Applications of Inductive Logic Programming

Ivan Bratko
Basser Department of Computer Science*
The University of Sydney, Australia
and
Dept. of Electrical Eng. and Computer Sc.,
Ljubljana University
Tržaška 25, 61000 Ljubljana, Slovenia
Ivan.Bratko@fer.uni-lj.si

Ross King
Biomolecular Modelling Laboratory
Imperial Cancer Research Fund
P.O. Box 123
44 Lincoln's Inn Fields
London WC2A 3PX, U.K.
ross@fir.lif.icnet.uk

Abstract

Some applications of Inductive Logic Programming (ILP) are presented. Those applications are chosen that specifically benefit from *relational* descriptions generated by ILP programs, and from ILP's ability to accommodate background knowledge. Applications included are: drug design, predicting the secondary structure of proteins, and design of finite-element meshes. Some other applications are briefly described. The practical advantages and disadvantages of ILP learning are discussed.

1 Introduction

In this paper we present some applications of Inductive Logic Programming (ILP). We chose those applications that specifically benefitted from *relational* descriptions generated by ILP programs, and from ILP's ability to make use of relational information known prior to learning. This information which depends on the domain of application is accommodated into the learning process as "background knowledge".

In the main part of the paper we describe in some detail applications to:

1. Drug design
2. Predicting the secondary structure of proteins
3. Design of finite-element meshes

The first two applications belong to the area of biomolecular modelling. This aims to understand the inter-relationships of chemical formula, three-dimensional structure, and function of molecules of biological importance. The two problems in biomolecular modelling have been tackled by the ILP program GOLEM [35]. The first problem is to model quantitatively the structure activity relationship of a series of drugs [26, 27, 21, 22]. The second topic is the prediction of the local secondary structure of a protein from its sequence [36]. In both problems rules are learned that: perform as well or better than conventional approaches, and provide insight into the stereochemistry of the systems.

Our third application, the finite-element mesh design arises in numerical computation. Finite element methods require that the object to be analysed is partitioned into finite elements. The resulting partition is called a finite-element mesh. For a given object, among enormous number of possible meshes only some are suitable for finite element computation. There is no known general method that would enable automatic determination of optimal, or reasonably good meshes. ILP techniques have been successfully applied by Dolšak et al. [12, 14, 13] towards the automation of mesh design.

*Until Jan. 1, 1994

In addition to the three main applications, we briefly describe some other interesting applications: learning qualitative models of dynamic systems, program construction from high level specification, learning loop invariants in procedural programs, learning chess patterns, and innovative design.

2 Drug design

2.1 Background

Drug design starts with the discovery of a "lead" compound. This is a molecule with demonstrable but weak activity of the desired type; e.g. inhibition of the growth of cancer cells. In general the molecular mechanism of action of the lead drug is unknown, making it impossible to infer directly what changes would give the lead drug higher activity. It is therefore usual practice in rational drug design to synthesise chemically an exploratory series of drugs (analogues) that are variations of the lead drug. From these analogues a Quantitative Structure-Activity Relationship (QSAR) is induced which relates drug structure to activity. This QSAR can then be used to infer the activity of analogues that have not been synthesised. Any improved method of QSAR would be important both medically, with the discovery of better and safer drugs, and economically (the world drug market in 1989 was estimated to be worth $70 billion). The standard method for QSAR was proposed by Hansch, where the (Hansch) attributes of a series of analogues are linked to activity by an empirical multivariate regression equation [20, 18]. This multivariate equation is usually linear on the attributes and their squares. Many other statistical techniques have been used to form QSAR's, e.g. principal component analysis, linear discrimination [31]. In recent years neural networks have started to be widely used [47, 1, 3, 2]. Initial promising work has also been done applying machine learning to drug design [5, 30, 26, 27].

Two drug design problems were used to test the application of GOLEM: the inhibition of Escherichia Coli Dihydrofolate Reductase (DHFR) by pyrimidines [19, 47, 26, 21] (Figure 1), and the inhibition of rat/mouse tumour DHFR by triazines [46, 22].

2.2 Method

The input to GOLEM consisted of

1. The observed activities of the drugs represented as paired examples of greater activity, e.g.

   ```
   great(d20,d15)
   ```

   which states that drug no. 20 has higher activity than drug no. 15.

