Chapter 14

LOGIC-BASED MACHINE LEARNING

Stephen Muggleton and Flaviu Marginean
Department of Computer Science
University of York
Heslington, York, YO1 5DD
United Kingdom

Abstract The last three decades has seen the development of Computational Logic techniques within Artificial Intelligence. This has led to the development of the subject of Logic Programming (LP), which can be viewed as a key part of Logic-Based Artificial Intelligence. The subtopic of LP concerned with Machine Learning is known as "Inductive Logic Programming" (ILP), which again can be broadened to Logic-Based Machine Learning by dropping Horn clause restrictions. ILP has its roots in the ground-breaking research of Gordon Plotkin and Ehud Shapiro. This paper provides a brief survey of the state of ILP applications, theory and techniques.

Keywords: Inductive logic programming, machine learning, scientific discovery, protein prediction, learning of natural language

1 INTRODUCTION

As envisaged by John McCarthy in 1959 (McCarthy, 1959), Logic has turned out to be one of the key knowledge representations for Artificial Intelligence research. Logic, in the form of the First-Order Predicate Calculus provides a representation for knowledge which has a clear semantics, together with well-studied sound rules of inference. Interestingly, Turing (Turing, 1950) and McCarthy (McCarthy, 1959) viewed a combination of Logic and Learning as being central to the development of Artificial Intelligence research. This article is concerned with the topic of Logic-Based Machine Learning (LBML). The modern study of LBML has largely been involved with the study of the learning of logic programs, or Inductive Logic Programming (ILP) (Muggleton, 1991). Many of the foundational results of both Gordon Plotkin (Plotkin, 1971) and Ehud Shapiro (Shapiro, 1983) are still central to the study and conceptual framework of ILP. This is true despite the fact that Plotkin did not employ Horn clause restrictions.

ILP is a general form of Machine Learning which involves the construction of logic programs from examples and background knowledge. The subject has inherited its logical tradition from Logic Programming and its empirical orientation from Machine Learning. Robert Kowalski (Kowalski, 1980) famously described Logic Programming (LP) with the following equation.

$$\text{LP} = \text{Logic} + \text{Control}$$

The equation emphasizes the role of the programmer in providing sequencing control when writing Prolog programs. In a similar spirit we might describe ILP as follows.

$$\text{ILP} = \text{Logic} + \text{Statistics} + \text{Computational Control}$$

The logical part of ILP is related to the formation of hypotheses while the statistical part is related to evaluating their degree of belief. As in LP the Computational Control part is related to the sequencing of search carried out when exploring the space of hypotheses.

The structure of the paper is as follows. Section 2 introduces the key elements of ILP, provides a formal framework (Section 2.1) for the later discussion and discusses how Bayesian inference (Section 2.3) is used as a preference mechanism. Section 3 describes applications of ILP to problems related to discovery of biological function. ILP has a strong potential for being applied to Natural Language Processing (NLP). The resultant research area of Learning Language in Logic (LLL) is described in Section 4 together with some encouraging preliminary results. Conclusions concerning ongoing ILP research are given in Section 5.

2 ILP

ILP algorithms take examples E of a concept (such as a protein family) together with background knowledge B (such as a definition of molecular dynamics) and construct a hypothesis h which explains E in terms of B. For example, in the protein fold domains (Section 3.2.2), E might consist of descriptions of molecules separated into positive and negative examples of a particular fold (overall protein shape). This is exemplified in Figure 14.1 for the fold "4-helical-up-and-down-bundle". A possible hypothesis h describing this class of proteins is shown in Figure 14.2. The hypothesis is a definite clause consisting of a *head* (fold(..,..)) and a *body* (the conjunction length(..), .. helix(..)). In this case "fold" is the predicate involved in the examples and hypothesis, while "length", "position", etc. are defined by the background knowledge. A logic program is simply a set of such definite clauses. Each of E, B and h are logic programs.

In the context of knowledge discovery a distinct advantage of ILP over black box techniques, such as neural networks, is that a hypotheses such as that shown in Figure 14.2 can, in a straightforward manner, be made readable by translating it into the following piece of English text.

> The protein P has fold class "Four-helical up-and-down bundle" if it contains a long helix H1 at a secondary structure position between 1 and 3, and H1 is followed by a second helix H2.

Such explicit hypotheses are required for experts to be able to use them within the familiar human scientific discovery cycle of debate, criticism and refutation.

