Applications of Inductive Logic Programming

Ivan Bratko(1), Stephen H. Muggleton (2)

(1) University of Ljubljana, Faculty of Electrical Eng. and Computer Sc., Slovenia.

(2) Oxford University Computer Laboratory, UK.

Abstract

A logic-based approach to Machine Learning, called Inductive Logic Programming (ILP), is outlined, and several applications of ILP are reviewed. These benefit from the ability of ILP to use background knowledge in the learning process. The results from a variety of experimental applications of ILP are presented together with a discussion of the relative advantages of ILP and other Machine Learning approaches.

1 Introduction

Techniques of Machine Learning have been successfully applied to various problems [1, 13]. Most of these applications rely on attribute-based learning, exemplified by the well-known programs CART [5] and C4.5 [23], which induce decision trees from examples. The advantages of attribute-based learning are: relative simplicity, efficiency, and existence of effective techniques for handling noisy data. However, attribute-based learning is limited to non-relational descriptions of objects in the sense that the learned descriptions do *not* specify *relations* among the objects' parts. Attribute-based learning thus has the following strong limitations:

- the background knowledge can be expressed in rather limited form,
- the lack of relations makes the concept description language inappropriate for some domains.

Examples of such domains are presented in this paper.

An attempt to overcome the limitations of attribute-based learning has led to a number of recent development of several programs which learn at the level of first-order predicate logic. These include FOIL [22], Golem [16], LINUS [14] and Progol [24]. This has led to the inception of a new area of Machine Learning called *Inductive Logic Programming* [17] (for recent developments see [18, 15, 20]). ILP is characterised by its lock-step development of theory, implementations and applications.

The learning problem in ILP is normally stated as follows: given *background knowledge* B, expressed as a set of predicate definitions, positive examples E^+ and negative examples E^-, an ILP system will construct a logic formula H such that:

1. all the examples in E^+ can be logically derived from $B \wedge H$, and

2. no negative example in E^- can be logically derived from $B \wedge H$.

Typically for ILP systems B, H, E^+ and E^- will each be logic programs. $B \wedge H$ is simply the logic program B extended by the logic program H. The use of Prolog throughout allows for a highly versatile representation language for all constituents of the problem. This versatility is reflected in the wide variety of ILP applications. ILP differs from other machine learning approaches owing to its insitence on a particular representation language. This has advantages in integrating techniques and theory with those inherited from the Logic Programming school.

One of the main advantage of ILP over attribute-based learning is that it enables the user to provide domain-specific background knowledge to be used in learning. The use of background knowledge enables the user both to develop a suitable problem representation and to introduce problem-specific constraints into the learning process. By contrast attribute-based learners typically learn more or less from scratch. So in ILP, if the problem is to learn to distinguish cyclic from acyclic graphs, the graphs can be introduced by representing their edges as background knowledge. In addition a recursive definition of the notion of a path within a graph can be provided. If the problem is to learn about properties of chemical compounds, the molecular structures can be introduced as background knowledge in terms of the atoms and bonds between them. If the task is to automatically construct a model of a physical system from the observed behaviours, mathematical apparatus that is considered useful for the modelling domain is included in the background knowledge. Application of ILP involves development of a good representation of the examples together with relevant background knowledge. A general purpose ILP system is then applied.

The ILP framework can also be applied to automatic program synthesis from examples as follows. The existing, known predicates are introduced to a general ILP system as background knowledge. The target program is specified by examples of its input/output vectors.

A common ILP exercise of this kind is the induction of the quick-sort program from examples, saying for instance that the list [4,1,2] sorts into [1,2,4]. Suitable background knowledge contains the definition of the predicates for list concatenation, and for partitioning of a list, with respect to some value, into the lists of "small" and "big" elements. Using this background knowledge and some ten examples and counter examples, a typical ILP system will induce the known Prolog program for quick-sort in a few seconds of CPU time.

In the rest of this paper we describe selected applications of ILP. We chose those applications that specifically benefit from the ILP's predicate logic descriptions, and from the background facility in ILP.

