Logic Programs as a Basis for Machine Learning

Claude Sammut

The Turing Institute[1]
36 North Hanover Street
Glasgow G1 2AD
Scotland

ABSTRACT

First order predicate logic appears frequently in Artificial Intelligence. In learning programs, it is often the language used to describe concepts, rules, examples, events, *etc*. This paper presents an overview of research in logic-related learning systems and describes those features of first order logic which have made it such a useful tool. Two developments are of particular interest to us: the use of logic in what is now called "constructive induction", and the benefits to machine learning contributed by logic programming.

1. Introduction

Machine learning conferences are often the scenes of battles between rival factions convinced that *their* approach to learning is the better one. This paper may either be seen as an attempt to bring peace between several tribes or as firing a shot in another battle. For example, we try to show how the use of first order logic can unite two apparently different approaches, namely data-driven and model-driven learning. However in advocating logic we may incur the wrath of those who claim that logic is not a suitable language for knowledge representation. It is hoped that the advantages we discuss below will out-weigh the disadvantages. Our argument is presented as an overview of some learning methods which are dependent on a logic formalism.

Learning algorithms are often placed in either of two categories: *data-driven* or *model-driven*. Data-driven learning makes generalisations by finding common sub-expressions in the descriptions of a set of examples, whereas model-driven learning uses a model of the domain in which learning is taking place to propose generalisations. Data-driven systems appear to use a "weak method" because they do not take advantage of domain knowledge and they usually require a large set of examples to perform the learning task accurately. Model-driven systems appear to use a "strong method" because their approach is knowledge-based and may require

[1] On leave from the Department of Computer Science, University of New South Wales, Sydney, Australia.

only a small number of examples to learn a concept description or operation. However, they usually require a considerable amount of priming with background knowledge.

Trends frequently come and go in Artificial Intelligence. For some time there was heavy concentration of effort in data-driven learning. More recently the model-driven approach has come to the fore. Explanation-based learning is one instance of model-driven systems. It is our contention that by adopting a well-behaved description language, such as first-order logic (and horn-clause logic, in particular), not only can both approaches be accommodated, but a number of disadvantages can be eliminated from them.

The overview that we present will concentrate on those learning systems which have been influenced by research in logic programming and theorem proving. The connection between these endeavours and machine learning is that the clausal form of logic brings simplifications not only to problem solving but also to machine learning. This simplification has made a significant contribution to "constructive induction" which is at the heart of learning systems which enhance their own descriptive power by adding new terms to the concept description language. That is, the language grows with the learner's experience.

2. Growing Description Language

Banerji (1964, 1969, 1980) presents one of the earliest uses of first-order logic in learning. He was especially concerned that the description language of a learning system should be able to *grow*. That is, new terms can be added to the language as the system learns. There should not be a pre-defined and fixed "vocabulary", because when the limits of the language are reached, so too are the limits of the system as a whole. For example, LEX (Mitchell, Utgoff, Nudel and Banerji, 1981) used a fixed language to describe algebraic expressions. When it was found that LEX was unable to learn some concepts because the language could not describe the necessary expressions, a method for extending the language had to be found (Utgoff, 1986).

Following Banerji, Cohen (1978) wrote a program, called CONFUCIUS, which successfully demonstrated that a program using first-order logic as its description language could learn in an variety of domains. More importantly, it had the property that concepts which had been learned could be stored in the system's knowledge base and used in the description of other concepts. Hence it satisfied the objective of creating a learning program whose ability to describe concepts grows over time.

Sammut's program, Marvin (Sammut, 1981), extended CONFUCIUS in several ways. The representation language used in Marvin became Horn-clause logic and this had two major benefits. Firstly, horn clauses can be viewed as productions which specify how to generalise a description. This leads to a very simple but powerful learning mechanism. The second benefit was that horn clauses are easy to execute as programs (as is done in Prolog) and so a concept's description can generate instances of that concept. This feature was used to show counter-examples to the trainer during learning.