2. A representation of the structure of the drugs. This is probably the most fruitful part of the application of ILP

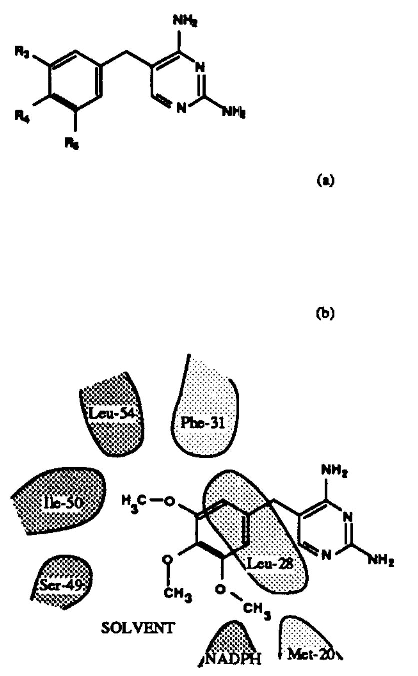

Figure 1: (a) the structure of trimethoprim analogues; (b) a cartoon of the interaction of trimethoprim with DHFR from X-ray structures (Champness, et al., 1986; Matthews, et al., 1985). Faint stippling indicates that the residue lies below the plane of the phenyl ring, darker stippling that the atoms are above.

programs. For the pyrimidine drugs chemical structure is represented in the form:

```
struc(d55, Cl, NH2, CH3).
```

which states that drug no. 55 has a Cl substituted at position 3, a NH2 group substituted at position 4, and a CH3 group substituted at position 5. For the triazine drugs chemical structure is represented in the form:

```
struc3(d217, Cl, absent).
struc4(d217, (CH2)4, subst14).
subst(subst14, SO2F, Cl).
```

This is the Prolog representation of the drug 217 - 3-Cl, 4-(CH2)C6H3-4'-Cl, 3'-SO2F. To represent the pyrimidine drugs in a propositional form takes 30 attributes, and the triazines 60 attributes.

3. The background knowledge of the chemical properties of the Substituents in which the properties were assigned heuristically. The particular properties are: polarity, size, flexibility, hydrogen-bond donor, hydrogen-bond acceptor, pi_donor, pi_acceptor, polarisability, branching and sigma_effect. This was represented using different predicates for each property and by typing the values, e.g. `polar(Br,polar3)` states that Br has polarity of value 3.

4. In-built arithmetic so information was explicitly given about the relative values of these properties for the substituent groups e.g.

```
gt(polar4, polar3).
```

The output generated by GOLEM consisted of induced definition of the relation

```
great( Drug1, Drug2)
```

in terms of the background predicates.

The application of GOLEM to drug design problems was compared on the above data with propositional symbolic methods, neural network and traditional statistical methods.

The following algorithms were compared: from propositional symbolic methods - CART [10], and M5 [40]; from neural networks - back-propagation [45]; and from traditional statistics - linear regression, and nearest-neighbour.

For the 55 pyrimidine drugs a five fold cross-validation study was carried out. There were 2198 background facts, 1394 positive facts, and 1394 negative facts.

For the 186 triazine drugs a six fold cross-validation study was carried out. For each split there were 2933 background, 1000 positive facts, and 1000 negative facts.

2.3 Results

In terms of the rank correlation of predicted activity of test drugs, there was found to be no significant difference between any of the methods on either dataset at the 5% level.

However, the output rules of GOLEM are more easily understood by synthetic chemists than the output of the rival methods (with the possible exception of linear regression using no squared attributes). The logical language of ILP is closer to that used by chemists than that of propositional symbolic methods, or the mathematical language of statistics and neural networks. It is possible to produce logical rules as a post-processing step from mathematical equations, but this is a rather roundabout way of doing things. Understandable rules are important: to avoid acceptance of correct but trivially inappropriate rules [41], and more importantly, to allow insight into the problem. A key goal in carrying out a QSAR study is to gain insight into the chemical problem, which a synthetic organic chemist can apply to the synthesis of new drugs. For both the pyrimidine and triazine datasets it was found possible using machine learning to produce effective and easily understandable rules.

An example of a rule for the pyrimidine data as a Prolog clause is:

```
great(A, B)  :-
   struc(A,Pos_a3,Pos_a4,_), struc(B,_,_,h),
   h_donor(Pos_a3,h_don0),
   pi_acceptor(Pos_a3,pi_acc0),
   polar(Pos_a3,Pol_a3), great0_polar(Pol_a3),
   size(Pos_a3,Siz_a3), less3_size(Siz_a3),
   polar(Pos_a4,_).
```

The interpretation in English is:

Drug A is better than drug B if
  drug A has a substituent at positions 3 with
  hydrogen donor = 0 and
  pi-acceptor = 0 and
  polarity > 3 and
  size < 3 and
  drug A has a substituent at positions 4 and
  drug B has no substituent at positions 5.