2 Finite Element Mesh Design

Finite element (FE) methods are used extensively by engineers and modelling scientists to analyse stresses in physical structures. These structures are represented quantitatively as finite collections of elements. The deformation of each element is computed using linear algebraic equations. Finite element methods require that the objects being modelled are partitioned into a finite number of elements, resulting in a finite element mesh (Figure 1). In order to design a numerical model of a physical structure it is necessary to decide the appropriate resolution of the mesh. Considerable expertise is required in choosing these resolution values. Too fine a mesh leads to unnecessary computational overheads when executing the model. Too coarse a mesh produces intolerable approximation errors.

Normally some regions of the object require a denser mesh whereas in other regions a coarser mesh still suffices for good approximation. There is no known general method that would enable automatic determination of optimal, or even reasonably good meshes. However, many examples of successful meshes for particular objects have been accumulated in the practice of FE computations. These meshes can be used as sources of examples for learning about the construction of good meshes.

In general the mesh depends on the geometric properties of the object, the forces acting on it, and the relations between different components of the object. The mesh density in a region of the object depends also on the adjacent regions. Because of these relational dependences, the ILP approach most naturally applies to the mesh design problem.

In the application of ILP to this problem [8], an object to be partitioned is represented as (1) a set of edges, (2) the properties of the edges, and (3) relations among the edges. These properties and relations are represented as part of background knowledge by predicates, such

Figure 1: Partition of an object into a finite element mesh.

as: `short(Edge)`, `loaded(Edge)`, `not_loaded(Edge)`, `two_side_fixed(Edge)`, `neighbour_xy(Edge1,Edge2)`, etc. In experiments to learn a characterization of the density of a mesh in terms of these relations, ten meshes known to be numerically adequate have been used as sources of examples for learning. The target relation to be learned is: `mesh(Edge,N)` where `Edge` is the name of an edge in the structure, and `N` is the recommended number of finite elements along this edge. The available meshes comprise several hundreds of edges. Each edge is used as an example for learning, and typically some additional positive examples are derived from the meshes. The typical number of examples is between 300 and 600. Negative examples are generated by a form of closed-world assumption which gives rise to several thousands of negative examples.

Several relational learning algorithms were tried on this data including Golem [16], LINUS [14], FOIL [22] and CLAUDIEN [7]. The resulting set of rules were of interest to expert users of the finite element methods. According to their comments, these rules reveal interesting relational dependences. Here we give an interesting example of such a rule (in Prolog syntax as output by Golem, except that the variables were mnemonically renamed).

```
mesh( Edge, 7) :-
  usual_length( Edge),
  neighbour_xy( Edge, EdgeY),
  two_side_fixed( EdgeY),
  neighbour_zx( EdgeZ, Edge),
  not_loaded( EdgeZ).
```

This rule says that partitioning `Edge` into 7 elements is appropriate if `Edge` has "usual length", and has a neighbour `EdgeY` in the xy-plane so that `EdgeY` is fixed at both ends, and `Edge` has another neighbour `EdgeZ` in the xz-plane so that `EdgeZ` is not loaded. The following recursive rule was also generated by Golem.

```
mesh( Edge, N)  :-
  equal( Edge, Edge2),
  mesh( Edge2, N).
```

This rule is interesting as it expresses, by recursion, a recurrent pattern in the structure. The rule observes that an edge's partition can be determined by looking for an edge of the same length and shape positioned symmetrically in the same object. In other words, this can be viewed as Golem's discovery that an edge may inherit a suitable partition from similar edges in the structure. Of course, for this rule to be computationally useful, at least some of the equivalent edges must have its partition determined by some other rule.

The accuracy of the induced rule sets was investigated in detail in [8]. One method for estimating accuracy is cross-validation whereby a subset, say 90%, of all the available examples (that is edges) are used for learning, and the remaining examples are used for testing. Using this method, the test set accuracy of the rules induced by Golem was (on average) found to be as follows: the rules suggested correct partition of an edge into finite elements in 78% of all test cases, incorrect in 2% of the cases, and 20% of the test edges remained undecided (not covered by the induced clauses). Although the proportion of undecided edges here seems rather high, it is within an acceptable range for the practice of mesh design. Because of some general local consistency constraints used in mesh generators, many of the omissions can be automatically recovered.