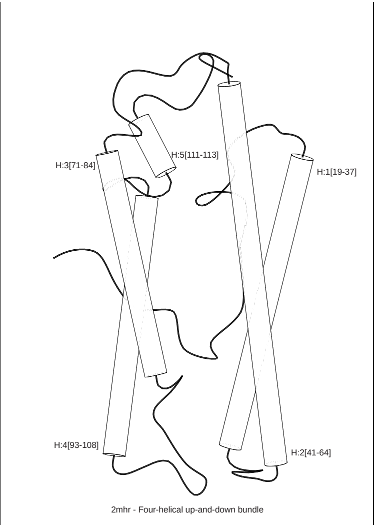
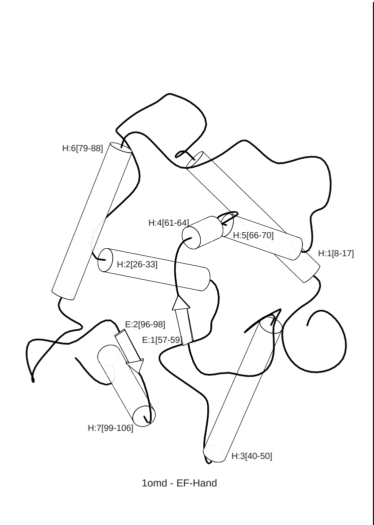
Positive	Negative
	

Figure 14.1 A positive and a negative example of the protein fold "4-helical-up-and-down-bundle". 3-D arrangement of secondary structure units is shown for α-helices (cylinders) and β-sheets (arrows). Each secondary structure unit is labelled according to the index of its first and last amino acid residue.

```
fold('Four-helical up-and-down bundle',P) :-
    helix(P,H1),
    length(H1,hi),
    position(P,H1,Pos),
    interval(1≤ Pos ≤ 3),
    adjacent(P,H1,H2),
    helix(P,H2).
```

Figure 14.2 An hypothesised definite clause for 4-helical-up-and-down-bundles

2.1 FORMAL FRAMEWORK FOR ILP

The normal framework for ILP (Muggleton and De Raedt, 1994; Nienhuys-Cheng and de Wolf, 1997) is as follows. As exemplified by the protein folds problem, described in the last sub-section, the learning system is provided with background knowledge B, positive examples E^+ and negative examples E^- and constructs an hypothesis h. B, E^+ E^- and h are each logic programs. A logic program (Lloyd, 1987) is a set of definite clauses each having the form

$$h \leftarrow b_1, .., b_n$$

where h is an atom and $b_1, .., b_n$ are atoms. Usually E^+ and E^- consist of ground clauses, those for E^+ being definite clauses with empty bodies and those for E^- being clauses with head 'false' and a single ground atom in the body.

In the text below the logical symbols used are: $\wedge$ (logical and), $\vee$ (logical or), $\models$ (logical entailment), $\Box$ (Falsity). The conditions for construction of h are as follows.

Necessity: $B \not\models E^+$

Sufficiency: $B \wedge h \models E^+$

Weak consistency: $B \wedge h \not\models \Box$

Strong consistency: $B \wedge h \wedge E^- \not\models \Box$

Note that neither *sufficiency* nor *strong consistency* are required for systems that deal with noise. The four conditions above capture *all* the logical requirements of an ILP system. However, for any B and E there will generally be many h's which satisfy these conditions. Statistical preference is often used to distinguish between these hypotheses (see Section 2.3). Both *Necessity* and *Consistency* can be checked using a theorem prover. Given that all formulae involved are definite, the theorem prover used need be nothing more than a Prolog interpreter, with some minor alterations, such as iterative deepening, to ensure logical completeness.

2.2 DERIVING ALGORITHMS FROM THE SPECIFICATION OF ILP

The *sufficiency* condition captures the notion of generalizing examples relative to background knowledge. A theorem prover cannot be directly applied to derive h from B and E^+. However, by simple application of the Deduction Theorem the *sufficiency* condition can be rewritten as follows.

Sufficiency*: $B \wedge \overline{E^+} \models \overline{h}$

This simple alteration has a profound effect. The negation of the hypothesis can now be deductively derived from the negation of the examples together with the background knowledge. This is true no matter what form the examples take and what form the hypothesis takes. This approach of turning an inductive problem into one of deduction is called *inverse entailment* (Muggleton, 1995). Methods for ensuring completeness of inverse entailment have been a subject of debate recently (Yamamoto, 1997; Muggleton, 1998).

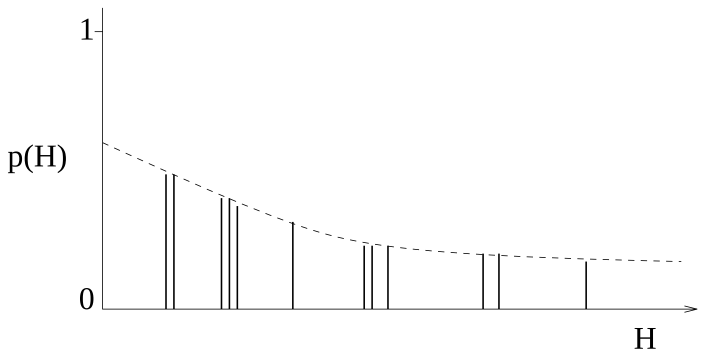


Figure 14.3 Prior over hypotheses. Hypotheses are enumerated in order of descending prior probability along the X-axis. The Y-axis is the probability of individual hypotheses. Vertical bars represent hypotheses consistent with E.