Taken together the pyrimidine rules generated by GOLEM can be related to what is known about the position of the pyrimidine drugs crystal structure of the protein E. coli DHFR (Figure 1) [32]. (There is no equivalent crystal structure for the triazines.) The pyrimidine rules state that each of the 3, 4, and 5 positions should not be hydrogen. This is in keeping with the suggestion [43] that an important role of the 4 position is to force the 3 and 5 position away from the 4 position. Both positions 3 and 5 are partially buried in a hydrophobic environment and this is consistent with the restrictions in the rules on polarity, size, hydrogen-bond donor, sigma effect, and flexibility. The rules suggest that substituents at position 4 should be pi_donors, such a condition will force this group to lie in the plane of the aromatic ring, and this has been suggested as a requirement for a favourable drug [44]. Because both positions 3 and 5 have similar chemical locations in DHFR, one cannot decide whether the rules relate exclusively to position 3 or 5, which explains the symmetry of the rules. This shows that it is possible using an ILP system to infer important information about the binding of drugs to proteins.

3 Secondary structure prediction

3.1 Background

Proteins are biological macromolecules whose activity, such as catalysis, results from its complex three-dimensional structure (called tertiary structure). However structure determination remains difficult so today there are around 500 different tertiary structures known. In contrast, the chemical structure of the protein, known as the amino-acid sequence, can readily be obtained from the gene sequence and today more than 30,000 sequences are known.

Experiments suggest that it should be possible to predict theoretically the three-dimensional structure of a protein from its sequence, for reviews see [4]. This problem, known as the protein folding problem, is one of the most important in molecular biology. Most proteins have segments of the chain that adopt a regular local conformation, known as secondary structure. There are two major types, α-helices and β-sheets, and typically between 5 to 15 residues of the chain can be assigned to one local region of secondary structure. Thus a protein chain that ranges between 50 and 1000 amino-acid residues will have several regions of α-helices and/or β-sheets linked by less regular (but still defined) conformation known as coil. The prediction of secondary structure is one step towards solving the protein folding problem.

Many early methods of secondary structure prediction were based on simple statistical analyses and these achieved about 60% accuracy for a three state (α,β, and coil) prediction [24]. Recently, approaches using larger data sets [16], and/or sophisticated analyses including neural networks [28, 42] yielded only marginal improvements of up to 70% accuracy. One method of improving prediction is to restrict the algorithm to a subset of all proteins that have just α-helices and coil. Recently neural networks by Kneller et al. [28] have been applied to this α/α class and yielded 76% accuracy. However a major problem with neural networks is that the weights do not provide insight into principles of folding. Here we report on the use of GOLEM to secondary structure prediction. This extends earlier work [25] and full details can be found in [36].

3.2 Method

The structural data consisted of a training set of 12 proteins with a testing set of 4 proteins that were not related (i.e. non-homologous). The absence of any homology between the training and testing set is essential for an evaluation of the power of any algorithm. We note that there are several studies where this requirement has been ignored.

The input to GOLEM was:

1. The observations of the location in the protein of the α-helices coded as: `alpha(protein_name,position)`. For example: `alpha(155C,110)` states that residue 110 in protein 155C is in an α-helix.

2. The background information of the amino-acid sequence coded as: `position(155C,110,v)` that states residue at position 110 in 155C is a valine.

3. The background knowledge about the chemical properties of each of the 20 different amino-acid residues. These properties were: hydrophobic, very_hydrophobic, hydrophilic, positive, negative, neutral, large, small, tiny, polar, aliphatic, aromatic, hydrogen_bond_donor, hydrogen_bond_acceptor, not_aromatic, small_or_polar, not_polar, aromatic_or_very_hydrophobic, either_aromatic_or_aliphatic, not_proline, not_lysine. These were coded as for example: `aliphatic(v)`.

4. Built-in arithmetic that explicitly coded information about the sequential relationship of residues.

3.3 Results

The first application of GOLEM yielded rules that led to a speckled prediction with individual residues not predicted as helical but within a run of helical residues. A bootstrapping procedure was followed where the predictions made by GOLEM were used as background knowledge and GOLEM learnt how to smooth the data. This was repeated twice and the resultant rules generalised by hand to include symmetry.

