Theories for mutagenicity: a study in first-order and feature-based induction

Ashwin Srinivasan S.H. Muggleton

Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford

M.J.E. Sternberg R.D. King

Biomolecular Modelling Laboratory, Imperial Cancer Research Fund, Lincoln's Inn Fields, London.

Abstract

A classic problem from chemistry is used to test a conjecture that in domains for which data are most naturally represented by graphs, theories constructed with Inductive Logic Programming (ILP) will significantly outperform those using simpler feature-based methods. One area that has long been associated with graph-based or structural representation and reasoning is organic chemistry. In this field, we consider the problem of predicting the mutagenic activity of small molecules: a property that is related to carcinogenicity, and an important consideration in developing less hazardous drugs. By providing an ILP system with progressively more structural information concerning the molecules, we compare the predictive power of the logical theories constructed against benchmarks set by regression, neural, and tree-based methods.

1 Introduction

Constructing theories to explain observations occupies much of the creative hours of scientists and engineers. Programs from the field of Inductive Logic Programming (ILP) [14] are being developed to assist in this activity. By using problem-specific background knowledge encoded as logic programs, these programs have constructed restricted first-order logic solutions for problems in molecular biology [12,17], stress analysis in engineering [6] and electronic circuit diagnosis [8]. One feature common to all these applications is that the data involved are most naturally represented by graphs (that is, they involve "structure"). That a technique capable of constructing relational theories *can* be used in such situations is, in itself, unsurprising. More interesting however is the following conjecture:

Preprint submitted to Elsevier Science *16 November 1995*

Conjecture. In domains that are "naturally" structural, ILP will significantly outperform simpler feature-based methods that can only use pre-selected attributes.

The intuition underlying the conjecture is that it would be impossible to anticipate all relevant structural features required for a problem. While an ILP algorithm would automatically discover such features "from first principles", a feature-based learner would be restricted to the ones given to it *ab initio*. In an area that is usually associated with structural data, we test this hypothesis by studying the theories constructed by the ILP program *Progol*, and comparing them against those constructed by three feature-based methods of data-fitting: linear regression, neural network and classification trees.

The problem we consider in this paper comes from the field of organic chemistry where the constructs used by chemists in normal discourse are clearly graph-based. These are either based on atom/bond connectivities or on Cartesian co-ordinates. In this naturally structural setting, we consider discovering rules for mutagenicity in nitroaromatic compounds. These compounds occur in automobile exhaust fumes and are also common intermediates in the synthesis of many thousands of industrial compounds [5]. Highly mutagenic nitroaromatics have been found to be carcinogenic [1]. Clearly, it is of considerable interest to the chemical industry to determine accurate methods for predicting mutagenicity. Besides directing the development of less hazardous new compounds, such methods would also have applicability in areas such as antimicrobial agents where it is not possible to determine mutagenicity using standard tests (this is because of the toxicity of the agents to test organisms). Traditionally, explanations for mutagenicity are constructed in two stages. First, regression-like models using pre-selected features (or attributes) are obtained. Second, these models are re-interpreted using basic chemical constructs to give a better understanding of the principles underlying mutagenic action. The experiments reported here seek to determine if there is any advantage in dealing directly with the actual structural representation of molecules.

Even when restricted to a particular problem, a direct comparison of algorithms can be difficult. Suppose a theory constructed by an ILP algorithm *does* perform better than one constructed by a propositional one. It could then be claimed that the propositional method performed worse because 1) the features provided were inappropriate; or 2) the background knowledge for the ILP algorithm somehow "gave" it the answer; 3) the propositional algorithm was not the best in its class or inappropriate values were provided for its parameters. Similar claims could be made, of course, if situations were reversed. For the data used here, we have taken the following measures to counter such claims:

Attributes. The attributes used for propositional learning are those that highly experienced chemists believe to be relevant. We rely on their professional judgement in this matter.

Background knowledge. To clarify the role played by background knowledge in constructing ILP theories, we commence with an extremely sparse representation consisting only of atom and bond properties of the molecules. This is then enriched, and results are compared against the propositional benchmarks at each step.

Algorithms. We have used algorithms that should be amongst the most powerful. In [15] the advantages of *Progol* over other ILP systems have been tabulated. We use standard techniques for linear regression. The learning rule used by the neural technique relies on the back-propagation of errors. Changes in weight are calculated by solving a set of differential equations as described in [9]. This technique removes the need for learning-rate or momentum parameters [18]. Finally, the procedure embodied by the CART algorithm [2] as implemented by the Ind package [3] is used to construct classification trees. The choice of propositional algorithms is supported by their good performances on a variety of general test problems (as reported in [13]) and chemical structure activity problems in particular (as reported in [10,11]). The CART procedure (as implemented by Ind) determines automatically the parameter settings that minimise the tree's estimated error-rate.

The paper is organised as follows. Section 2 clarifies the precise scope of this study. Materials available are described in Section 3, the design of experiments in Section 4, and results in Sections 5. Section 6 concludes this study.