3 Predicting the mutagenicity of chemical compounds

The construction of new scientific knowledge from real-world data remains an active focus for machine learning. One such problem is the Structure/Activity Relationships (SAR) of chemical compounds. This forms the basis of rational drug design. One widely used method of SAR stems from the work of Hansch [11] and uses regression/discrimination to predict activity from molecular properties such as hydrophobicity, sigma effect, molar reflectivity and LUMO (the energy of the Lowest Unoccupied Molecular Orbital). This and many other traditional approaches are limited in their representation of molecular connectivity and structure. They take into account the global attributes of a molecule, but do not comprehensively consider the *structural relationships* in the molecule. Thus some possibly important information, comprised as patterns in the molecular structure, may remain unexploited.

The ILP approach allows, however, that the complete structural information is taken into account. An ILP system *Progol* has been applied to the problem of identifying Ames test mutagenicity within a series of heteroaromatic nitro compounds [21, 24]. Hansch and coworkers have studied 230 compounds using classical regression [6]. For 188 compounds, they successfully obtained a linear regression function using hydrophobicity, LUMO and two hand-crafted binary attributes indicative of some structural properties. This regression formula predicts high mutagenicity with very acceptable accuracy. However the remaining 42 compounds could not be successfully modelled by regression and no structural principles were proposed. This subset of 42 compounds will be therefore referred to as "regression unfriendly". *Progol* was applied to this mutagenicity data using the split of the compounds into those with high mutagenicity and the rest as suggested by Hansch and coworkers. All the compounds were represented relationally in terms of atoms, bonds and their partial charge. This information was automatically generated by the modelling program QUANTA^{TM}, and was represented as about 18300 Prolog facts (unit Horn clauses) for the entire set of 230 compounds. For the 188 compounds found to be amenable to regression, the additional Hansch attributes of LUMO and hydrophobicity were also provided. All this was supplied to *Progol* as background knowledge for learning. For these compounds, *Progol* constructed the following theory. A compound is highly mutagenic if it has (1) a LUMO value $\leq$ -1.937; or (2) a LUMO value $\leq$ -1.570 and a carbon atom merging six-membered aromatic rings; or (3) a LUMO value $\leq$ -1.176 and an aryl-aryl bond between benzene rings; or (4) an aliphatic carbon with partial charge $\leq$ -0.022. The theory has an estimated accuracy of 89%. This matches the accuracy of both the regression analysis of Hansch and coworkers, and a more recent effort using neural networks [26]. It should be noted, however, that *Progol*'s theory is easier to comprehend and was generated automatically, without access to any structural indicator variables hand-crafted by experts.

The advantage of ILP, however, became particularly clear on the remaining subset of the 42 "regression unfriendly"' compounds. For these, *Progol* derived a single rule with an accuracy of 88% estimated from a leave-one-out validation (Figure 2). This is significant at $P < 0.001$. In contrast, linear regression and linear discrimination on the parameters used by Hansch and coworkers yield theories with accuracies estimated at 69% and 62% which are no better than random guesses supplied with the default accuracy of 69%. Perhaps even more important than the predictive accuracy, *Progol*'s rule provides the new chemical insight that the presence of a five-membered aromatic carbon ring with a nitrogen atom linked by a single bond followed by a double bond indicates mutagenicity. *Progol* has therefore identified a new structural feature that is an alert for mutagenicity.

