scales up quite naturally to incorporate constraints. Nevertheless, that work does not address some of the hardest issues, such as how to ensure the efficiency of inductive learning systems based on this ordering and how to design the right types of constraint generators. These questions require much further research involving real-world applications such as multimedia databases.

One final point about special purpose reasoners in ILP is worth making. Constructing a constraint may be thought of as inventing a predicate. Predicate invention within ILP has a long history (32, 43, 45, 29). General techniques for predicate invention encounter the problem that the space of "inventable" predicates is unconstrained, and hence allowing predicate invention is roughly equivalent to removing all bias from inductive learning. While removing bias may sound at first to be a good idea, inductive learning in fact requires bias (27, 28). Special purpose techniques for constraint construction appear to make it possible to perform predicate invention in way that is limited enough to be effective (40, 3).

6. Parallel Execution

Although ILP has numerous advantages over other types of machine learning, including advantages mentioned at the start of the previous section, it has two particularly notable disadvantages—excessive run time and space requirements. Fortunately for ILP, at the same time that larger applications are highlighting these disadvantages, parallel processing "on the cheap" is becoming widespread. Most notable is the widespread use of "Beowulf clusters" (1) and of "Condor pools" (24), arrangements that connect tens, hundreds, or even thousands of personal computers or workstations to permit parallel processing. Admittedly, parallel processing cannot change the order of the time or space complexity of an algorithm. But most ILP systems already use broad constraints, such as maximum clause size, to hold down exponential terms. Rather, the need is to beat back the large constants brought in by large real-world applications.

Yu Wang and David Skillicorn recently developed a parallel implementation of Progol under the Bulk Synchronous Parallel model and claim superlinear speedup from this implementation (39). The remainder of this section describes how large-scale parallelism can be achieved very simply in a complete general-to-specific search ILP algorithm. From this discussion, one can imagine more interesting approaches for other types of general-to-specific such as greedy search.

The ideal in parallel processing is a decrease in processing time that is a linear function, with a slope near 1, of the number of processors used. (In some rare cases it is possible to achieve superlinear speed-up.) The barriers to achieving the ideal are (1) overhead in communication among processes and (2) competition for resources among processes. Therefore, a good parallel scheme is one where the processes are relatively independent of one another and hence require little communication or resource sharing. The key observation in the design of the parallel ILP scheme is that two competing hypotheses can be tested against the data completely independently of one another. Therefore the approach advocated here is to distribute the hypothesis space among different processors for testing against the data. These processors need not communicate with one another during testing, and they need not write to a shared memory space.

In more detail, for complete search a parallel ILP scheme can employ a master-worker design, where the master assigns different segments of the hypothesis space to workers that then test hypotheses against the data. Workers communicate back to the master all hypotheses achieving a pre-selected minimum valuation score (e.g. 95 % accuracy) on the data. As workers become free, the master continues to assign new segments of the space until the entire space has been explored. The only architectural requirements for this approach are (1) a mechanism for communication between the master and each worker and (2) read access for each worker to the data. Because data do not change during a run, this scheme can easily operate under either a shared memory or message passing architecture; in the latter, we incur a one-time overhead cost of initially communicating the data to each worker. The only remaining overhead, on either architecture, consists of the time spent by the master and time for master-worker communication. In "needle in a haystack" domains, which are the motivation for complete search, one expects very few hypotheses to be communicated from workers to the master, so overhead for the communication of results will be low. If it also is possible for the master to rapidly segment the hypothesis space in such a way that the segments can be communicated to the workers succinctly, then overall overhead will be low and the ideal of linear speed-up can be realized. One implementation of this approach has, in fact, already been tested on the pharmacophore discovery task mentioned in the introduction (17).

Undoubtedly there are a variety of other complete search, or exact, parallel schemes that can be implemented, and the investigation of such schemes is a key area for further research. For all such schemes, two crucial questions should be answered. First, under what conditions is the new parallel scheme faster than existing ones? Second, are the solutions returned by the parallel complete search significantly better than those returned by a stochastic search? This second question brings us back to our second research direction, that of stochastic ILP algorithms. Of course, it is not necessary to choose between stochastic and parallel algorithms. The stochastic algorithms proposed earlier can themselves be implemented on a parallel processor, in the simplest case by replacing restarts by independent searches running at the same time. How will the results of parallel stochastic searches compare with those of parallel complete searches, if both searches are provided with the same number of processors and the same amount of time?