2.3 BAYESIAN FRAMEWORK

It is not sufficient to specify the ILP framework in terms of the logical relationships which must hold between E, B and h. For any given E and B there will be many (possibly infinitely many) choices for h. Thus a technique is needed for defining a preference over the various choices for h. One approach to doing so involves defining a Bayesian prior probability distribution over the learner's hypothesis space (Muggleton, 1994a). This is illustrated in Figure 14.3 [1]. According to Bayes' theorem the hypothesis (h_{MAP}) with maximum posterior probability in hypothesis space H is as follows.

$$\begin{aligned} h_{MAP} &= \mathrm{argmax}_{h\in H} P(h|E) \\ &= \mathrm{argmax}_{h\in H} \frac{P(E|h)P(h)}{p(E)} \\ &= \mathrm{argmax}_{h\in H} P(E|h)P(h) \end{aligned}$$

Within ILP Bayesian approaches (Muggleton, 1994b) have been used to investigate the problem of learning from positive examples only (Muggleton, 2000) and issues related to predicate invention (Khan et al., 1998) (a relational form of feature construction). Learning from positive examples and predicate invention are important in both natural language domains (see Section 4) and in problems involving scientific discovery (see Section 3).

3 DISCOVERY OF BIOLOGICAL FUNCTION

Understanding of a variety of metabolic processes is at the center of drug development within the pharmaceutical industry. Each new drug costs hundreds of millions of pounds to develop. The majority of this cost comes

[1]Note that in the case of the potentially infinite hypothesis space employed in ILP, it is not possible to have a uniform distribution which assigns non-zero prior probabilities to all hypotheses

from clinical tests on efficacy and side-effects. The increasing supply of data both from the human genome project and existing drug databases is producing increasing interest in computational techniques which could reduce drug development costs by supporting automated discovery of biological functions.

Biological functions are regulated by the docking of small molecules (ligands) with sites on large molecules (proteins). Drugs, such as beta-blockers, mimic natural small molecules, such as adrenaline. Effectiveness of drugs depends on the correct shape and charge distribution of ligands. Thus beta-blockers inhibit the binding of adrenaline, and so stop over-stimulation of heart muscle in patients prone to heart attacks.

Results on scientific discovery applications of ILP are separated below between those related to small molecules (such as ligands) and those related to proteins.

3.1 SMALL MOLECULES

3.1.1 Structure-activity prediction. The majority of pharmaceutical R&D is based on finding slightly improved variants of patented active drugs. This involves laboratories of chemists synthesising and testing hundreds of compounds almost at random. The average cost of developing a single new drug is around $300 million. In (King et al., 1992) it was shown that the ILP system Golem (Muggleton and Feng, 1992) was capable of constructing rules which accurately predict the activity of untried drugs. Rules were constructed from examples of drugs with known medicinal activity. The accuracy of the rules was found to be slightly higher than traditional statistical methods. More importantly the easily understandable rules provided insights which were directly comparable to the relevant literature concerning the binding site of dihydrofolate reductase.

3.1.2 Mutagenesis. In (King et al., 1996; Srinivasan et al., 1996) the ILP system Progol (Muggleton, 1995) was used to predict the mutagenicity of chemical compounds taken from a previous study in which linear regression had been applied. Progol's predictive accuracy was equivalent to regression on the main set of 188 compounds and significantly higher (85.7% as opposed to 66.7%) on 44 compounds which had been discarded by the previous authors as unpredictable using regression. Progol's single clause solution for the 44 compounds was judged by the domain experts to be a new structural alert for mutagenesis.

3.1.3 Pharmacophores. In a series of "blind tests" in collaboration with the pharmaceutical company Pfizer UK, Progol was shown (Finn et al., 1998) capable of re-discovering a 3D description of the binding sites (or pharmacophores) of ACE inhibitors (a hypertension drug) and an HIV-protease inhibitor (an anti-AIDS drug).