2 The Mutagenesis Problem: Experimental Aim

Our intentions here are to investigate if an ILP algorithm can utilise the natural encoding of molecules in terms of atom and bond connectivities to significantly outperform an algorithm that is only capable of using pre-selected structural attributes.

Figure 1 shows different representations that could be of interest for comparative studies of ILP and feature-based learners. The scope of this study is the shaded box.

ILP algorithm

Feature-based algorithm		NS only	PS only	NS+PS	S only	NS+S	PS+S	NS+PS+S
	NS only							
	PS only							
	NS + PS							

Fig. 1. Potential studies comparing ILP and feature-based algorithms. The representation used by the algorithms is as follows: NS: non-structural attributes; PS: pre-coded structural attributes; S: explicit representation of structure. The study in this paper falls in the shaded square.

3 Materials

3.1 Data

We have chosen to study the mutagenicity of 230 compounds listed in [5]. The authors of [5] propose a linear regression model to predict mutagenicity. They use the following independent variables:

$logP$: log of the compound's octanol/water partition coefficient (hydrophobicity);
ϵ_{LUMO}: energy of the compounds lowest unoccupied molecular orbital. This is obtained from a quantum mechanical molecular model;
I_1: an 'indicator variable' that is set to 1 for all compounds containing 3 or more fused rings; and
I_a: an 'indicator variable' that takes the value 1 for "...five examples of acenthrylenes and shows that these are much less active than expected for some unknown reason" ([5], pp 788).

In the terminology of Figure 1 $logP$ and ϵ_{LUMO} are NS attributes, and $I_{1,a}$ are PS attributes. The latter were chosen specifically, based on chemical knowledge of mutagenicity. The authors of [5] further identify 188 compounds as being amenable to a regression analysis with these 4 attributes. The remaining 42 compounds were not used in constructing the regression model.

We confine this study to the simple task of discriminating compounds with positive log mutagenicity from those which have zero or negative log mutagenicity. Of the 230 compounds, 138 have positive levels of log mutagenicity (as reported in [5]). These are labelled "active" and constitute the source of positive examples. The remaining 92 are labelled "inactive" and constitute the source of negative examples. Figure 2 shows the distribution of compounds into these classes for the different subsets identified in [5].

Compounds	"Active"	"Inactive"	Total
"Regression friendly"	125	63	188
"Regression unfriendly"	13	29	42
All	138	92	230

Fig. 2. Class distribution of compounds.

3.2 Structural representation of molecules

We now consider an explicit structural representation of the molecules (the S component of Figure 1). This will be used as background knowledge for the ILP algorithm. A prominent study involving the analysis of drug structures with ILP was first reported by [12]. While it highlighted the advantages of logic-based learning, all drugs studied were variants of a basic template, and all that was required was substitutions into 3 positions on that template (see Figure 3). This is reflected by the fact that the rules obtained in that study were largely propositional.

In contrast, the compounds in this study are considerably more diverse and incapable of being represented by one or more templates (see Figure 4). The most primitive structural representation of molecules that is practical is in terms of the atomic and bonding properties of the molecule.

The atom and bond structures of the 230 compounds were obtained from the standard molecular modelling package QUANTA. For each compound QUANTA automatically obtains the atoms, bonds, bond types (for example, aromatic, single, double etc.), atom types (for example, aromatic carbon, aryl carbon etc.), and the partial charges on atoms. QUANTA automatically classifies bonds into one of 8 types, and atoms into one of 233 types (most of which relate to different types of carbon atoms). The output was a set of Prolog facts of the form:

bond(compound,atom1,atom2,bondtype), stating that *compound* has a bond of *bondtype* between the atoms *atom1* and *atom2*. For example, an aromatic bond between atoms *d2_1* and *d2_2* in compound *d2* is represented by QUANTA as *bond(d2,d2_1,d2_2,7).*

atm(compound,atom,element,atomtype,charge), stating that in *compound atom* has element *element* of *atomtype* and partial charge *charge*. For example,

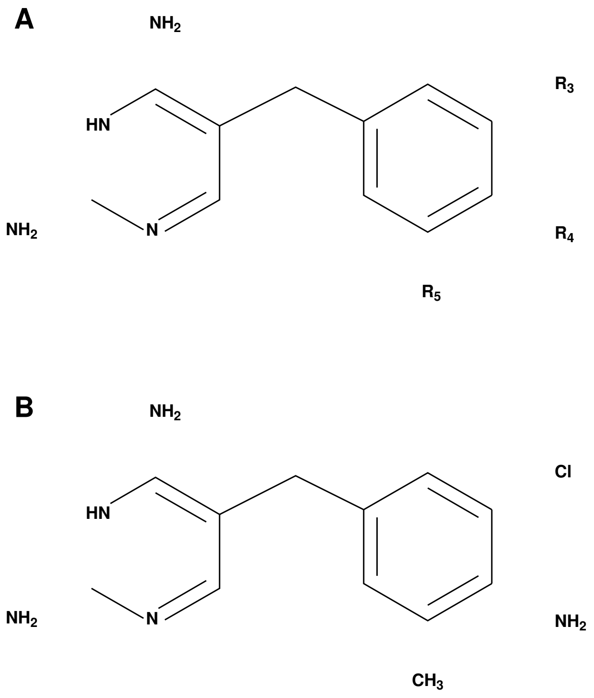


Fig. 3. Typical data format for compounds used in the analysis of drug structures with ILP. A) Template of 2,4-diamino-5(substituted-benzyl)pyrimidines R3, R4, and R5 are the three possible substitution positions. B) Example compound: $3-Cl$, $4-NH_2$, $5-CH_3$.