7. Interaction with Human Experts

To discover new knowledge from data in fields such as telecommunications, molecular biology, or pharmaceuticals, it would be beneficial if a machine learning system and a human expert could act as a team, taking advantage of the computer's speed and the expert's knowledge and skills. ILP systems have three properties that make them natural candidates for collaborators with humans in knowledge discovery:

Declarative Background Knowledge ILP systems can make use of declarative background knowledge about a domain in order to construct hypotheses. Thus a collaboration can begin with a domain expert providing the learning system with general knowledge that might be useful in the construction of hypotheses. Most ILP systems also permit the expert to define the hypothesis space using additional background knowledge, in the form of a *declarative bias.*

Natural descriptions of structured examples Feature-based learning systems require the user to begin by creating features to describe the examples. Because many knowledge discovery tasks involve complex structured examples, such as molecules, users are forced to choose only composite features such as molecular weight—thereby losing information—or to invest substantial effort in building features that can capture structure (see (41) for a discussion in the context of molecules). ILP systems allow a structured example to be described naturally in terms of the objects that compose it, together with relations among those objects. The 2-dimensional structure of a molecule can be represented directly using its atoms as the objects and bonds as the relations; 3-dimensional structure can be captured by adding distance relations.

Human-Comprehensible Output ILP systems share with propositional–logic learners the ability to present a user with declarative, comprehensible rules as output. Some ILP systems can return rules in English along with visual aids. For example, the pharmacophore description and corresponding figures in Section 2 were generated automatically by Progol.

Despite the useful properties just outlined, ILP systems—like other machine learning systems—have a number of shortcomings as collaborators with humans in knowledge discovery. One shortcoming is that most ILP systems return a single theory based on heuristics, thus casting away many clauses that might be interesting to a domain expert. But the only currently existing alternative is the version space approach, which has unpalatable properties that include inefficiency, poor noise tolerance, and a propensity to overwhelm users with too large a space of possible hypotheses. Second, ILP systems cannot respond to a human expert's questions in the way a human collaborator would. They operate in simple batch mode, taking a data set as input, and returning a hypothesis on a take-it-or-leave-it basis. Third, ILP systems do not question the input data in the way a human collaborator would, spotting surprising (and hence possibly erroneous) data points and raising questions about them. Some ILP systems will flag mutually inconsistent data points but to our knowledge none goes beyond this. Fourth, while a human expert can provide knowledge-rich forms of hypothesis justification, for example relating a new hypothesis to existing beliefs, ILP systems merely provide accuracy estimates as the sole justification.

To build upon ILP's strengths as a technology for human-computer collaboration in knowledge discovery, the above shortcomings should be addressed. ILP systems should be extended to display the following capabilities.

1. Maintain and summarize alternative hypotheses that explain or describe the data, rather than providing a single answer based on a general-purpose heuristic.

2. Propose to human experts practical sequences of experiments to refine or distinguish between competing hypotheses. Significant advances in the automatic proposal of experiments by an ILP system have been made recently by Bryant and colleagues (2).

3. Provide non-numerical justification for hypotheses, such as relating them to prior beliefs or illustrative examples (in addition to providing numerical accuracy estimates).

4. Answer an expert's questions regarding hypotheses.

5. Consult the expert regarding anomalies or surprises in the data.

In fact, the other four directions already discussed in this paper already go a long way toward capabilities 1 and 3. Both stochastic search and parallel search technologies make it possible to find a potentially large number of alternative hypotheses more efficiently, thus helping to provide capability 1. The ability to provide probability distributions over the bindings for variables in competing hypotheses can potentially provide much more information about these hypotheses. This additional information can be useful in justifying one hypothesis over another. As human experts look more closely at hypotheses, and ask more details about how they fit the data (beyond a simple accuracy number), again these probability distributions can provide further insight.

Work relevant to capability 2 for ILP has been done recently with a goal very different from human-computer interaction. Bryant and colleagues have developed a system to automatically propose *and execute* experiments related to yeast metabolism (2). The system contains an ILP system interfaced with a robot to perform experiments. Each experiment tests whether a particular "knock-out" strain of yeast will grow on a particular medium. The "knock-outs" are variants of yeast, each with a single gene altered so that its functionality is lost to the organism. The goal of this work is to automatically induce a logical model for a portion of yeast metabolism. Although the spirit of this work is virtually the opposite of human-computer interaction, the approach to experiment proposal is relevant for human-computer interaction.

Addressing human-computer interface issues obviously requires a variety of logical and artificial intelligence expertise. Thus contributions from other areas of artificial intelligence and computational logic, such as the study of logical agents, will be vital.