Every learning system must have some way of representing concepts and examples of concepts. For convenience, the term *concept* is used to stand for all those things which a program is intended to learn including plans, heuristics, scripts, *etc.* We refer to the method of representation as the concept description language. It must have a primitive set of attributes or terms to begin with. The key to being able to extend a language through learning is that a reference to a newly learned concept should be allowed to appear in other concept descriptions in any place where a primitive could be used. Thus a growing language allows the learning system to construct its own high-level attributes with which it can simplify concept descriptions. Indeed, a simplification mechanism can be at the heart of a learn ing program, as in Muggleton's DUCE (Muggleton, 1987).

To be able to use a concept description in place of a primitive, it must be possible to interpret the description as a recognition procedure. For example, if we have just learned a description for the concept *chair,* then we should be able to refer to *chair* in, say, the description of an office scene and expect that the chair "program" will recognise the chairs in the scene. Of course, many knowledge representation systems allow the declarative description of a concept to be interpreted as a recognition procedure. However, it is not always easy to use new concepts in place of primitive attributes. For example, network representations require extensions providing for sub-nets which can be hidden behind a single node of a higher level network. Frame systems require the ability to describe a slot-type by other frames. But while it is possible to achieve the same effect with a variety of representation methods, extensibility is a natural and easily implemented feature of horn-clause logic. Moreover, a description in horn-clause logic is a logic program, so we have all the advantages of a clean, declarative description of concepts and also a language powerful enough to be a general purpose programming language and thus applicable in a large variety of domains.

To recognise an object, it is only necessary to use a horn clause in a forward chaining manner. Suppose we have a set of clauses:

$$C_1 \leftarrow P_{11} \wedge P_{12} \tag{1}$$

$$C_2 \leftarrow P_{21} \wedge P_{22} \wedge C_1 \tag{2}$$

and a instance:

$$P_{11} \wedge P_{12} \wedge P_{21} \wedge P_{22} \tag{3}$$

The P_{ij} denote primitive terms. Applying the clauses (1) and (2) as if they were simple productions, clause (1) recognises the first two terms in expression (3) reducing it to $P_{21} \wedge P_{22} \wedge C_1$. This can be reduced by clause (2) to C_2. Thus, clauses (1) and (2) recognise expression (3) as the description of an instance of concept C_2. Of course, instead of the primitive terms P_{ij} we could have references to other concepts, just as C_1 is referred to in clause (2). This allows us to build arbitrarily complex recognition procedures. Moreover, adding new terms to the description language becomes a trivial case of adding a new clause to the existing set.

Another important feature of a horn clause description language is that when the clauses are executed in a backward chaining manner, they produce instances of concepts. When we apply backward chaining to horn clauses, effectively we ask the system to prove a proposition. The proof of a concept is the existence of an instance of it. Of course, the proof procedure is non-deterministic in the sense that any instance will do as a proof. This property of a logic-based representation is important when part of the learning program's task is to actively explore its task environment. We will return to learning in reactive environments following a discussion of induction using horn clause logic.

3. Inverting Resolution

Robinson's (1965) resolution principle has had far-reaching effects for theorem-proving and logic programming. Resolution provides an efficient means of deriving a solution to a problem, giving a set of axioms which define the task environment. Whereas resolution takes two terms and resolves them into a most general unifier, *anti-unification* finds the least general generalisation of two terms. Plotkin (1970, 1971) first investigated the derivation of least general generalisations, a problem which was later taken up by Vere (1975, 1977) and Buntine (1986). An inverse of resolution provides an extremely powerful induction method (Muggleton, 1987; Muggleton and Buntine, 1987).

The method of least general generalisations is based on the idea of *subsumption*. This states that a clause C_1 subsumes, or is more general than, another clause C_2 if there is a substitution σ such that $C_2 \supseteq C_1\sigma$. A least general generalisation is a generalisation which is less general than any other such generalisation. For example, the least general generalisation of

$$p(g(a), a) \tag{4}$$

and

$$p(g(b), b) \tag{5}$$

is

$$p(g(X), X). \tag{6}$$

Note that under the substitution {a/X} (6) is equivalent to (4) and under the substitution { b/X} (6) is equivalent to (5). See Plotkin (1970).