3.1.4 Carcinogenicity. Progol was entered into a world-wide carcinogenicity prediction competition run by the National Toxicology Program (NTP) in the USA. Progol was trained on around 300 available compounds, and made use of its earlier rules relating to mutagenicity. In the first round of the

Method	Type	Accuracy	P
Ashby†	Chemist	0.77	0.29
Progol	ILP	0.72	1.00
RASH†	Biological potency analysis	0.72	0.39
TIPT†	Propositional ML	0.67	0.11
Bakale	Chemical reactivity analysis	0.63	0.09
Benigni	Expert-guided regression	0.62	0.02
DEREK	Expert system	0.57	0.02
TOPKAT	Statistical discrimination	0.54	0.03
CASE	Statistical correlation analysis	0.54	< 0.01
COMPACT	Molecular modeling	0.54	0.01
Default	Majority class	0.51	0.01

Figure 14.4 Comparative accuracies on the first round of the Predictive Toxicology Evaluation (PTE-1). Here P represents the binomial probability that Progol and the corresponding toxicity prediction method classify the same proportion of examples correctly. The "Default" method predicts all compounds to be carcinogenic. Methods marked with a † have access to short-term in vivo rodent tests that were unavailable to other methods. Ashby and RASH also involve some subjective evaluation to decide on structural alerts.

competition Progol produced the highest predictive accuracy of any automatic system entered (Srinivasan et al., 1997) (see Figure 14.4).

3.2 PROTEINS

3.2.1 Protein secondary structure prediction.. In (Muggleton et al., 1992) Golem was applied to one of the hardest open problems in molecular biology. The problem is as follows: given a sequence of amino acid residues, predict the placement of the main three dimensional sub-structures of the protein. The problem is of great interest to pharmaceutical companies involved with drug design. For this reason, over the last 20 years many attempts have been made to apply methods ranging from statistical regression to decision tree and neural net learning to this problem. Published accuracy results for the general prediction problem have ranged between 50 and 60%, very close to majority-class prediction rates. In our investigation it was found that the ability to make use of background knowledge from molecular biology, together with the ability to describe structural relations boosted the predictivity for a restricted sub-problem to around 80% on an independently chosen test set.

3.2.2 Discovery of fold descriptions. Protein shape is usually described at various levels of abstraction. At the lower levels each family of proteins contains members with high sequence similarity. At the most abstract level folds describe proteins which have similar overall shape but are very different at the sequence level. The lack of understanding of shape determination has made protein fold prediction particularly hard. However, although there are around 300 known folds, around half of all known proteins are members of the 20 most populated folds. In (Turcotte et al., 1998) Progol was applied to discover rules governing the 20 most populated protein folds. The assignment to folds was taken from the SCOP (Structural Classification of Proteins) database (Brenner et al., 1996). Progol was used to learn rules for the five most populated folds of the four classes (alpha/alpha, beta/beta, alpha/beta and alpha+beta). The rules had an average cross-validated accuracy of $75 \pm 9\%$. The rules identified known features of folds. For instance, according to one rule the NAD binding fold where a short loop between the first beta-strand and alpha-helix is required to bind to biological cofactor molecule NAD. A questionnaire, designed by Mike Sternberg, requested named folds to be paired with fold descriptions generated by Progol. The questionnaire was sent to a selection of the world's top protein scientists. Progol successfully identified structural signatures of protein folds that were only know by the world's top expert (Dr Murzin, Cambridge).

4 LEARNING LANGUAGE IN LOGIC

The telecommunications and other industries are investing substantial effort in the development of natural language grammars and parsing systems. Applications include information extraction; database query (especially over the telephone); tools for the production of documentation; and translation of both speech and text. Many of these applications involve not just parsing, but the production of a semantic representation for a sentence.

4.1 WHY IS ILP GOOD FOR NLP?

Hand development of such grammars is very difficult, requiring expensive human expertise. It is natural to turn to machine learning for help in automatic support for grammar development. The currently dominant paradigm in grammar learning is statistically-based ((Magerman, 1995; Collins, 1996; Briscoe and Carroll, 1993; Krotov et al., 1994; Krotov et al., 1997)). This work is all, with a few recent small-scale exceptions, focussed on syntactic or lexical properties. No treatment of semantics or contextual interpretation is possible because there are no annotated corpora available of sufficient size. The aim of statistical language modeling is, by and large, to achieve wide coverage and robustness. The necessary trade-off is that depth of analysis cannot also be achieved. Statistical parsing methods do not deliver semantic representations capable of supporting full interpretation. Traditional rule-based systems, on the other hand, achieve the necessary depth of analysis, but at the sacrifice of robustness: hand-crafted systems do not easily extend to new types of text or application.

In this paradigm disambiguation is addressed by associating statistical preferences, derived from an annotated training corpus, with particular syntactic or semantic configurations and using those numbers to rank parses. While this can be effective, it demands large annotated corpora for each new application, which are costly to produce. There is presumably an upper limit

What is the highest point of the state with the largest area?
```
answer(P, (high-point(S,P), largest(A, (state(S), area(S,A))))).
```

What are the major cities in Kansas?
```
answer(C, (major(C), city(C), loc(C,S),
equal(S,stateid(kansas)))).
```

Figure 14.5 Form of examples

on the accuracy of these techniques, since the variety of language means that it is always possible to express sentences in a way that will not have been encountered in training material.