4 Some other applications of ILP

Biological clasification of river water quality. River water quality can be monitored and assessed by observing various biologocal species present in the river. In particular, the river-bed macro-invertebrates are considered to be suitable indicators of the quality of water. Different species have different sensitivity to pollutants, and therefore the structure of the macro-invertebrate community in a river is well correlated with the degree and type of pollution. Džeroski et al. [9] used ILP to analyse the relation between the samples of macro-invertebrates and the quality class of water. For learning, they used 292 field samples of benthic communities taken from British Midlands rivers, classified by an expert river ecologist into five water quality classes. They constructed a relational representation of these samples and used the ILP systems Golem [16] and CLAUDIEN [7] for inducing logic clauses from the data. The induced clauses were judged by experts to be intuitively appealing and largely consistent with their knowedge. In particular, the experts appreciated the symbolic explicitness of the generated descriptions. They considered this as a major advantage over neural net learning that was also applied to the same data.

Biomolecular modelling. ILP applications in biomolecular modelling aim to improve the understanding of the inter-relationships of chemical formula, three-dimensional structure, and function of molecules of biological importance. An overview of such applications of ILP can be found in [25]. These ILP applications involved applying Golem [16] to protein secondary structure prediction [19], prediction of β-sheet topology and qualitatively modelling the structure activity relationship (QSAR) of a series of drugs [12]. For secondary structure prediction Golem yielded predictive accuracies well in excess of any other contemporary approach. In the case of QSAR predictive accuracies were not significantly higher than those produced by linear regression. However, in all three studies Golem discovered rules that provided insight into the stereochemistry of the system. Statistical techniques and neural networks do not do so, and are thus highly impaired for scientific discovery problems.

Inducing program invariants with ILP. In formally proving the correctness of procedural programs, one needs to find suitable conditions that always hold at given points in the program. Such a precondition has to be sufficiently strong to imply the postcondition of the program. Of particular interest is the problem of finding suitable conditions that are true inside program loops, called loop invariants. In general, the construction of loop invariants is considered difficult, and is usually done simply by guessing. Bratko and Grobelnik [3] explored the idea that ILP techniques can be used for automatically constructing loop invariants. A program that is to be proved correct can be executed, and the resulting execution traces can be used as learning examples for an ILP system. The states of the program variables at a given point in the program represent positive examples for the condition associated with that point in the program. Negative examples can be generated by employing a kind of "controlled closed-world assumption". In [3] suitable loop invariants were straightforwardly induced for simple programs that are used in typical correctness proof exercises. The automatic induction of an invariant for a parallel program was also demonstrated. The scaling up of this approach to larger programs has not been investigated yet.

Data refinement in program design. In program construction from higher order specification, functions in the specification language (higher level) are to be implemented in the target language (lower level). Thereby abstract data types at the higher level are to be refined into concrete data types at the target language level. For example, sets can be reified into lists. In [3] this refinement problem is formulated in the ILP framework. As an illustration, the general ILP program Markus [10] was used to implement by induction the set union operation from abstract, high level specification.

Innovative design from first principles. Bratko [2] formulated an approach to innovative design as an ILP problem. The design process is viewed as the process of structuring available elementary components in such a way that they together realize some specified target behaviour. The approach addresses the design from “first principles” in the sense that the functional behaviour of an artifact is derived from the *physics* of the elementary components available to the designer. The approach involves: specification of the target artifact by examples of its intended behaviour, *qualitative physics* definition of the behaviour of the elementary components available, and ILP as the mechanism for conceptually constructing the device. As an illustration, the Markus program [10] was applied to constructing simple electric circuits from examples of their intended behaviour and the qualitative physics of some simple electrical components.

Qualitative system identification A fundamental problem in the theory of dynamic systems is system identification. This can be defined as follows: given examples of the behavior of a dynamic system, find a model that explains these examples. Motivated by the hypothesis that it should be easier to learn qualitative than quantitative models, Bratko et al. [4] formulated the qualitative identification problem as an ILP problem. In their work, models are sets of Qualitative Differential Equations (QDEs) that constrain the values of the system variables. A Prolog implementation of QDE constraints normally used in qualitative physics is provided as background knowledge for learning. Example behaviors of the modelled system are used as positive training examples, while negative examples are generated as near misses. Models of simple dynamical systems have been induced using general ILP systems.