> QUANTA encodes the fact that atom *d2_1* in compound *d2* is an aromatic carbon atom with partial charge 0.067 by the fact *atm(d2,d2_1,c,22,0.067)*.

The resulting 12203 ground unit clauses on atomic structure and bonding generated by QUANTA form the basic building blocks for the structural representation of the 230 molecules. In this paper, we will refer to this set of clauses as $S1$, and it forms the most primitive structural description of the chemical compounds under study.

Using the atom and bond description, it is possible to define libraries of elementary chemical concepts. Appendix A does this, providing definitions of methyl groups, nitro groups, aromatic rings, hetero-aromatic rings, connected rings, ring length, and the three distinct topological ways to connect three benzene rings. The ILP algorithm used here can directly utilise the non-ground definitions in Appendix A. However, for reasons of efficiency, an equivalent ground tabulation of these definitions is used. This ground tabulation, comprised of 1433 ground unit clauses, along with the clauses in $S1$ are here termed $S2$. In turn, clauses in sets $S1$ and $S2$ (Figure 5) will be provided as background

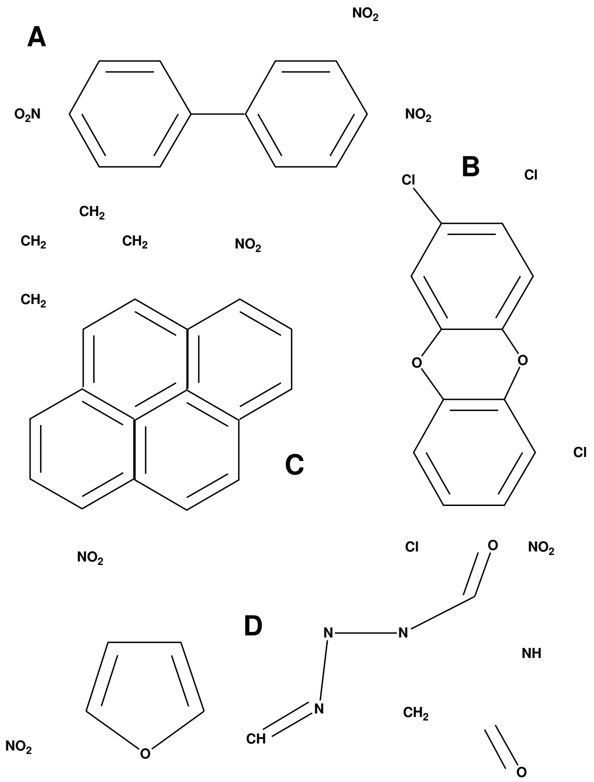


Fig. 4. Examples of the diverse set of aromatic and heteroaromatic nitro compounds used here. A) 3,4, 4'-trinitrobiphenyl B) 2-nitro-1,3,7, 8-tetrachlorodibenzo-1,4-dioxin C) 1,6-dinitro-9,10,11, 12-tetrahydrobenzo[e]pyrene D) nitrofurantoin.

Set of clauses	Structural definitions in the set of clauses
S1	atm/5, bond/4
S2	atm/5, bond/4, tabulation of definitions in Appendix A

Fig. 5. Structural definitions available for an ILP learner.

knowledge to the ILP algorithm.[1] We note here that $S2 \models S1$.

What is the difference in providing the definitions in $S2$ to an ILP learner, and the specialised structural attributes $I_{1,a}$ (as described in Section 3.1) to a feature-based learner? The definitions $S2$ state no more than generic chemical concepts that can be used as "building-blocks" to construct arbitrarily complex chemical descriptions. In contrast, the attributes $I_{1,a}$ were developed by the authors of [5] specifically for the compounds studied here.

[1] All data described in this paper, along with the Prolog definitions comprising background knowledge $S1$ and $S2$ are available by ftp access to *ftp.comlab.ox.ac.uk*. The relevant directory is *pub/Packages/ILP/Datasets/mutagenesis*.