8. Conclusions

ILP has attracted great interest within the machine learning and artificial intelligence communities at large because of its logical foundations, its ability to utilize background knowledge and structured data representations, and its comprehensible results. But most of all, the interest has come from ILP's application successes. Nevertheless, ILP needs further advances to maintain this record of success, and these advances require further contributions from other areas of computational logic. System builders and parallel implementation experts are needed if the ILP systems of the next decade are to scale up to the next generation of data sets, such as those being produced by Affymetrix's gene expression microarrays and Celera's shotgun approach to DNA sequencing. Research workers on probability and logic are required if ILP is to avoid being supplanted by the next generation of extended Bayes net learning systems. Experts on constraint satisfaction and constraint logic programming have the skills necessary to bring successful stochastic search techniques to ILP and to allow ILP techniques to extend to multimedia databases. Paraphrasing the closing words of Hilbert in his 1900 address: that ILP may completely fulfil its high mission, may the next decade bring gifted masters and many zealous and enthusiastic disciples!

Acknowledgements

The first author was supported in part by NSF grant 9987841, by the DARPA EELD program, and by grants from the University of Wisconsin Graduate School and Medical School.

References

[1] D. Becker, T. Sterling, D. Savarese, E. Dorband, U. Ranawake, and C. Packer. Beowulf: A parallel workstation for scientific computation. In *Proceedings of the 1995 International Conference on Parallel Processing (ICPP)*, pages 11–14, 1995.

[Botta et al.] M. Botta, A. Giordana, L. Saitta, and M. Sebag. Relational learning as search in a critical region. *Journal of Machine Learning Research.*

[2] C. Bryant, S. Muggleton, S. Oliver, D Kell, P. Reiser, and R. King. Combining inductive logic programming, active learning and robotics to discover the function of genes. *Electronic Transactions in Artificial Intelligence*, 5-B1(012):1–36, 2001.

[3] M. Craven and J. Kumlien. Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 77–86, Heidelberg, Germany, 1999. AAAI Press.

[4] M. Craven and S. Slattery. Combining statistical and relational methods for learning in hypertext domains. In *Proceedings of the Eighth International Conference on Inductive Logic Programming (ILP-98)*, pages 38–52. Springer Verlag, 1998.

[5] J. Cussens. Loglinear models for first-order probabilistic reasoning. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, pages 126–133. Stockholm, Sweden, 1999.

[6] S. Džeroski and N. Lavrač. An introduction to inductive logic programming. In S. Džeroski and N. Lavrač, editors, *Relational Data Mining*, pages 48–71. Springer, Berlin, 2001.

[7] S. Dzeroski. Inductive logic programming and knowledge discovery in databases. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining.* 1996.

[8] P. Finn, S. Muggleton, D. Page, and A. Srinivasan. Discovery of pharmacophores using Inductive Logic Programming. *Machine Learning*, 30:241–270, 1998.

[9] A. M. Frisch and C. D. Page. Building theories into instantiation. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, 1995.

[10] L. Getoor, N. Friedman, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In S. Dzeroski and N. Lavrac, editors, *Relational Data Mining*, chapter 13, pages 307–335. Springer, Berlin, 2001.

[11] A. Giordana, L. Saitta, M. Sebag, and M. Botta. Analyzing relational learning in the phase transition framework. In P. Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 311–318, Stanford, 2000. Morgan Kaufmann.

[12] C. Gomes, B. Selman, N. Crato, and H. Kautz. Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *Journal of Automated Reasoning*, 24:67–100, 2000.

[13] G. Gottlob. Subsumption and implication. *Information Processing Letters*, 24(2): 109–111, 1987.

[14] D. Heckerman. A tutorial on learning with bayesian networks. Microsoft Technical Report MSR-TR-95-06, 1995.

[15] D. Heckerman, E. Horvitz, and B. Nathwani. Toward normative expert systems, part I: The pathfinder project. *Methods of Information in Medicine*, 31:90–105, 1992.

[16] D. Hilbert. Mathematical problems. *Bulletin of the American Mathematical Society*, 8:437–479, 1902. English translation of original German, provided by Mary Winston.

[17] A.H. Kamal, J. Graham, and C.D. Page. An approach to parallel data mining for pharmacophore discovery. In *Proceedings of the Tenth International Conference on Intelligent Systems*, pages 100–103, Washington, D.C., June 2001.

[18] K. Kersting and L. De Raedt. Towards combining inductive logic programming and bayesian networks. In *Proceedings of the Eleventh International Conference on Inductive Logic Programming*, pages 118–137. Berlin: Springer LNAI 2157, 2001.

[19] K. Kersting, L. De Raedt, and S. Kramer. Interpreting bayesian logic programs. In L. Getoor and D. Jensen, editors, *Proceedings of the AAAI-2000 Workshop on Learning Statistical Models from Relational Data*, Austin, Texas, 2000. AAAI Press, Technical Report WS-00-06.