Subsumption served a very useful tool in several learning systems. Vere (1975) developed a method for efficiently searching for possible substitutions among literals. However, as Buntine (1986) points out, subsumption alone has its limitations. In particular, it is unable to take advantage of background information which may assist generalisation. We use Buntine's example to illustrate the problem. Suppose we are given two instances of a concept *cuddly_pet,*

$$\text{cuddly\_pet}(X) \leftarrow \text{fluffy}(X) \wedge \text{dog}(X) \tag{7}$$

$$\text{cuddly\_pet}(X) \leftarrow \text{fluffy}(X) \wedge \text{cat}(X) \tag{8}$$

Suppose we also know the following:

$$\text{pet}(X) \leftarrow \text{dog}(X) \tag{9}$$

$$\text{pet}(X) \leftarrow \text{cat}(X) \tag{10}$$

According to subsumption, the least general generalisation of (7) and (8) is:

$$\text{cuddly\_pet}(X) \leftarrow \text{fluffy}(X) \tag{11}$$

In light of the background knowledge, this is an over-generalisation and potentially dangerous since any fluffy object could be considered a cuddly pet. A more appropriate one would be:

$$\text{cuddly\_pet}(X) \leftarrow \text{fluffy}(X) \wedge \text{pet}(X) \qquad (12)$$

In fact, Sammut (1981) took the conservative view that generalisation should only be done when the relevant background knowledge is available. Observing an example such as (7), Sammut's program, Marvin, uses clause (9) as a rewrite rule to produce a generalisation which is clause (12). That is, when the right hand side of a background clause matches a subset of literals in the right hand side of an instance clause, then those literals are replaced by the left hand side of the background clause. Initially, no generalisation could occur because Marvin had no background knowledge and observations were simply stored in the knowledge base in horn-clause form. However, as the knowledge base expanded, so did Marvin's ability to generalise. Sammut and Banerji (1986) attempted to formalise this process, but it was left to Buntine (1986) and Muggleton (1987) to produce more elegant theories for generalisation.

Muggleton's DUCE (1987) relies on a set of operators to *compact* the description of a set of examples to a simpler description. Each example is described by a propositional horn-clause. Some operators preserve the equivalence of descriptions but reduce the number of symbols required while others produce generalisations. There are six operators in all and they are the basis for a method of anti-unification. Indeed, one of the goals of this work was to produce a complete inverse of resolution. All six operators are necessary for the completeness of the theory, but pairs of operators are sufficient for induction. We will describe one such pair and refer the interested reader to Muggleton's paper (1987) for the complete description. A first-order version of DUCE, called Cigol[2], is under construction (Muggleton and Buntine, 1987).

Absorption. Given a set of clauses, the body of one of which is completely contained in the bodies of the others, such as:

$$X \leftarrow A \wedge B \wedge C \wedge D \wedge E$$

$$Y \leftarrow A \wedge B \wedge C$$

we can hypothesise:

$$X \leftarrow Y \wedge D \wedge E$$

$$Y \leftarrow A \wedge B \wedge C$$

In fact, this is the generalisation rule used in Marvin.

Intra-construction. This is the distributive law of Boolean equations. Intra-construction takes a group of rules all having the same head, such as:

[2] Cigol is logiC backwards.

$$X \leftarrow B \wedge C \wedge D \wedge E$$

$$X \leftarrow A \wedge B \wedge D \wedge F$$

and replaces them with:

$$X \leftarrow B \wedge D \wedge Z$$

$$Z \leftarrow C \wedge E$$

$$Z \leftarrow A \wedge F$$

Note that intra-construction automatically creates a new term in its attempt to simplify descriptions. This satisfies Banerji's desire for a growing description language, since, as we stated earlier, the new term is able to be used in the descriptions of other concepts. At any time during induction there may be a number of applicable operators. The one chosen is the operator which will result in the greatest compaction. This is DUCE's heuristic for guiding its search for a generalisation.