3.3 Algorithms

In [15], the shortcomings of various ILP algorithms are discussed, along with possible methods of overcoming them using a form of generalisation known as "mode-directed" inverse resolution. This forms the basis of the *Progol* algorithm [16] that is capable of dealing efficiently with non-determinate, non-ground logic programs. We use an early Prolog implementation of *Progol*, called *P-Progol*. At the time of writing this paper S.H. Muggleton has implemented a version called *CProgol* in the C language. Details of obtaining this version can be found in [16]. *P-Progol* is available on request from Ashwin Srinivasan (electronic mail: ashwin@comlab.ox.ac.uk). The implementation includes on-line documentation that clarifies the points of difference with the C version. The theory underlying both versions is the same and is described fully in [16]. However, they have different search strategies and pruning facilities. Consequently, given the same restrictions on output language and computational resources the two versions will typically compute different answers to a given problem from the set of allowable answers. For convenience, in the rest of this paper, we shall refer to *P-Progol* as *Progol*.

The dataset of aromatic nitrocompounds used here has previously been studied using linear regression [5] and in part, by a neural network algorithm using back-propagation of errors [22]. We repeat these experiments here. [2]

The non-parametric classification tree algorithm CART [2] completes the triad of feature-based algorithms used here. The CART algorithm used was as implemented in the Ind package. [3] Following the terminology used in [13] we will refer to "the Ind implementation of CART" as *IndCART*. Results on a variety of test problems in [13] show the performance of *IndCART* to be slower, but comparable in accuracy to the commercial version of CART.

4 Method

The data have previously been split into two subsets in [5]. 188 compounds were found to be amenable to regression on the 4 attributes described earlier (Section 3.1). For the remaining 42, regression was found to give poor results. To avoid bias against linear regression, we continue to treat these as two distinct subsets.

[2] The regression was achieved using the Minitab package (Minitab Inc, Pennsylvania State University, Pa). The implementation of the back-propagation algorithm was supplied by J Hirst of the Imperial Cancer Research Fund.

[3] NASA Ames Research Centre, MS 269-2, Moffat Field, CA 94035-1000).

For each of the two sets of compounds under study, we adopt the following k-fold cross-validation design:

(i) Randomly assign the compounds in the set to k (approximately) equal partitions. Each partition will, in turn, be withheld to form a "test" set. The compounds in the other partitions will provide the "training" data for constructing theories for predicting members in the "active" class (see Section 3.1).
(ii) For each of the k training data sets:
  (a) Construct theories using linear regression, neural network, and *IndCART*, with the 4 pre-selected attributes listed in Section 3.1;
  (b) Construct a theory using *Progol*, using the two non-structural attributes in Section 3.1, and in turn, the sets $S1$ and $S2$ of structural definitions described in Section 3.2; and
  (c) Record the predictions of each theory so constructed on the data set withheld from the algorithm.
(iii) Estimate the performance of each algorithm by counting the proportion of errors of prediction.
(iv) Analyse for any significant difference in performance by the ILP algorithm. In turn, the null hypothesis here is that the proportion of examples correctly classified by the ILP algorithm is the same as that by linear regression, neural network, and *IndCART*.

For the purposes of this study, k is assigned the value 10 for the subset of 188 compounds, and 42 for the remaining 42 compounds (that is, on the latter, a leave-one-out procedure is adopted). The empirical estimates of predictive accuracy derived in Step 3 above are used to complete the table in Figure 6.

Algorithm and representation	Data set	
	188	42
Linear regression + NS + PS	...	...
Neural network + NS + PS	...	...
IndCART + NS + PS	...	...
Progol + NS + S1	...	...
Progol + NS + S2	...	...

Fig. 6. Predictive accuracy estimates to be determined. Legend: NS refers to the non-structural attributes $logP$ and ϵ_{LUMO}; PS refers to the pre-selected structural attributes $I_{1,a}$; $S_{1,2}$ are the sets of structural definitions described earlier.

A quantitative analysis for significant differences in Step 4, should account for the fact that all algorithms are tested on the same sample. The appropriate statistical test for this is the McNemar's test for changes [7]. Given a pair of algorithms being compared, the null hypothesis is that the proportions of examples correctly classified by both algorithms is the same. By "correctly classified" we mean that compounds with positive log mutagenicity in the test set are classified as active, and those with zero or negative values are classified as inactive. An quantitative assessment of the level of significance is rendered difficult because of repetitive cross-comparisons. More details on this can be found in Appendix C. Here, in the spirit of statistical corrections available for other situations, (for example, the Bonferroni adjustment), we adopt a cautious interpretation of the level of significance.

Finally, the algorithm *Progol* requires a specification of a hypothesis language, before it can construct theories. For reference, the specification statements used in this study and other details required for repeating the experiments with *Progol* are in Appendix B.

5 Experimental results and discussion

Figure 7 tabulates the estimates of predictive accuracy following the experimental method of Section 4. The method used to calculate these entries and contingency tables for evaluating significant differences in accuracy estimates are in Appendix C.

On the data set of 188 compounds, the estimated predictive accuracies for theories constructed by Regression, Neural Network, IndCART, and Progol + NS + S2 are all approximately 89%. It is instructive to examine the classification errors made by these 4 algorithms. Figure 8 is a tabulation of examples misclassified by none, 1, 2, 3 and all 4 algorithms. The figure also shows the distribution of these cases into the "active" and "inactive" decision classes.