[20] R. King, S. Muggleton, A. Srinivasan, and M. Sternberg. Structure-activity relationships derived by machine learning: the use of atoms and their bond connectives to predict mutagenicity by inductive logic programming. *Proceedings of the National Academy of Sciences*, 93:438–442, 1996.

[21] C. Lee. *A completeness theorem and a computer program for finding theorems derivable from given axioms.* PhD thesis, University of California, Berkeley, 1967.

[22] T. Leung, M. Burl, and P. Perona. Probabilistic affine invariants for recognition. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, 1998.

[23] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision*, 43(1):29–44, 2001.

[24] M. Litzkow, M. Livny, and M. Mutka. Condor—a hunter of idle workstations. In *Proceedings of the International Conference on Distributed Computing Systems*, pages 104–111, 1988.

[25] N. Marchand-Geneste, K. Watson, B. Alsberg, and R. King. A new approach to pharmacophore mapping and qsar analysis using inductive logic programming. application to thermolysin inhibitors and glycogen phosphorylase b inhibitors. *Journal of Medicinal Chemistry*, 45(2):399–409, January 2002.

[26] J. Marcinkowski and L. Pacholski. Undecidability of the horn-clause implication problem. In *Proceedings of the 33rd IEEE Annual Symposium on Foundations of Computer Science*, pages 354–362. IEEE, 1992.

[27] T.M. Mitchell. The need for biases in learning generalizations. Technical Report CBM-TR-117, Department of Computer Science, Rutgers University, 1980.

[28] T.M. Mitchell. Generalisation as search. *Artificial Intelligence*, 18:203–226, 1982.

[29] S. Muggleton. Predicate invention and utilization. *Journal of Experimental and Theoretical Artificial Intelligence*, 6(1):127–130, 1994.

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

[31] S. Muggleton. Learning stochastic logic programs. In *Proceedings of the AAAI2000 Workshop on Learning Statistical Models from Relational Data*. AAAI, 2000.

[32] S. Muggleton and W. Buntine. Machine invention of first-order predicates by inverting resolution. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 339–352. Kaufmann, 1988.

[33] S. Muggleton, R. King, and M. Sternberg. Protein secondary structure prediction using logic-based machine learning. *Protein Engineering*, 5(7):647–657, 1992.

[34] L. Ngo and P. Haddawy. Probabilistic logic programming and bayesian networks. *Algorithms, Concurrency, and Knowledge: LNCS 1023*, pages 286–300, 1995.

[35] L. Ngo and P. Haddawy. Answering queries from context-sensitive probabilistic knowledge bases. *Theoretical Computer Science*, 171:147–177, 1997.

[36] M. Sebag and C. Rouveirol. Tractable induction and classification in FOL. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, pages 888–892. Nagoya, Japan, 1997.

[37] B. Selman, H. Kautz, and B. Cohen. Noise strategies for improving local search. In *Proceedings of the Twelfth National Conference on Artificial Intelligence.* AAAI Press, 1994.

[38] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 440–446. AAAI Press, 1992.

[39] D. Skillicorn and Y. Wang. Parallel and sequential algorithms for data mining using inductive logic. *Knowledge and Information Systems*, 3(4):405–421, 2001.

[40] A. Srinivasan and R.C. Camacho. Numerical reasoning with an ILP system capable of lazy evaluation and customised search. *Journal of Logic Programming*, 40:185–214, 1999.

[41] A. Srinivasan, S. Muggleton, R. King, and M. Sternberg. Theories for mutagenicity: a study of first-order and feature based induction. *Artificial Intelligence*, 85(1,2):277–299, 1996.

[42] M. Turcotte, S. Muggleton, and M. Sternberg. Application of inductive logic programming to discover rules governing the three-dimensional topology of protein structures. In *Proceedings of the Eighth International Conference on Inductive Logic Programming (ILP-98)*, pages 53–64. Springer Verlag, 1998.

[43] R. Wirth and P. O'Rorke. Constraints on predicate invention. In *Proceedings of the 8th International Workshop on Machine Learning*, pages 457–461. Kaufmann, 1991.

[44] F. Zelezny, A. Srinivasan, and D. Page. Lattice-search runtime distributions may be heavy-tailed. In S. Matwin, editor, *Proceedings of the Twelfth International Conference on Inductive Logic Programming.* Springer-Verlag, 2002.

[45] J. Zelle and R. Mooney. Learning semantic grammars with constructive inductive logic programming. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 817–822, San Mateo, CA, 1993. Morgan Kaufmann.