The accuracy of secondary structure prediction is conventionally expressed as the percentage of residues whose state is correctly predicted. The final prediction, after smoothing, was 78% and 81% on the training and testing sets. The better improvement on the testing set rather than the training data reflects the standard error in each of these values of about 2%. This result can be compared with the recent work on neural networks on α/α class proteins that was 76% accurate [28].

The rules described the properties of residues at different positions along the helix (e.g. Figure 2). Inspection showed that well-known chemical principles of α-helix formation are described in the rules, for example the preference for one face to be formed from oily (i.e. hydrophobic) residues. However new insights into protein folding which may be included in the patterns are not easy to discern.

4 Finite-element mesh design

This problem arises in numerical computation based on finite-element methods. For example, given an object and forces acting on it, finite-element methods can be used to compute the pressure and deformations throughout the object. Finite element methods, however, require that the object is partitioned into finite elements, resulting in a *finite element mesh.*

For each element of the mesh, constraints in the form of equations are stated. The constraints approximately state the physical laws modelling the behaviour of the individual elements. These approximations are sufficiently accurate if the elements are sufficiently small. Generally, the finer the mesh, the smaller the error. However, a dense mesh results in a large number of equations, leading to a lengthy computation when solving the corresponding system of equations. The complexity of computation is often measured in days or weeks of CPU time and can easily become prohibitive. The

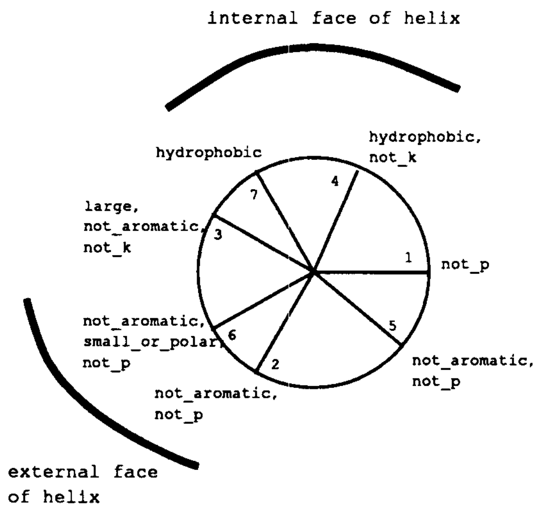

Figure 2: Schematic representation of a rule for secondary structure prediction.

problem, then, is to find a suitable compromise between the density and coarseness of the mesh.

Normally some regions of the object require 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 reasonably good meshes. However, expert users of finite element methods are capable of making good guesses about proper density of the mesh in various regions of the objects. Unfortunately, the experts have difficulties in forming general rules that would enable the automation of such guesses. 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.

Attribute-based learning is unsuitable for this learning problem because in general the mesh depends on the geometric properties of the object and forces acting on it and relations between various components of the object. The mesh density in a region of the object depends also on the adjacent regions.

Here we describe the application of ILP to this problem following Dolšak et al. [12, 13], also presented in [6]. An earlier experiment is described in [14].

An object to be partitioned is represented as a set of edges and the properties of and relations among the edges. These important aspects of the object are captured in this ILP application as properties and relations, such as:

```
short( Edge)
usual_length( Edge)
loaded( Edge)
not_loaded( Edge)
two_side_fixed( Edge)
neighbour_xy( Edge1, Edge2)
neighbour_xz( Edge1, Edge2)
...
```

The meaning of these relations is straightforward. For example, an edge is "two_side_fixed" if it is fixed at both ends. neighbour_xy(Edge1,Edge2) means that the edges are adjacent in an xy-plane. In an experiment to learn a characterisation of the density of a mesh in terms of these relations, five meshes known to be numerically adequate were used as sources of examples for learning. The relation to be learned was:

```
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 target definition of this relation is to be learned in terms of the properties and relations in the given structure. All the five meshes used comprised altogether 278 edges, that is 278 positive examples for learning. The number of finite elements along the edges varied between 1 and 17. In edges with high partition, say 10, it was assumed that a similar partition would still make a good mesh, so 10 ± 1 was considered acceptable and sometimes used as another positive example. Negative examples were generated according to the closed-world assumption: if the given partitioning of an edge was 3, say, then partitionings such as 4, 5, etc. were taken as negative examples. This finally gives the following number of facts for learning in this experiment:

357 positive examples
2840 negative examples
2132 background facts

Several relational learning algorithms were tried on this data: GOLEM [35], LINUS [23] and FOIL [39]. The results obtained with GOLEM were judged to be the most satisfactory. GOLEM generated a large number of rules, some of them being practically irrelevant. For example, although logically correct, they were computationally useless when applied to classifying new edges. On the other hand, some rules appeared useful. Fortunately it was possible to formalise the criteria for distinguishing useful rules from the others. These criteria were implemented as a short Prolog program (Dolšak 1991) for postprocessing the rules generated by GOLEM.

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. The following is an example of such a generated rule (the generated syntax is that of Prolog clauses):