A simple chi-squared test reveals a highly significant heterogeneity in the misclassification. That is, the 188 examples in Figure 8 are from sub-groups whose members vary in classifiability in some generic sense: some cases being more difficult than average for sufficiently competent classifiers of whatever kind (such as those considered in the figure) and easy cases being more easy for all kinds. Likening the algorithms to four guns of different make but with similar hit frequency firing at 188 distant targets, we can consider the following 3 scenarios:

(i) Inter-target variability does not exist, and any variation is due to chance. In the absence of intrinsic inter-target variation the number of cases where 4 hits, 3 hits etc. are observed will follow a binomial law (approximating to Poisson when misses are rare).
(ii) There is some inter-target variability, and some targets are generically

Algorithm	Data set	
	188	42
Linear regression + NS + PS	0.89 (0.02)	0.67 (0.07)
Neural network + NS + PS	0.89 (0.02)	0.69 (0.07)
IndCART + NS + PS	0.88 (0.02)	0.83 (0.06)
Progol + NS + S1	0.82 (0.03)[1]	0.83 (0.06)[2]
Progol + NS + S2	0.88 (0.02)	0.83 (0.06)[2]
Default class	0.66 (0.03)	0.69 (0.07)

Fig. 7. Predictive accuracy estimates of theories of mutagenicity. Estimates in the first column are from a 10-fold cross-validation, and those in the second from a leave-one-out procedure. The "Default class" algorithm is one that simply guesses majority class. Estimated standard error is shown in parentheses after each accuracy value. NS refers to the non-structural attributes $logP$ and ϵ_{LUMO}; PS refers to the pre-selected structural attributes $I_{1,a}$; $S_{1,2}$ are the subsets of structural definitions described in Figure 5. Superscripts on *Progol* results refer to the pairwise comparison in turn of *Progol* against regression, neural network and *IndCART*. With the null hypothesis that *Progol* and its adversary classify the same proportions of examples correctly, superscript 1 indicates that the probability of observing the classification obtained is ≤ 0.05 for each of regression, neural network and *IndCART*; and 2 indicates that this probability is ≤ 0.05 for regression and neural network only (but not *IndCART*).

easier to miss than others, but this variability takes the form of an increase in the size of the variance over that given by the binomial (or Poisson) curve, but with no change in general form.

(iii) There is gross inter-target variability, and the targets are drawn from 2 different sub-populations exemplifying radically different difficulty levels. Most are easy, and all guns score hits almost every time. A minority are hard, and all guns almost always miss. An extreme case of this has been termed the 'ceiling' effect, and is the subject of an interesting discussion in [4].

The tabulations in Figure 8 contradict (i). As to discriminating between (ii) and (iii), the results are indecisive. Pending more data (should they become available), most data analysts would lean towards (ii) as it departs less abruptly from (i). If (ii) were correct, new data would be expected to fill out the concavity in the third row of Figure 8, making more of a 'tail'. If (iii) were to be the case. then the frequency in the last row would build, producing a clearly bimodal distribution.

Algorithm disagreeing with oracle class	Number of examples		
	Active	Inactive	Total
0	105	48	153
1	6	6	12
2	3	2	5
3	6	3	9
4	5	4	9
			188

Fig. 8. Classification errors made by none, 1, 2, 3 and all of Linear regression, Neural network, *IndCART* and *Progol* + NS + S2 on subset of 188 "regression friendly" compounds.

What then, is to be made of the central claim being investigated here? Figure 7 suggests that a structural representation of molecules consisting only of atom and bond definitions (the column headed *Progol* + NS + $S1$) is inadequate. With this there appears little to recommend *Progol* over its propositional adversaries, despite its positive contributions on the "regression unfriendly" data (here, also seen to be "neural net unfriendly"). Matters improve with the inclusion of the elementary structural concepts described in Appendix A (the column headed *Progol* + NS + $S2$). With this, there appears to be no disadvantage in using *Progol*, and their is evidence of an advantage in some cases (for example, when data are unsuitable for regression). With this structural representation, it appears that the results support the following (weaker) claim instead:

> In domains that are naturally structural, with modest requirements of the background knowledge ILP will perform comparably to a feature-based method that can only use pre-selected attributes.

Do the definitions in Appendix A qualify under "modest" background knowledge requirements? We believe that they do, as they express little more than very elementary concepts that are generic to chemistry. In comparison, the propositional attributes hide the realities of complex assays (for $logP$), quantum mechanical calculations (for ϵ_{LUMO}), and specialised chemical expertise with mutagenicity (for $I_{1,a}$).

While experimental results point to the weaker claim above, there appear to be good reasons to recommend *Progol* over both linear regression and neural net methods. *Progol* does as well when the latter do well, and there is some evidence that it outperforms them when they fail to do so (for the subset of 42 compounds, the regression and neural net algorithms little better than one that simply guesses the default class). This is not the case however with *IndCART*: estimates of the predictive performance of *Progol*'s theories are virtually indistinguishable from *IndCART*'s on either subset of the data.