The alternative method for disambiguation and contextual resolution is to use an explicit domain theory which encodes in a set of logical axioms the relevant properties of the domain. While this has been done for small scale domains (e.g. (Hobbs et al., 1993)), the currently fashionable view is that it is impractical for complex domains because of the unmanageably large amount of hand-coded knowledge that would be required. However, if a large part of this domain knowledge could be acquired (semi-)automatically, this kind of practical objection could be met. From the NLP point of view the promise of ILP is that it will be able to steer a mid-course between these two alternatives of large scale, but shallow levels of analysis, and small scale, but deep and precise analyses. ILP should produce a better ratio between breadth of coverage and depth of analysis.

4.2 HOW CAN ILP BE USED FOR NLP?

In grammar learning the logical theory to be synthesised consists of grammar rules together with semantic representations. Examples are sentences from the language being learned and the background knowledge represents an existing partial grammar, perhaps supplemented with constraints on the possible forms of rules. In grammatical disambiguation and contextual interpretation the theory to be synthesised consists of a set of axioms or logical statements prescribing properties of the domain relevant to these tasks. These properties may include an ontology or type hierarchy and (perhaps default) statements about typical properties of and relations holding between the entities in the domain. The examples consist of both correct and incorrect analyses or contextual interpretations for sentences, or sentence-context pairs, where contexts can be represented as disambiguated sentences. The background knowledge may consist of a partial domain theory, or an encoding of whatever existing constraints on disambiguation or interpretation are known. The resulting theory is used as a filter on hypothesised alternative interpretations.

4.3 GEOGRAPHIC DATABASE QUERIES

ILP has been used for learning grammar and semantics using the CHILL system (Zelle and Mooney, 1996b). In this case, background knowledge and examples were taken from an existing database of US geographical facts. Each example consisted of a sentence paired with its semantics as shown in Figure 14.5 (figure taken from (Mooney, 1997)).

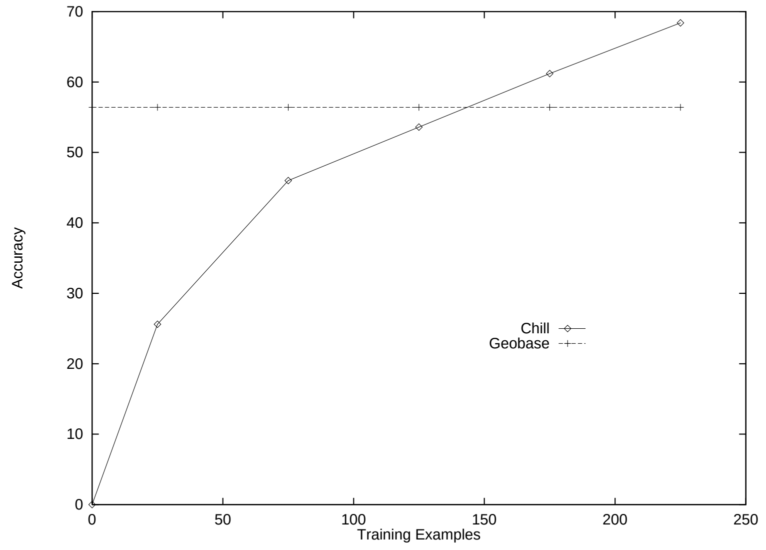


Figure 14.6 CHILL's accuracy on learning grammar and semantics

The data was gathered by asking subjects to generate appropriate questions. Each question was then paired with appropriate logical queries to give 250 examples. Figure 14.6 (taken from (Mooney, 1997)) shows CHILL's accuracy on progressively larger training sets averaged over 10 trials. The line labelled "Geobase" shows the accuracy of an existing commercially-developed hand-coded system for the same domain. CHILL outperforms the existing system when trained on 175 or more examples.

4.4 MORPHOLOGY

Mooney and Califf (Mooney and Califf, 1995) have applied ILP to learning the past tense of English verbs. Learning of English past-tense has become a benchmark problem in the computational modeling of human language acquisition (Rumelhart and McClelland, 1986; Ling, 1994). In (Mooney and Califf, 1995) it was shown that a particular ILP system, FOIDL, could learn this transformation more effectively than previous neural-network and decision-tree methods. FOIDL's first-order default rule style representation was demonstrated by the authors as producing a predictive accuracy advantage in this domain.