Buntine (1986) produced a theory for induction of horn-clauses called *generalised subsumption*. In this framework, all subsumption is done relative to a logic program. In this case, the logic program is the set of clauses which represent the learner's background knowledge. Buntine also examined the very important problem of defining and limiting the search space for induction. The following section draws on a number of his observations.

4. Searching for Generalisations

An appealing approach to the search problem is Mitchell's Version Space (Mitchell, 1977). Here, a bi-directional search of the space of sentences in the description language is used to locate the sentence representing the best generalisation. The given instances of a concept form a bound on the most specific sentence which could be a generalisation. A bound on the most general sentence is provided by the legal expressions in the language which could possibly be generalisations of the instances. The worst case is a sentence which covers the entire universe. New positive examples force the bounds on the most specific sentences to become more general. New negative instances force the bounds on the most general sentences to become more specific. When these boundaries meet and provided that all positive and negative examples can be correctly classified, we have the sought for generalisation. Unfortunately, it is not easy to apply the Version Space method to horn-clause logic because of the richness of the language. Since so many things are describable, the branching factor even in a bi-directional search becomes prohibitive.

Marvin adopted a strictly specific-to-general search in which only those sentences which could be derived from the current knowledge base were considered. This had the effect of reducing the search space dramatically, but at the expense of a rather large assumption that the system will have learned all the background knowledge necessary for learning further concepts. Since Marvin was being trained by a "friendly" tutor, this assumption was justified, but it is obviously too restrictive for more general learning tasks. Another problem was that as the knowledge base grew, so did the search space.

Shapiro (1981) adopted a general-to-specific search strategy in his Model Inference System (MIS). First, a language L, which is a subset of a logic programming language (effectively pure Prolog) is defined. It is assumed that the model (or set of concepts, in our terminology) to be inferred is fully describable in ***L.*** A *refinement operator* must also be specified. This operator shows how a more specific sentence in ***L*** can be derived from another given sentence. The search procedure uses refinement as its means of moving down branches in the search tree. The most significant drawback of MIS was that it required a fixed description language. If ***L*** were too general, then the search space would become very large. It is also not possible to add new terms to the language. It is worth noting that MIS was a side-effect of Shapiro's main research interest at the time, namely debugging logic programs. Automatically debugging the null program is the same as automatically synthesising a program. Since logic programs can be used to describe concepts, learning, automatic programming and debugging are all united!

Muggleton (1987) used a measure of compaction to guide DUCE's search. Since there may be more than one compaction operator applicable at any time, a measure is applied to each rule to determine which one will result in the greatest compaction. The measure of compaction is the reduction in the number of symbols in the set of clauses after applying an operator. Each operator has an associated formula for computing this reduction. Naively carrying out the computation would require applying all the operators to all clauses and choosing the best. Obviously this is too expensive for a large data set so a significant subset of rules is chosen instead. DUCE also uses a bit-string representation of clauses to speed up pattern matching. It should also be noted that an *oracle,* that is human supervisor, can reject any hypotheses that DUCE produces by the application of a rule. This will force it to search for other hypotheses using a best-first search. The oracle also has the opportunity to interpret a new hypothesis and give it a sensible name.

5. Learning and Logic

Now let use examine how the evolution of the ideas presented above affects our view of machine learning in general. The most important result we wish to emphasise is that an elegant theory often leads to elegant solutions to problems which are both practical and efficient. The theory may also help to tie together a number of seemingly separate issues.

At the beginning of this paper we mentioned one such separation between data-driven and model-driven learning algorithms. It should be evident that viewing data and knowledge bases as logic programs gives us a mechanism capable of performing both data-driven and model-driven learning. For example, DUCE's induction algorithm looks for regularities among data using a search that is dictated by the form of the clauses in the data. Since the data are represented by the same horn clause form as concepts, the search can be assisted by background knowledge of the concepts already learned. The uniform representation means that no special mechanism has to be built to combine data-driven and model-driven approaches.