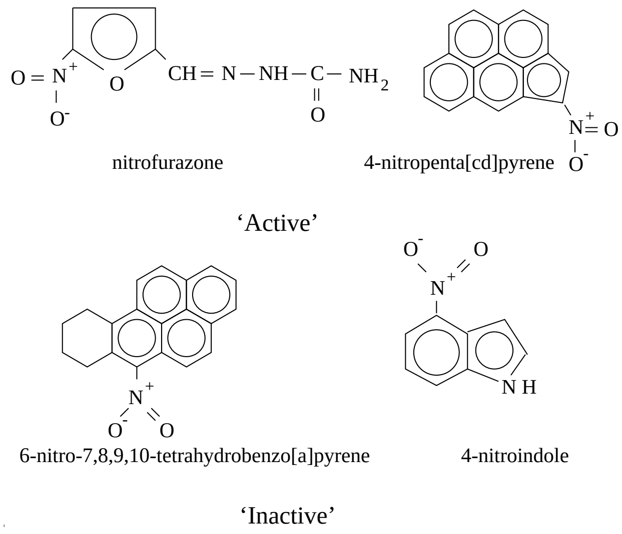
In the light of findings here, what then are the main advantages of using an ILP program in structural domains? The results show that it is possible to obtain theories with high predictive accuracy even without access to the specific chemical expertise that led to encoding the specialised structural attributes $I_{1,a}$. In [5] the authors decided on these particular attributes specifically to suit this data-set of 230 compounds. Indeed, without access to attributes $I_{1,a}$, theories from the propositional algorithms are not nearly as effective (see Figure 9). In this situation *Progol* theories are never significantly worse than that from a propositional method (even with only the structural definitions in subset $S1$).

Algorithm	Data set	
	188	42
Linear regression + NS	0.85 (0.03)	0.67 (0.07)
Neural network + NS	0.86 (0.03)	0.64 (0.07)
IndCART + NS	0.82 (0.03)	0.83 (0.06)

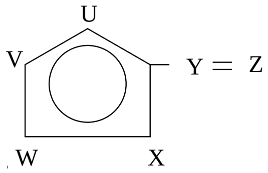
Fig. 9. Predictive accuracy estimates of theories of mutagenicity for propositional learners, without pre-selected structural features $I_{1,a}$. Accuracy estimates in the first column are from a 10-fold cross-validation, and those in the second from a leave-one-out procedure. Estimated standard error is shown in parentheses after each accuracy value. NS refers to the non-structural attributes $logP$ and ϵ_{LUMO}.

The primary edge that an ILP learner like *Progol* retains is the ability to discover concepts other than those expressed by the pre-selected attributes. This is highlighted in the results obtained on the subset of 42 compounds. For these, the *Progol* theory is a single rule that encodes the structure shown in Figure 10. This is a new structural alert for high mutagenicity in chemical compounds. The conjugated double bond should stabilise the five-membered aromatic ring, thus allowing for a greater time for the compound to diffuse to the target site, which in turn causes an increase in mutagenicity.

Further evidence is provided by a closer examination of the the pre-selected structural attribute I_a. This was devised specifically to flag five compounds in the data set. In [5] the authors state "...The very low activity of the [five] acenthrylenes is surprising in that most of the other large polycyclic aromatic compounds are reasonably well fit. This deviant group cries out for further investigation." ([5], pp 793). No such special consideration was



(A)



(B)

Fig. 10. (A) Some 'regression unfriendly' compounds and (B) the structural property found by *Progol*

needed for *Progol*. The explanation as discovered by *Progol* states that all five acenthrylenes are in a class of compounds whose ϵ_{LUMO} values are at most -1.145, and have at least one five-membered ring.

6 Conclusions

In this paper, we have looked at whether an ILP algorithm can significantly outperform feature-based algorithms in a domain that is most naturally described by relational definitions. The resulting study is an intense cross-examination of an ILP algorithm on a real-world problem: the experiments reported in [13] do much towards clarifying the comparative worth of propositional algorithms, but they were not specifically designed to test ILP techniques. The experiments here are a first step towards this goal, and the results amount to a single "data point" concerning the value of using the ILP algorithm *Progol*. Further insight stands to be gained by repeating the experimental procedure on other task domains, and by comparing the performance of *Progol* with that of other structure-based induction techniques like *FOIL* [20] and *FOCL* [19]. Some initial results towards this latter goal are available in [21].