```
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 an appropriate partitioning of Edge is 7 if Edge 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 is a recursive rule also generated by GOLEM:

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

This observes that an edge's partition can be determined by looking for an edge of the same length and shape positioned similarly in the same object. Of course, for this rule to be computationally useful, at least some of such equivalent edges must have its partition determined by some other rule.

The accuracy of the induced knowledge base was estimated by a cross-validation mehod. Thereby a subset of 10% of the example edges was effectively removed from the training set. The remaining 90% of the data was used for rule induction, and the so induced rules were applied to the removed 10% of the data now used as a test set. This was repeated ten times.

The results can be summarised as follows. On the average, the classification on the test set was correct in 78% of the tested edges, incorrect in 2% of the edges, and an edge remained unclassified (partition unknown) in 20% of the test edges. An edge remains unclassified if there is no induced rule covering the edge.

In another, more practically realistic evaluation attempt, the generated knowledge base was applied to determining a mesh for a completely new structure, one not used for learning. In this case, 67% of the edges were classified correctly, 22% incorrectly, and 11% remained unclassified.

These results, together with a well known rule of thumb for mesh design, were then used to automatically construct a complete mesh with a commercial automatic mesh generator, resulting in the optimal mesh.

5 Some other applications of ILP

Here we briefly describe some other applications of ILP although they have not been elaborated to the extent of the ones described above. Their contribution is mainly in the demonstration of possibilities and various problem formulations as ILP problems.

5.1 Learning qualitative models of dynamic systems

Bratko et al. [9] describe an approach, in the ILP framework, to learning qualitative models of dynamic systems. The formalism of Qualitative Differential Equations (QDE) is used as the representation for learned qualitative models. A Prolog implementation of QDE constraints is input as background knowledge, and example behaviours of the modelled dynamic system are used as examples for learning. So the general ILP problem formulation becomes instantiated for the purpose of learning qualitative models as follows:

$$QDEconstr \wedge QualitativeModel \vdash ExampleBehaviours$$

A definition for *QualitativeModel* is to be induced. Bratko et al. used the general ILP program GOLEM [35] to induce qualitative models in the above setting. It was possible to induce models of simple dynamic systems, such as simple oscillator or two connected containers system, usually called U-tube. Džeroski and Bratko [15] describe similar experiments using another general ILP program mFOIL.

5.2 Data refinement in program construction

Bratko and Grobelnik [8] show how ILP techniques can be applied to the problem of data reification. 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 reified into concrete data types at the target language level. For example, sets can be reified into lists. In [8] this refinement problem is formulated in the ILP framework. As an illustration, the general ILP program Markus [17] was used to implement the set union operation as list concatenation.

5.3 Inducing loop invariants

Bratko and Grobelnik [8] propose a way of using ILP to induce loop invariants needed to prove the correctness of procedural programs. The procedural 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 predicate associated with that point in the program. Negative examples are generated by employing a kind of "controlled closed-world assumption". This relies on the knowledge about the functional dependences among the variables in the program. Exploiting this, negative examples are generated by corresponding modifications of the positive examples. In [8] some simple programs typically used in correctness proof exercises were considered. Loop invariants for these programs were induced by straightforward application of general ILP programs including GOLEM [35], FOIL [39] and Markus [17].

5.4 Learning chess patterns

Morales [34] wrote a Prolog program for inducing definitions of meaningful chess patterns from example chess positions in which these patterns occur. The program was merely shown a board position and was not told which pieces belong to the pattern of interest. Typical patterns whose definitions were successfully induced by the program include fork, pin and skewer. Background knowledge consisted of primitive predicates including those specifying the board position, move legality, king in check etc. The basic ILP mechanism used in Morales' program is relative least general generalisation (RLGG) as in GOLEM.

5.5 Design from first principles

Bratko [7] formulated the problem of innovative design "from first principles" 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 proposed 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, Markus [17] was applied to constructing simple electric circuits from examples of their intended behaviour and the qualitative physics of some simple electrical components.