5 Conclusion

ILP has been applied to difficult, industrially relevant and not yet satisfactorily solved problems. In the main applications described, the results obtained with ILP using real industrial or environmental data are better than with any other known approach, with or without ML. In many of these applications, the users – domain specialists – are becoming increasingly interested in the understandability, or meaningfulness, of the induced concept descriptions. This helps them to obtain new insights in their problem domains.

In all applications, general purpose ILP systems were used. Accordingly, a typical ILP application amounts to designing a good relational representation of the problem, including the definition of relevant background knowledge. A major strength of ILP systems, compared with other Machine Learning approaches, is that they accept background knowledge in the form as general as Prolog programs. A major obstacle to more effective use of ILP at present is the relative inefficiency of the existing ILP systems, and their rather limited facilities for handling numerical data.

Acknowledgements

The authors would especially like to acknowledge the input of Ashwin Srinivasan, Ross King and Michael Sternberg for their involvement in experimental results on structural molecular biology applications. This work was supported partly by the Esprit Basic Research Action ILP (project 6020), an SERC Advanced Research Fellowship held by Stephen Muggleton and a Research Fellowship supporting Stephen Muggleton at Wolfson College, Oxford.

References

[1] Bratko, I. (1993) Applications of Machine Learning: towards knowledge synthesis. *New Generation Computing*. Vol. 11, 343-360.

[2] Bratko, I. (1993b) Innovative design as learning from examples. *Proc. Int. Conf. Design to Manufacture in Modern Industries*, Bled, Slovenia, June 1993.

[3] Bratko, I., Grobelnik, M. (1992) Inductive learning applied to program construction and verification. In: *AI techniques for Information Processing* (ed. J. Cuena) North-Holland 1993. Also in: *Proc. ILP'93 Workshop*, Bled, Slovenia, April 1993.

[4] Bratko, I., Muggleton, S., Varšek, A. (1992) Learning qualitative models of dynamic systems. In: *Inductive Logic Programming* (ed. S. Muggleton) London: Academic Press.

[5] Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J. (1984) *Classification and Regression Trees.* Belmont, CA: Wadsworth.

[6] Debnath, A.K., Lopez de Compadre, R.L., Debnath, G., Schusterman, A.J. and Hansch, C. (1991) *Jnl. Medicinal Chemistry*, **34**, 786–797.

[7] De Raedt, L. and Bruynooghe, M. (1993). A theory of clausal discovery. *Proc. Thirteenth International Joint Conference on Artificial Intelligence*, pages 1058–1063. Morgan Kaufmann, San Mateo, CA.

[8] Dolšak, B, Bratko, I., Jezernik, A. (1994) Finite-element mesh design: an engineering domain for ILP application. Proc. Fourth Int. Workshop on Inductive Logic Programming ILP-94, Bad Honnef/Bonn.

[9] Džeroski, S., De Haspe, L., Ruck, B.M., Walley, W.J. (1994). Classification of river water quality data using machine learning. In *Proc. Fifth International Conference on the Development and Application of Computer Techniques to Environmental Studies (ENVIROSOFT'94).* To appear.

[10] Grobelnik, M. (1992) Markus: an optimized model inference system. *Logic Approaches to Machine Learning Workshop*, Vienna, August 1992.

[11] Hansch, C., Malong, P.P., Fujita, T. and Muir, M. (1962) *Nature*, **194**, 178–180.

[12] King, R.D., Muggleton, S., Lewis, R.A. and Sternberg, M.J.E. (1992). Drug design by machine learning : the use of inductive logic programming to model the structure-activity relationship of trimethoprim analogues binding to dihydrofolate reductase. *Proc. Nat. Acad. Sci.*, USA. 89, 11322-11326.

[13] Langley, P., Simon, H. (1995) Applications of Machine Learning and Rule Induction. CACM, this issue.

[14] Lavrač, N., Džeroski, S., and Grobelnik, M. (1991) Learning nonrecursive definitions of relations with LINUS. In *Proc. Fifth European Working Session on Learning*, pages 265–281. Springer, Berlin.

[15] Lavrač, N., Džeroski, S. (1994) *Inductive Logic Programming: Techniques and Applications.* Chicester, England: Ellis Horwood.