For the problem considered here, empirical results appear to support the thesis that given moderate background knowledge, *Progol* is capable of outperforming linear regression and neural network techniques. Interestingly, taken together, the symbolic machine-learning algorithms *IndCART* and *Progol* do better than their subsymbolic counterparts. Although the results do not demonstrate any real difference in predictive accuracy in using either *Progol* or a non-parametric classification tree method, this does not detract from the role for an ILP algorithm in such domains. That an algorithm like *Progol* is able to well without access to "expert" structural features is not an insignificant achievement, and suggests that including further basic structural descriptions (for example, the Cartesian coordinates of atoms) could prove useful. Finally, the ability of an algorithm like *Progol* to discover unexpected new concepts, such as the one in Figure 10 or the explanation for the acenthrylene compounds, will continue to be an important asset in its favour, and one that is unlikely to be matched by a feature-based algorithm.

Acknowledgements

This research was supported partly by the Esprit Basic Research Action ILP (project 6020), the SERC project 'Experimental Application and Development of ILP' and an SERC Advanced Research Fellowship held by Stephen Muggleton. Stephen Muggleton is also supported by a Research Fellowship of Wolfson College Oxford. Ross King and Michael Sternberg are supported by the Imperial Cancer Research Fund. The authors are indebted to to Donald Michie for his advice and interest in this work. Thanks are also due to Rui Camacho and David Page for discussions on *Progol*, and the referees of this paper for their valuable comments.

References

[1] J. Ashby and R.W. Tennant. Definitive relationships among chemical structure, carcinogenicity and mutagenicity for 301 chemicals tested by the U.S. NTP. *Mutation Research*, 257:229–306, 1991.

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

[3] W. Buntine. Ind package of machine learning algorithms. Technical Report 244-17, Research Institute for Advanced Computer Science, NASA Ames Research Center, Moffett Field, CA 94035, 1992.

[4] P.R. Cohen. *Empirical methods for Artificial Intelligence.* MIT Press, Cambridge, MA, 1995.

[5] A.K. Debnath, R.L Lopez de Compadre, G. Debnath, A.J. Schusterman, and C. Hansch. Structure-Activity Relationship of Mutagenic Aromatic and Heteroaromatic Nitro compounds. Correlation with molecular orbital energies and hydrophobicity. *Journal of Medicinal Chemistry*, 34(2):786 – 797, 1991.

[6] B. Dolsak and S. Muggleton. The application of Inductive Logic Programming to finite element mesh design. In S. Muggleton, editor, *Inductive Logic Programming*, pages 453–472. Academic Press, London, 1992.

[7] B.S. Everitt. *The analysis of contingency tables.* Chapman and Hall, London, second edition, 1992.

[8] C. Feng. Inducing temporal fault diagnostic rules from a qualitative model. In S. Muggleton, editor, *Inductive Logic Programming*, pages 473–486. Academic Press, London, 1992.

[9] C.W. Gear. *Numerical Initial Value Problems is Ordinary Differential Equations.* Prentice-Hall, Edgewood Cliffs, NJ, 1971.

[10] J.D. Hurst, R.D. King, and M.J.E. Sternberg. Quantitative structure-activity relationships by neural networks and inductive logic programming. ii. the inhibition of dihydrofolate reductase by pyrimidines. *Journal of Computer-Aided Molecular Design*, 8:421–432, 1994.

[11] J.D. Hurst, R.D. King, and M.J.E. Sternberg. Quantitative structure-activity relationships by neural networks and inductive logic programming. ii. the inhibition of dihydrofolate reductase by triazines. *Journal of Computer-Aided Molecular Design*, 8:421–432, 1994.

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

[13] D. Michie, D.J. Spiegelhalter, and C.C. Taylor, editors. *Machine Learning, Neural and Statistical classification.* Ellis-Horwood, New York, 1994.

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

[15] S. Muggleton. Inductive logic programming: derivations, successes and shortcomings. *SIGART Bulletin*, 5(1):5–11, 1994.

[16] S. Muggleton. Inverse Entailment and Progol. *New Gen. Comput.*, 1995. to appear.

[17] S. Muggleton, R. King, and M. Sternberg. Predicting protein secondary structure using inductive logic programming. *Protein Engineering*, 5:647–657, 1992.

[18] A. J. Owens and D. L. Filkin. Efficient training of the back-propagation network by solving a system of stiff ordinary differential equations. In *Proceedings IEEE/INNS International Joint Conference of Neural Networks*, pages 381–386, Washington DC, 1989.

[19] M. Pazzani and D. Kibler. The role of prior knowledge in inductive learning. *Machine Learning*, 9:54–97, 1992.

[20] J.R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[21] A. Srinivasan, S.H. Muggleton, and R.D. King. Comparing the use of background knowledge by inductive logic programming systems. Technical Report PRG-TR-9-95, Oxford University Computing Laboratory, Oxford, 1995.

[22] D. Villemin, D. Cherqaoui, and J.M. Cense. Neural network studies: quantitative structure-activity relationship of mutagenic aromatic nitro compounds. *J. Chim. Phys*, 90:1505–1519, 1993.