6 Discussion

In this section we discuss the practical advantages and disadvantages of ILP, particularly in comparison with other approaches to Machine Learning.

There have been many successful applications of Machine Learning [6, 29]. The most common and practically significant form of learning has been attribute-based learning. Well known families of such learning programs are TDIDT [38], AQ [33], and CN2 [11]. The following advantages of attributional learning contribute to its success in practical applications:

- It is computationally efficient and relatively well understood.
- It is easy to understand by the users and straightforward to apply.
- Very effective methods exist for handling noisy and incomplete data in attributional learning.

However, attribute-based learning also has strong limitations:

- Background knowledge can only be expressed in rather limited form.
- Lack of relational descriptions makes the concept description language inappropriate for some domains.

Attribute-based descriptions are essentially equivalent to propositional logic. Both learning examples and induced concept descriptions employ global attributes of objects and not relations among their parts. This is not sufficiently expressive for describing concepts in some application areas. Good examples of such problem areas described in this paper are biomolecular modelling and the finite-element mesh design. In these and in the other applications mentioned in the paper, the ability of using background knowledge in ILP was essential. For example, in learning qualitative models the whole mathematical basis (QDE constraints) for qualitative modelling was introduced as background knowledge.

For molecular modelling, using an ILP learning system, such as GOLEM, has advantages and disadvantages. Its main advantage is that it can naturally represent the structural (relational) features of chemicals. The representation used by GOLEM for the drugs and proteins is more natural and closer to the language of chemists than the propositional attribute based representation. An illustration of this is the fact that several drugs were excluded from the triazine study because they would greatly increase the complexity of the propositional representation - they had rare positional substitutions. Including these drugs in the trial would have doubled the number of attributes. In contrast, it would have been trivial to change GOLEM's representation to include these drugs. It is generally much easier to change an ILP representation than a propositional one. Some workers consider that a propositional representation is sufficient to describe chemical problems. This seems unlikely, while it is possible to go a long way with propositional learning schemes (e.g. CART and M5) by the clever creation of new attributes. It is difficult to see how a propositional learning algorithm would be capable of learning over arbitrary chemical structures (not just templates). It is also not always easy to create clever new attributes [37].

A disadvantage of many ILP systems is that arithmetic is not built in. In the molecular modelling applications it had to be made explicit (e.g. through the predicate greater_than). This lack of numerical facilities in most present ILP systems makes them more suitable for classification problems and not regression. However, recent systems are increasingly paying more attention to including numerical facilities. Another practical drawback of some well known ILP programs (GOLEM, FOIL) is that they require background knowledge to be represented as ground facts. Also, the restriction to determinate literals only in some systems has turned out to be rather limiting in some applications. A minor disadvantage with the form of GOLEM used in molecular modelling was that its method of searching the space of rule combinations was inefficient. A better method is needed of ensuring that the rules found cover as many examples as possible. It may have been better to introduce exceptions to rules by means of the Closed-World Specialization algorithm [48].

On the computational complexity side, ILP systems in general tend to be slower than propositional systems, and tend to be able to be applied to fewer examples. Therefore if the data is not structured in some way it is probably best to use a propositional learning method. But if the data is structured then an ILP system is the one to chose.

Acknowledgements

I.B. would like to thank for support from the Australian Research Council (ARC), Commision of the European Communities (ESPRIT III Basic Research Project no. 6020 Inductive Logic Programming), and the Slovenian Ministry for Science and Technology. R.D.K. is supported by the Imperial Cancer Research Fund. He thanks M.J.E. Sternberg, J.D. Hirst, S. Muggleton, and R.A. Lewis. In addition, the authors would like to thank David Powers for help in finalising this paper.

References

[1] Andrea, T.A. and Kalayeh, H. (1991). Applications of neural networks in quantitative structure-activity relationships of dihydrofolate reductase inhibitors. *J. Med. Chem.* 34, 2824-2836.

[2] Aoyama, T., and Ichikawa, H. (1992). Neural Networks as Nonlinear Structure-Activity Relationship Analysers. Useful Functions of the Partial Derivative Method in Multilayer Neural Networks. *J. Chem. Inf. Comput.Sci.* 32, 492-500.

[3] Aoyama, T., Suzuki, Y., and Ichikawa, H. (1989). Neural networks applied to pharmaceutical problems. I. Method and application to decision making. *Chem. Pharm. Bull.* 39, 2558-60.