[16] Muggleton, S. and Feng, C. (1990). Efficient induction of logic programs. *Proceedings of the first conference on algorithmic learning theory*, Arikawa, S., Goto, S., Ohsuga, S. and Yokomori, T., eds. (Japanese Society for Artificial Intelligence, Tokyo) pp. 368-381.

[17] Muggleton, S. (1991). Inductive Logic Programming. *New Generation Computing* 8(4) 295-318.

[18] Muggleton, S. (ed) (1992) *Inductive Logic Programming.* Academic Press.

[19] Muggleton, S., King, R.D. and Sternberg, M.J.E. (1992). Protein secondary structure prediction using logic. *Prot. Eng.* 5(7) 647-657.

[20] Muggleton, S., De Raedt L. (1994). Inductive Logic Programming: Theory and Methods. *Journal of Logic Programming* 19,20 629-679.

[21] Muggleton S., (1994). Bayesian Inductive Logic Programming. *Proceedings of the eleventh international conference on machine learning*, Cohen, W., Hirsh, H., eds. pp. 371-379.

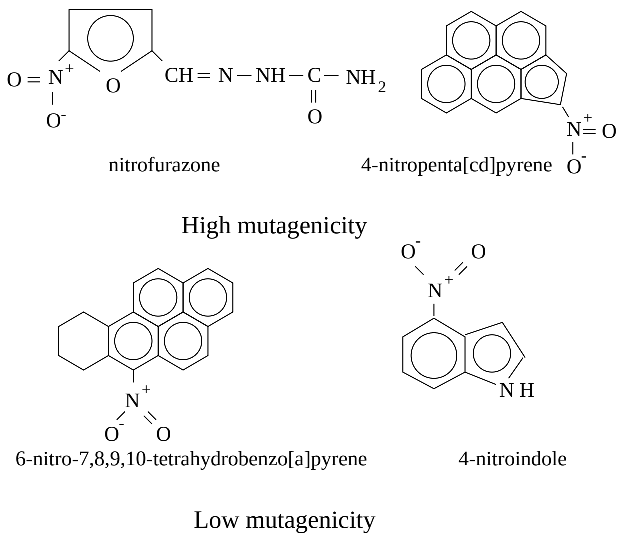
[22] Quinlan, J.R. (1990) Learning logical definitions from relations. *Machine Learning, Vol. 5*, pp 239-266.

[23] Quinlan, J.R. (1993) *C4.5: Programs for Machine Learning.* San Francisco: Morgan Kaufmann.

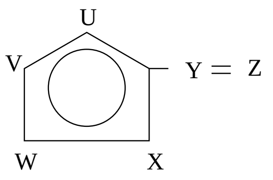
[24] Srinivasan, A., Muggleton, S.H., King, R.D., Sternberg, M.J.E. (1994) Mutagenesis: ILP experiments in a non-determinate biological domain. In *Proc. Fourth Int. Workshop on Inductive Logic Programming ILP-94*, Bad Honnef/Bonn.

[25] Sternberg M., King R., Lewis R., Muggleton S. (1994) Application of Machine Learning to Structural Molecular Biology. *Philosophical Transactions of the Royal Society B*, 344 365-731.

[26] Villemin, D., Cherqaoui, D. and Cense J.M. (1993) *J. Chim. Phys.* **90**, 1505–1519.



(A)



(B)

Figure 2: "Regression unfriendly" compounds and the structural feature found by *Progol*. **(A)** Some of the compounds found not to be amenable to analysis by statistical methods of regression or discrimination [6]. No structural rules/alerts have previously been proposed for mutagenesis in these compounds. **(B)** *Progol* identified the alert of a double bond conjugated to a five membered aromatic ring via a carbon atom. The atoms U–Z do not necessarily have to be carbon atoms. This is the most compressive explanation for mutagensis for the 42 compounds possible within the hypothesis language used by *Progol*. The alert is present in the two high mutagenic compounds shown in **(A)** and not present in the two low mutagenic compounds.