However, more recently Muggleton and Bain (Muggleton and Bain, 1999) have shown that ILP prediction techniques based on Analogical Prediction (AP) produce even higher accuracies on the same data. AP is a half-way house between instance-based learning and induction. Thus AP logical hypotheses are generated on the fly for each instance to be predicted. The form of examples and hypotheses is shown in Figure 14.7. A comparison of learning curves for various systems is shown in Figure 14.8. The horizontal line labelled "Default rules" represents the following simple Prolog program which adds a 'd' to verbs ending in 'e' and otherwise adds 'ed'.

```
past(A,B) :- split(A,B,[e]), split(B,A,[d]), !.
```

Examples	**Hypotheses**
past([w,o,r,r,y],[w,o,r,r,i,e,d]). past([w,h,i,z],[w,h,i,z,z,e,d]). past([g,r,i,n,d],[g,r,o,u,n,d]).	past(A,B) :- split(A,C,[r,r,y]), split(B,C,[r,r,i,e,d]).

Figure 14.7 Form of examples and hypotheses for past tense domain

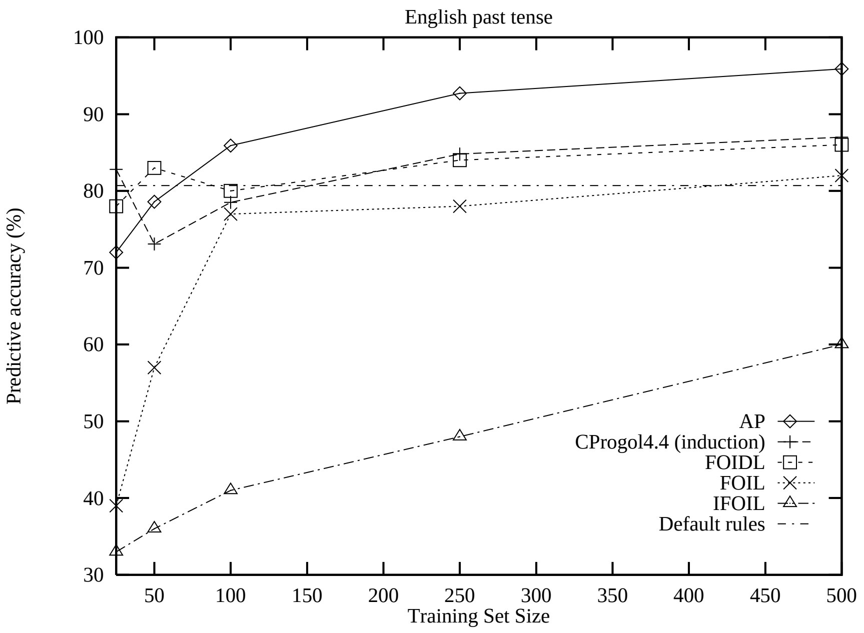


Figure 14.8 Learning curves for alphabetic English past tense. Comparisons were made between AP, CProgol4.4, FOIDL, FOIL, IFOIL and a hand-coded set of default rules. Results were averaged over 10 random chosen training sets of sizes 25,50,100,250,500 with accuracies measured over test sets of size 500.

```
past(A,B) :- split(B,A,[e,d]).
```

The differences between AP and all other systems are significant at the 0.0001 level with 250 and 500 examples.

4.5 WHY IS NLP GOOD FOR ILP?

From the ILP point of view NLP has recently been recognised as a challenging application area. Some successes have been achieved in using ILP to learn grammar and semantics ((Zelle and Mooney, 1996b; Zelle and Mooney, 1993; Zelle and Mooney, 1996a; Cussens et al., 1997)). The existence within NLP problems of hierarchically defined, structured data with large amounts of relevant logically defined background knowledge provides a perfect testbed for stretching ILP technology in a way that would be beneficial in other application areas (Bratko and Muggleton, 1995; Sternberg et al., 1994; King et al., 1996; Srinivasan et al., 1996; Finn et al., 1998). The York system Progol (Muggleton, 1995) is arguably the most general purpose and widely applied ILP system. Most ILP systems concentrate on the issue of learning a single (usually non-recursive) concept and assume a set of completely and correctly defined background predicates. By contrast NLP applications need techniques to deal with simultaneous completion and correction of a set of related (often recursively defined) predicates. It is also expected to be necessary to implement new techniques for automatic augmentation of the set of background predicates, (Khan et al., 1998), in order to handle incompleteness of available vocabulary.

5 CONCLUSION

In his presentation of ILP biological discovery results to the Royal Society (Sternberg et al., 1994) Sternberg emphasised the aspect of joint human-computer collaboration in scientific discoveries. Science is an activity of human societies. It is the author's belief that computer-based scientific discovery must support strong integration into the existing social environment of human scientific communities. The discovered knowledge must add to and build on existing science. The ability to incorporate background knowledge and re-use learned knowledge together with the comprehensibility of the hypotheses, have marked out ILP as a particularly effective approach for scientific knowledge discovery.