[4] Blundell, T.L., Sibanda, B.L., Sternberg, M.J.E. and Thornton, J.M. (1987). Knowledge-based prediction of protein structures and the design of novel molecules. *Nature* (London) 326, 347-352.

[5] Bolis, G., Pace, L. D., and Fabrocini, F. (1991). A machine learning approach to computer-aided molecular design. *J. Comp.-Aided Mol. Des.* 5, 617-628.

[6] Bratko, I. (1993a) Applications of Machine Learning: towards knowledge synthesis. *New Generation Computing*, Vol. 11, pp. 343-360, OHMSHA Ltd. and Springer-Verlag (Tokyo).

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

[8] Bratko, I., Grobelnik, M. (1992) Inductive learning applied to program construction and verification. *Proc. AIFIPP Workshop*, Madrid, Sept. 1992. Revised version: *Proc. ILP'93 Workshop*, Bled, Slovenia, April 1993.

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

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

[11] Clark, P., Niblett, T. (1989) The CN2 induction algorithm. *Machine Learning*, Vol. 3, no. 4, 261-284.

[12] Dolšak, B. (1991) Determining the geometric model of finite element meshes using AI methods. University of Maribor: CAD Center (M.Sc. thesis) Maribor, Slovenia.

[13] Dolšak, B., Jezernik, K., Bratko, I. (1992) A knowledge base for finite element mesh design. *Proc. ISSEK Workshop 92*, Bled, Slovenia, sept. 1992.

[14] Dolšak, B., Muggleton, S. (1991) The application of Inductive Logic Programming to Finite Element Mesh Design. *Proc. First Int. Workshop on Inductive Logic Programming*, Viana do Castelo, Portugal, 1991. Also in: *Inductive Logic Programming* (ed. S. Muggleton) London: Academic Press, 1992.

[15] Džeroski, S., Bratko, I. (1992) Handling noise in inductive Logic Programming. *Proc. Second Int. ILP Workshop on Inductive Logic Programming*, Tokyo, Japan. ICOT TM-1182.

[16] Gibrat, J.F., Garnier, J. and Robson, B. (1987). Further developments of protein secondary structure prediction using information theory. New parameters and consideration of residue pairs. *J Mol Biol* 198, 425- 443.

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

[18] Hansch, C. (1969). A quantitative approach to biochemical structure- activity relationships. *Acc. Chem. Res.* 2, 232-239.

[19] Hansch, C., Li, R.-l., Blaney, J.M. and Langridge, R. (1982). Comparison of the inhibition of Escherichia coli and Lactobacillus casei dihydrofolate reductase by 2,4-diamino-5-(substituted-benzyl) pyrimidines: quantitative structure-activity relationships, X-ray crystallography, and computer graphics in structure-activity analysis. *J. Med. Chem.* 25, 777-784.

[20] Hansch, C., Maloney, P.P., Fujita, T. and Muir, R.M. (1962). Correlation of biological activity of phenoxyacetic acids with Hammett substituent constants and partition coefficients. *Nature* 194, 178-180.

[21] Hirst, J. D., King, R. D., and Sternberg, M. J. E. (1993a). Quantitative Structure-Activity Relationships: Neural Networks and Inductive Logic Programming Compared Against Statistical Methods. I The Inhibition of Dihydrofolate Reductase by Pyrimidines. J. Med. Chem (submitted).

[22] Hirst, J. D., King, R. D., and Sternberg, M. J. E. (1993b). Quantitative Structure- Activity Relationships: Neural Networks and Inductive Logic Programming Compared Against Statistical Methods. II The Inhibition of Dihydrofolate Reductase by Triazines. J. Med. Chem (submitted),

[23] Lavrač, N., Džeroski, S, Grobelnik, M. (1991) Learning non-recursive definitions of relations with LINUS. *EWSL '91: Machine Learning: Proceedings of the European Working Session on Learning* , Porto, Portugal (ed. Y. Kodratoff) Springer-Verlag, Lecture Notes in Artificial Intelligence.

[24] Kabsch, W. and Sander, C. (1983). How good are predictions of protein secondary structure. *FEBS Lett* 155, 179-182.

[25] King, R.D. and Sternberg, M.J.E. (1990). A machine learning approach for the prediction of protein secondary structure. *J. Mol. Biol.* 216, 441- 457.

[26] 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.

