Inductive Logic Programming: entry for the MIT encyclopaedia of Cognitive Sciences

Stephen Muggleton
Computer Science Department,
University of York.

December 9, 1997

Inductive Logic Programming (ILP) is the topic area of Computer Science involved with the automatic synthesis and revision of logic programs (see LOGIC PROGRAMMING) from partial specifications. The word "Inductive" is used in the sense of Philosophical rather than Mathematical Induction. In his *Posterior analytics* Aristotle introduced Philosophical Induction (in Greek *epagogue*) as the study of the derivation of general statements from specific instances (see INDUCTION). This can be contrasted with deduction, which involves the derivation of specific statements from more general ones. For instance, induction might involve the conjecture that a) "all swans are white" from the observation b) "all the swans in that pond are white", where b) can be derived deductively from a). In the *Principles of Science* the nineteenth century philosopher and economist Jevons gave some simple demonstrations that inductive inference could be carried out by reversal of deductive rules of inference. The idea of reversing deduction has turned out to be one of the strong lines of investigation within ILP.

In ILP the general and specific statements are in both cases logic programs. A logic program is a set of Horn clauses (see LOGIC PROGRAMMING). Each Horn clause has the form *Head*←*Body*. Thus the definite clause active(X)←*charged(X), polar(X)* states that any X which is charged and polar is also active. In this case, active, charged and polar are called "predicates", and are properties which are either true or false of X. Within deductive logic programming the inference rule of *resolution* is used to derive

consequences from logic programs. According to resolution, given atom a and clauses C, D, from the clauses $a \wedge C$ and $D \wedge \overline{a}$ the clause $C \wedge D$ can be concluded. A connected set of resolutions is called a proof or deductive derivation.

The inductive derivations in ILP are sometimes thought of as inversions of resolution-based proofs (see "inverse resolution" in Muggleton and De Raedt 1994). Following Plotkin's work in the 1970's the logical specification of an ILP problem is thought of as consisting of three primary components: B, the background knowledge (eg. things cannot be completely black and completely white), E the examples (eg. the first swan on the pond is white) and H, the hypothesis (eg. all swans are white). The primary relation between these three components is that the background together with the hypothesis should allow the derivation of the examples. This can be written in logical notation as follows.

$$B, H \vdash E$$

Given only such logical specifications, it has long been known that induction is not sound. That is to say, H is not a necessary conclusion from knowing B and E.

A sound treatment is possible by viewing inductive inference as the derivation of statements with associated probabilities. This approach was advocated by the philosopher Carnap in the 1950's and has been taken up more recently in a revised form within ILP. The framework typically chosen is that of Bayesian inference (see BAYESIAN LEARNING). This is a probabilistic framework which allows calculations of the probability of certain events happening given that other events have happened. Thus suppose we imagine that nature, or a teacher, randomly and independently chooses a series of concepts to be learned, where each concept has an associated probability of coming up. Suppose also that for each concept a series of instances are randomly and independently chosen and associated with the label true or false depending on whether they are or are not examples of the chosen concept. From an inductive agent's point of view, prior to receipt of any examples the probability that any particular hypothesis H will fit the data is $p(H)$, the probability of H being chosen by the teacher. Likewise, $p(E)$ denotes the probability that the teacher will provide the example sequence E, $p(H|E)$ is the probability the hypothesis chosen was H given that the example sequence

was E and $p(E|H)$ is the probability that the example sequence was E given that the hypothesis chosen was H. According to Bayes's theorem

$$p(H|E) = \frac{p(H)p(E|H)}{p(E)}.$$

The most likely hypothesis H given the examples E is the one which maximises $p(H|E)$. In fact it is sufficient to maximise $p(H)p(E|H)$ since $p(E)$ is common to all candidate hypotheses. As with all Bayesian approaches the main issue is how to choose the inductive agent's prior probabilities over hypotheses. In common with other forms of machine learning, this is usually done within ILP systems using a MINIMUM DESCRIPTION LENGTH (MDL) approach, i.e. probability distribution which provides assigns higher probabilities to textually simple hypotheses.

Within any computational framework such as ILP, a key question involves the efficiency with which the inductive agent converges on the "correct" solution. Here a number of ILP researchers have taken the approach of COMPUTATIONAL LEARNING THEORY, which studies the numbers of examples required for an inductive algorithm to return with high probability an hypothesis which has a given degree of accuracy (this is Valiant's Probably Approximately Correct (PAC) model). One interesting result of these investigations has been that while it has been shown that logic programs cannot be PAC-learned as a class, time-bound logic programs (those in which the derivation of instances are of bounded length) are efficiently learnable within certain Bayesian settings in which the probability of hypotheses decays rapidly (eg. exponentially or with the inverse square) relative to their size.

One of the hard, and still largely unsolved problems within ILP is that of *predicate invention*. This is the process by which predicates are added to the background knowledge in order to provide compact representations for foreground concepts. Thus suppose you were trying to induce a phrase structured grammar for English, and you already have background descriptions for noun, verb and verb phrase, but no definition for a noun phrase. "Inventing" a definition for noun phrase would considerably simplify the overall hypothesised descriptions. However, the space of possible new predicates that could be invented is clearly large, and its topology not clearly understood.

ILP has found powerful applications in areas of scientific discovery in which the expressive power of logic programs is necessary for representing

the concepts involved. Most notably, ILP techniques have been used to discover constraints on the molecular structured of certain biological molecules. These semi-automated scientific "discoveries" include a new structural alert for mutagenesis and the suggestion of a new binding site for an HIV protease inhibitor. ILP techniques have also been demonstrated capable of building large grammars automatically from example sentences.

Recently the philosopher of science Gillies has made a careful comparison of techniques used in ILP with Bacon and Popper's conception of scientific induction. Gillies concludes that ILP techniques combine elements from Bacon's "pure" knowledge-free notion of induction and Popper's falsificationist approach. As indicated in this article, ILP has helped clarify a number of issues in the theory, implementation and application of inductive inference within a compuational logic framework.

References

Bratko I. and Muggleton S. (1995). Applications of Inductive Logic Programming. In Communications of the ACM, 38(11): 65-70.

Carnap R. (1952). The Continuum of Inductive Methods. Chicago University, Chicago.

Gillies, D.A. (1996). Artificial intelligence and scientific method. Oxford University Press, 1996.

Jevons, W.S. (1874). The Principles of Science: a Treatise on Logic and Scientific Method. MacMillan [republished in 1986 by IBIS publishing].

King R., Muggleton S., Srinivasan A., 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.

Lavrac N. and Dzeroski S. (1994). Inductive Logic Programming. Ellis Horwood.

Muggleton S. and de Raedt L. (1994). Inductive Logic Programming: Theory and Methods. In Journal of Logic Programming, 19,20: 629–679.

Muggleton S. (1995). Inverse entailment and Progol. In New Generation Computing Journal, 13, 245-286.

Plotkin, G. (1971). A further note on inductive generalisation. In Machine Intelligence 6, Edinburgh University Press.

Sternberg M., King R., Lewis R. and Muggleton S. (1994). Application of Machine Learning to Structural Molecular Biology. In Philosophical Transactions of the Royal Society B, 344: 365-371.

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