In the natural language area ILP has not only been shown to have higher accuracies than various other ML approaches in learning the past tense of English (see Section 4.4) but also shown to be capable of learning accurate grammars which translate sentences into deductive database queries (Zelle and Mooney, 1996b) (see Section 4.3). In both cases, follow up studies (Thompson et al., 1997; Džeroski and Erjavec, 1997) have shown that these ILP approaches to natural language problems extend with relative ease to various languages other than English.

The area of Learning Language in Logic (LLL) is producing a number of challenges to existing ILP theory and implementations. In particular, language applications of ILP require revision and extension of a hierarchically defined set of predicates in which the examples are typically only provided for predicates at the top of the hierarchy. New predicates often need to be invented, and complex recursion is usually involved. Similarly the term structure of semantic objects is far more complex than in other applications of ILP. Advances in ILP theory and implementation related to the challenges of LLL are already producing beneficial advances in other sequence-oriented applications of ILP. In addition LLL is starting to develop its own character as a sub-discipline of AI involving the confluence of computational linguistics, machine learning and logic programming.

Acknowledgements

Thanks are due to the funders and researchers who helped develop the technology and applications of ILP on the Esprit projects ECOLES (1989-1992), ILP I (1992-1995), ILPnet (1992-1995), ILP II (1996-1999), ILPnet2 (1998-2001), ALADIN (1998-2001) and the EPSRC Rule-based system project

(1990-1993), Experimental Application and Developments of ILP (1993-1996), Distribution-based Machine Learning (1995-1998), Closed Loop Machine Learning (1992-2002). The author would also like to acknowledge the personal support he received in carrying out research into ILP under an SERC Post-doctoral Fellowship (1990-1992), an EPSRC Advanced Research Fellowship (1993-1998) and Research Fellowship from Wolfson College (1993-1997). Warm thanks are offered to the author's wife Thirza and daughter Clare for their continuous cheerful support.

References

Bratko, I. and Muggleton, S. (1995). Applications of inductive logic programming. *Communications of the ACM*, 38(11):65–70.

Brenner, S., Chothia, C., Hubbard, T., and Murzin, A. (1996). Understanding protein structure: using scop for fold interpretation. *Methods in Enzymology*, 266:635–643.

Briscoe, T. and Carroll, J. (1993). Generalized probabilistic lr parsing of natural language (corpora) with unification-based grammars. *Computational Linguistics*, 19(1):25–59.

Collins, M. (1996). A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 184–191, Santa Cruz, California, USA.

Cussens, J., Page, D., Muggleton, S., and Srinivasan, A. (1997). Using Inductive Logic Programming for Natural Logic Processing. In Daelemans, W., Weijters, T., and van der Bosch, A., editors, *ECML'97 – Workshop Notes on Empirical Learning of Natural Language Tasks*, pages 25–34, Prague. University of Economics. Invited keynote paper.

Džeroski, S. and Erjavec, T. (1997). Induction of Slovene nominal paradigms. In Lavrač, N. and Džeroski, S., editors, *Proceedings of the 7th International Workshop on Inductive Logic Programming*, pages 141–148. LNAI 1297, Springer Verlag.

Finn, P., Muggleton, S., Page, D., and Srinivasan, A. (1998). Pharmacophore discovery using the inductive logic programming system Progol. *Machine Learning*, 30:241–271.

Hobbs, J. R., Stickel, M. E., Appelt, D. E., and Martin, P. (1993). Interpretation as abduction. *Artificial Intelligence*, 63:69–142.

Khan, K., Muggleton, S., and Parson, R. (1998). Repeat learning using predicate invention. In Page, C., editor, *Proc. of the 8th International Workshop on Inductive Logic Programming (ILP-98)*, LNAI 1446, pages 165–174, Berlin. Springer-Verlag.

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

King, R., Muggleton, S., Srinivasan, A., and Sternberg, M. (1996). 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.

Kowalski, R. (1980). *Logic for Problem Solving.* North Holland.

Krotov, A., Gaizauskas, R., Hepple, M., and Wilks, Y. (1997). Compacting the Penn Treebank Grammar. *Proceedings of the COLING-ACL'98 Joint Conference*, pages 699–703, Association for Computational Linguistics. Also: Research Memorandum CS-97-04, Department of Computer Science, University of Sheffield.

Krotov, A., Gaizauskas, R., and Wilks, Y. (1994). Acquiring a stochastic context-free grammar from the Penn Treebank. In *Proc. of the Third Conference on the Cognitive Science of Natural Language Processing.*

Ling, C. (1994). Learning the past tense of english verbs: the symbolic pattern associators vs. connectionist models. *Journal of Artificial Intelligence Research*, 1:209–229.

Lloyd, J. (1987). *Foundations of Logic Programming.* Springer-Verlag, Berlin. Second edition.

Magerman, D. (1995). Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 276–283, Cambridge, MA.