Another advantage of the logic programming representation is that concepts can be executed as programs to yield instances of themselves. This is very useful for learning programs which must interact with their environment in complex ways. For example, Marvin tests a generalisation by presenting instances of it to the trainer. The idea here is that it should not be necessary for the trainer to know the exact representation of Marvin's concept for him or her to judge how well Marvin is learning. After all, a teacher can usually only judge the knowledge level of a student by watching the performance of some task.

The concepts which can be represented by a logic programming system need not be restricted to static descriptions of objects. The language is flexible enough to be able to teach a logic-based system concepts of planning and problem solving. Note that an instance of a plan is the execution of it. So instead of showing an instance of a generalisation to a trainer, a learning system which has control of an agent in a robot world can execute a plan. As well as actions, the concept description contains the expected result of the plan. If the actual result meets this expectation then the experiment supports the generalisation. If the experiment causes a different result then the generalisation is refuted. Recently, Hume has moved a descendent of Marvin, called CAP (Sammut and Hume, 1987), into a simulated robot world (Hume, 1985) where many concepts are, in fact, plans of action. Attempts are also underway to install a logic-based learning system onboard a real robot. A similar approach has been taken by Carbonell and Gil (1987). Their representation language is based on first-order logic, although not horn clauses.

One aspect of learning in a robot domain is common to other problem solving domains, that is the importance of learning efficient ways of achieving a result even after an abstract concept of the solution has been acquired. This requirement to "operationalise" a concept has resulted in a surge of interest in *explanation-based generalisation*. Here also, we have another example of an elegant theory bringing order amidst confusion. Kedar-Cabelli and McCarty (1987) have shown how explanation-based generalisation can be viewed as resolution theorem proving. Mitchell, Keller and Kedar-Cabelli (1986) state the explanation-based generalisation problem as follows:

Given

> a concept (whose description is to be elaborated by EBG)
> a training example (of the concept)
> a domain theory (rules and facts about the domain)
> an operationality criterion (for the form of the plan to be learned)

determine

> a generalisation of the training example that is a sufficient concept definition for the target concept and that satisfies the operationality criterion.

The assumption of EBG is that the original declarative description of the target concept lacks the information necessary to be able to execute a program to achieve the target. Of course a logic programming language *can* execute a declarative description provided that there is enough domain knowledge to be able prove the concept. So EBG becomes a simple matter of tracing Prolog's execution of a program and generalising that trace. This is done by replacing constants by variables and dropping constraints introduced by the specific example.

6. Logic and the Future

In this section we discuss some of the challenging problems in logic-based learning which are currently under investigation.

Incremental Learning. A robot, learning legal behaviour in a task environment, must necessarily use an incremental learning method where concepts are built and refined after making observations and performing experiments. One of the main difficulties in learning by experimentation in a reactive environment is that experiments can never validate the robot's theory of behaviour of the world. Experiments usually involve noisy data, they can cause damage to the environment, they may cause misleading side-effects. So it is more than likely that at some point in its exploration of the world, the robot will have not only an incomplete theory, but also an incorrect one. Thus it is necessary to have a system which is robust enough to be able to repair its knowledge base as new information is acquired.

Muggleton's induction theory may part of the solution to this problem. It is often the case that the robot learner has acquired enough experience to form a correct theory for the behaviour it has observed, but it actually has an incorrect theory because early experiments gave misleading results. Even when later observations could correct the error, this will not happen unless the robot can reorganise its knowledge base to take into account the later evidence. A learner which uses a compaction algorithm to maintain its knowledge base can reorganise concepts and create new concepts. Evidence supporting a correct concept will result in the creation of new clauses. The already existing concept will fall into disuse by not being supported by new evidence. Where the correct and incorrect versions of the same concept are contradictory, it may be necessary to choose between the two by looking at the amount of supporting evidence each has.

Limiting Search. An advantage of having fixed description language is that it is easier to control the search space of a learning algorithm. In section 4 we discussed a number of aspects of search. Buntine's (1986) analysis of the complexity of the search space will aid in developing strategies and heuristics for improving the time complexity of the learning algorithms. An attempt at this was made by Sands (1984).