[27] King, R. D., Muggleton, S., Lewis, R. A., Srinivasan, A., Feng, C., and Sternberg, M. J. E. (1993). Drug Design Using Inductive Logic Programming. *Proceedings of the 26th Annual Hawaii International Conference on System Sciences* (1, 646-655). Los Alamitos, CA: IEEE Computer Society Press

[28] Kneller, D.G., Cohen, F.E. and Langridge, R. (1990). Improvements in protein secondary structure prediction by an enhanced neural network. *J. Mol. Biol.* 214, 171-182.

[29] Kodratoff, Y. , Langley, P. (1993, eds.) *Real-World Applications of Machine Learning* (workshop notes), Vienna, april 1993.

[30] Koile, K., Shapiro, R., Abarbanel, R. M., Webster, T. A., Lathrop, R. H., and Critchlow, R. E. (1991) Applying AI techniques to drug design: A preliminary report. *AI Approaches to Classification and Pattern Recognition in Molecular Biology* (119-125) AAAI Workshop.

[31] Martin, Y.C. (1978) *Quantitative Drug Design: A Critical Introduction.* New York: Marcel, Dekker, Inc.

[32] Matthews, D.A., Bolin, J.T., Burridge, J.M., Filman, D.J., Volz, K.W., Kaufman, B.T., Beddell, C.R., Champness, J.N., Stammers, D.K. and Kraut, J. (1985). Refined crystal structures of Escherichia coli and chicken liver dihydrofolate reductase containing bound trimethoprim. *J. Biol. Chem.* 260, 381-391.

[33] Michalski, R.S. (1983) A theory and methodology of inductive learning. In: *Machine Learning: an Artificial Intelligence Approach* (eds. R.S. Michalski, J.G. Carbonell and T.M. Mitchell) Palo Alto, CA: Tioga.

[34] Morales, E. (1992) Learning chess patterns. In *Inductive Logic Programming* (ed. S. Muggleton) London: Academic Press.

[35] 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.

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

[37] Quinlan, R.J. (1983) Learning efficient classification procedures and their application to chess end games. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchel (Eds.), *Machine Learning: An artificial intelligence approach (Vol 1).* San Mateo, CA: Morgan Kaufmann.

[38] Quinlan, J.R. (1986) Induction of decision trees. *Machine Learning Journal*, Vol. 1, 81-106.

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

[40] Quinlan, J.R. (1993) Combining instance-based and model based learning. *Proceedings of Tenth International Conference on Machine Learning.* San Mateo, CA: Morgan Kaufman.

[41] Ripley, B.D. (1992) Statistical Aspects of Neural Networks. *Seminaire Europeen de Statistique* Chapman and Hall

[42] Rost, B., and Sander, C. (1993) Prediction of Protein Secondary Structure at Better than 70% Accuracy. *J. Mol. Biol.* 232, 584-599.

[43] Roth, B., Aig, E., Rauckman, B. S., Srelitz, J. Z., Phillips, A. P., Ferone, R., Bushby, S. R. M., and Siegel, C. W. (1981). 2,4-Diamino-5-benzylpyrimidines and Analogues as Antibacterial Agents. 5. 3',5'-Dimethoxy-4'-substituted-benzyl Analogues of Trimethoprim. *J. Med. Chem.* 24, 933-941.

[44] Roth, B., Rauckman, B. S., Ferone, R., Baccanari, D. P., Champness, J. N., and Hyde, R. M. (1987). 2,4-Diamino-5-benzylpyrimidines as Antibacterial Agents. 7. Analysis of the Effect of 3,5-Dialkyl Substituent Size and Shape on Binding to Four Different Dihydrofolate Reductase Enzymes. *J. Med. Chem.* 30, 348-356.

[45] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature* 323, 533-536.

[46] Silipo, C., and Hansch, C. (1975). Correlation Analysis. Its application to the Structure-Activity Relationship of Triazines Inhibiting Dihydrofolate Reductase. *J. Am. Chem. Soc.* 97, 6849-6861.

[47] So, S.-S., and Richards, W. G. (1992). Application of Neural Networks: Quantitative Structure-Activity Relationships of the Derivatives of 2,4-Diamino-5-(substituted-benzyl)pyrimidines as DHFR Inhibitors. *J. Med. Chem.* 35, 3201-3207.

[48] Srinivasan, A., Muggleton, M., and Bain, M. (1992) Distinguishing exceptions from noise in non-monotonic learning. *Proc. Second Int. ILP Workshop on Inductive Logic Programming,* Tokyo, Japan. ICOT TM-1182.