McCarthy, J. (1959). Programs with commonsense. In *Mechanisation of thought processes*, volume 1. Her Majesty's Stationery Office, pages 75–91, London. Reprinted (with an added section on 'Situations, Actions and Causal Laws') in *Semantic Information Processing*, ed. M. Minsky (Cambridge, MA: MIT Press (1963)).

Mooney, R. (1997). Inductive logic programming for natural language processing. In Muggleton, S., editor, *Proceedings of the Sixth International Workshop on Inductive Logic Programming*, pages 3–21. Springer-Verlag, Berlin. LNAI 1314.

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

Muggleton, S. (1991). Inductive logic programming. *New Generation Computing*, 8(4):295–318.

Muggleton, S. (1994a). Bayesian inductive logic programming. In Cohen, W. and Hirsh, H., editors, *Proceedings of the Eleventh International Machine Learning Conference*, pages 371–379, San Mateo, CA. Morgan-Kaufmann. Keynote presentation.

Muggleton, S. (1994b). Bayesian inductive logic programming. In Warmuth, M., editor, *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory*, pages 3–11, New York. ACM Press. Keynote presentation.

Muggleton, S. (1995). Inverse entailment and Progol. *New Generation Computing*, 13:245–286.

Muggleton, S. (1998). Completing inverse entailment. In Page, C., editor, *Proceedings of the Eighth International Workshop on Inductive Logic Programming (ILP-98)*, LNAI 1446, pages 245–249. Springer-Verlag, Berlin.

Muggleton, S. (2000). Learning from positive data. *Machine Learning.* Accepted subject to revision.

Muggleton, S. and Bain, M. (1999). Analogical prediction. In *Proc. of the 9th International Workshop on Inductive Logic Programming (ILP-99)*, pages 234–246, Berlin. Springer-Verlag.

Muggleton, S. and Feng, C. (1992). Efficient induction of logic programs. In Muggleton, S., editor, *Inductive Logic Programming*, pages 281–298. Academic Press, London.

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

Muggleton, S. and De Raedt, L. (1994). Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19,20:629–679. $ftp://ftp.cs.york.ac.uk/pub/ML_GROUP/Papers/lpj.ps.gz$

Nienhuys-Cheng, S.-H. and de Wolf, R. (1997). *Foundations of Inductive Logic Programming.* Springer-Verlag, Berlin. LNAI 1228.

Plotkin, G. (1971). *Automatic Methods of Inductive Inference.* PhD thesis, Edinburgh University.

Rumelhart, D. and McClelland, J. (1986). On learning the past tense of english verbs. In *Explorations in the Micro-Structure of Cognition Vol. II*, pages 216–271. MIT Press, Cambridge, MA.

Shaprio, E. (1983). *Algorithmic Program Debugging.* PhD thesis, Yale University, MIT Press.

Srinivasan, A., , Muggleton, R. K. S., and Sternberg, M. (1997). Carcinogenesis predictions using ILP. In Lavrač, N. and Džeroski, S., editors, *Proceedings of the Seventh International Workshop on Inductive Logic Programming*, pages 273–287. Springer-Verlag, Berlin. LNAI 1297.

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

Sternberg, M., King, R., Lewis, R., and Muggleton, S. (1994). Application of machine learning to structural molecular biology. *Philosophical Transactions of the Royal Society B*, 344:365–371.

Thompson, C., Mooney, R., and Tang, L. (1997). Learning to parse natural language database queries into logical form. In *Workshop on Automata Induction, Grammatical Inference and Language Acquisition.* Paper accessible from www-univ-st-etienne.fr/eurise/pdupont.html

Turcotte, M., Muggleton, S., and Sternberg, M. (1998). Protein fold recognition. In Page, C., editor, *Proc. of the 8th International Workshop on Inductive Logic Programming (ILP-98)*, LNAI 1446, pages 53–64, Berlin. Springer-Verlag.

Turing, A. (1950). Computing machinery and intelligence. *Mind*, 59(236):435–460.

Yamamoto, A. (1997). Which hypotheses can be found with inverse entailment? In Lavrač, N. and Džeroski, S., editors, *Proceedings of the Seventh International Workshop on Inductive Logic Programming*, pages 296–308. Springer-Verlag, Berlin. LNAI 1297.

Zelle, J. and Mooney, R. (1993). Learning semantic grammars with constructive inductive logic programming. In *Proceedings of the Eleventh National*

Conference on Artificial Intelligence, pages 817–822, San Mateo, CA. Morgan Kaufmann.

Zelle, J. and Mooney, R. (1996a). Comparative results on using inductive logic programming for corpus-based parser construction. In *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*, pages 355–369. Springer, Berlin.

Zelle, J. and Mooney, R. (1996b). Learning to parse database queries using Inductive Logic Programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1050–1055, Portland, Oregon. AAAI Press/MIT Press.