7. Conclusion

We have tried to bring together the work of a number people to show how they have contributed to a coherent theory of learning where logic programs are the basis of the representation language for concepts. It has been claimed that this representation encourages a simple yet powerful approach to induction which is general enough to unite apparently different styles of learning: data-driven and model-driven, explanation-based and inductive learning.

References

Banerji, R.B. (1964). A Language for the Description of Concepts, *General Systems,* **9**.

Banerji, R.B. (1969). *Theory of* Problem Solving - An Approach to Artificial Intelligence, American Elsevier, New York.

Banerji, R.B. (1980). *Artificial Intelligence: A Theoretical Approach*, North Holland, New York.

Buntine, W. (1986). Generalised Subsumption, in *Proceedings of the European Conference on Artificial Intelligence*, London.

Carbonell, J.G. and Gil, Y. (1987). Learning by Experimentation, in *Proceedings of the Fourth International Machine Learning Worksho*p, ed. Pat Langley, pp. 256-266, Morgan Kaufmann, Los Altos.

Cohen, B.L. (1978). *A Theory of Structural Concept Formation and Pattern Recognition*, Ph.D. thesis, Department of Computer Science, University of New South Wales.

Hume, D. (1985). *Magrathea: A 3-D Robot World Simulation*, Honours thesis, Department of Computer Science, University of New South Wales, Sydney, Australia.

Kedar-Cabelli, S.T. and McCarty, L.T. (1987). Explanation-Based Learning as Resolution Theorem Proving, in *Proceedings of the Fourth International Machine Learning Workshop,* ed. Pat Langley, Morgan Kaufmann, Los Altos.

Mitchell, T.M., Keller, R.M. and Kedar-Cabelli, S.T. (1986). Explanation-Based Generalisation: A Unifying View, *Machine Learning*, **1**, pp. 47-80.

Mitchell, T.M., Utgoff, P.E., Nudel, B. and Banerji, R.B. (1981). Learning Problem Solving Heuristics Through Practice, in *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pp. 127-134, Vancouver.

Muggleton, S. (1987). Duce, An Oracle Based Approach to Constructive Induction, in *Proceedings of the International Joint Conference on Artificial Intelligence*, Milan.

Muggleton, S. and Buntine, W. (1987). *Towards a Constructive Induction in First-Order Predicate Calculus*, Turing Institute working paper.

Plotkin, G.D. (1970). A Note on Inductive Generalisation, in *Machine Intelligence 5*, pp. 153-163, ed. B. Meltzer and D. Michie, Edinburgh University Press.

Plotkin, G.D. (1971). A Further Note on Inductive Generalisation, in *Machine Intelligence 6*, pp. 153-163, ed. B. Meltzer and D. Michie, Elsevier, New York.

Robinson, J.A. (1965). A Machine Oriented Logic Based on the Resolution Principle, *Journal of the ACM*, **12**(1) pp, 23-41.

Sammut, C.A. (1981). Concept Learning by Experiment, in *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*,Vancouver.

Sammut, C.A. and Banerji, R.B. (1986). Learning Concepts by Asking Questions, in *Machine learning: An artificial intelligence approach (Vol. 2)*, ed. R.S. Michalski, T.M Mitchell and J.G. Carbonell, Los Altos, CA, Morgan Kaufmann.

Sammut, C.A. and Hume, D.V. (1987). Observation and Generalisation in a Simulated Robot World, in *Proceedings of the Fourth International Machine Learning Workshop*, ed. Pat Langley, pp. 267-273, Morgan Kaufmann, Los Altos.

Shapiro, E.Y. (1981). *Inductive Inference of Theories from Facts*, Technical Report 192, Yale University.

Sands, A. (1984). *A Search Strategy and Heuristics for a Concept Learner*, Honours thesis, Department of Computer Science, University of New South Wales, Sydney, Australia.

Utgoff, P.E. (1986). *Machine Learning of Inductive Bias*, Kluwer Academic Press, Boston.

Vere, S.A. (1975). Induction of Concepts in the Predicate Calculus, in *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, pp. 281-287, Tbilisi, USSR.

Vere, S.A. (1977). Induction of Relational Productions in the Presence of Background Information, in *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*.