A Some elementary chemical concepts defined in terms of the atom and bond structure of molecules

The following are Prolog definitions for some simple chemical concepts that can be defined directly using the atomic and bond structure of a molecule.

```
% In the following QUANTA bond type 7 is aromatic.

% Three benzene rings connected linearly
anthracene(Drug,[Ring1,Ring2,Ring3]) :-
        benzene(Drug,Ring1),
        benzene(Drug,Ring2),
        Ring1 @> Ring2,
        interjoin(Ring1,Ring2,Join1),
        benzene(Drug,Ring3),
        Ring1 @> Ring3,
        Ring2 @> Ring3,
        interjoin(Ring2,Ring3,Join2),
        \+ interjoin(Join1,Join2,_),
        \+ members_bonded(Drug,Join1,Join2).

% Three benzene rings connected in a curve
phenanthrene(Drug,[Ring1,Ring2,Ring3]) :-
        benzene(Drug,Ring1),
        benzene(Drug,Ring2),
        Ring1 @> Ring2,
        interjoin(Ring1,Ring2,Join1),
```

```
        benzene(Drug,Ring3),
        Ring1 @> Ring3,
        Ring2 @> Ring3,
        interjoin(Ring2,Ring3,Join2),
        \+ interjoin(Join1,Join2,_),
        members_bonded(Drug,Join1,Join2).


% Three benzene rings connected in a ball
ball3(Drug,[Ring1,Ring2,Ring3]) :-
        benzene(Drug,Ring1),
        benzene(Drug,Ring2),
        Ring1 @> Ring2,
        interjoin(Ring1,Ring2,Join1),
        benzene(Drug,Ring3),
        Ring1 @> Ring3,
        Ring2 @> Ring3,
        interjoin(Ring2,Ring3,Join2),
        interjoin(Join1,Join2,_).

members_bonded(Drug,Join1,Join2) :-
        member(J1,Join1),
        member(J2,Join2),
        bondd(Drug,J1,J2,7).


ring_size_6(Drug,Ring_list) :-
        atoms(Drug,6,Atom_list,_),
        ring6(Drug,Atom_list,Ring_list,_).

ring_size_5(Drug,Ring_list) :-
        atoms(Drug,5,Atom_list,_),
        ring5(Drug,Atom_list,Ring_list,_).

% benzene - 6 membered carbon aromatic ring
benzene(Drug,Ring_list) :-
        atoms(Drug,6,Atom_list,[c,c,c,c,c,c]),
        ring6(Drug,Atom_list,Ring_list,[7,7,7,7,7,7]).

carbon_5_aromatic_ring(Drug,Ring_list) :-
        atoms(Drug,5,Atom_list,[c,c,c,c,c]),
        ring5(Drug,Atom_list,Ring_list,[7,7,7,7,7]).

carbon_6_ring(Drug,Ring_list) :-
        atoms(Drug,6,Atom_list,[c,c,c,c,c,c]),
        ring6(Drug,Atom_list,Ring_list,Bond_list),
        Bond_list \== [7,7,7,7,7,7].
```

```
carbon_5_ring(Drug,Ring_list) :-
        atoms(Drug,5,Atom_list,[c,c,c,c,c]),
        ring5(Drug,Atom_list,Ring_list,Bond_list),
        Bond_list \== [7,7,7,7,7].

hetero_aromatic_6_ring(Drug,Ring_list) :-
        atoms(Drug,6,Atom_list,Type_list),
        Type_list \== [c,c,c,c,c,c],
        ring6(Drug,Atom_list,Ring_list,[7,7,7,7,7,7]).

hetero_aromatic_5_ring(Drug,Ring_list) :-
        atoms(Drug,5,Atom_list,Type_list),
        Type_list \== [c,c,c,c,c],
        ring5(Drug,Atom_list,Ring_list,[7,7,7,7,7]).

atoms(Drug,1,[Atom],[T]) :-
        atm(Drug,Atom,T,_,_),
        T \== h.
atoms(Drug,N1,[Atom1|[Atom2|List_a]],[T1|[T2|List_t]]) :-
        N1 > 1,
        N2 is N1 - 1,
        atoms(Drug,N2,[Atom2|List_a],[T2|List_t]),
        atm(Drug,Atom1,T1,_,_),
        Atom1 @> Atom2,
        T1 \== h.

ring6(Drug,[Atom1|List],[Atom1,Atom2,Atom4,Atom6,Atom5,Atom3],
        [Type1,Type2,Type3,Type4,Type5,Type6]) :-
        bondd(Drug,Atom1,Atom2,Type1),
        memberchk(Atom2,[Atom1|List]),
        bondd(Drug,Atom1,Atom3,Type2),
        memberchk(Atom3,[Atom1|List]),
        Atom3 @> Atom2,
        bondd(Drug,Atom2,Atom4,Type3),
        Atom4 \== Atom1,
        memberchk(Atom4,[Atom1|List]),
        bondd(Drug,Atom3,Atom5,Type4),
        Atom5 \== Atom1,
        memberchk(Atom5,[Atom1|List]),
        bondd(Drug,Atom4,Atom6,Type5),
        Atom6 \== Atom2,
        memberchk(Atom6,[Atom1|List]),
        bondd(Drug,Atom5,Atom6,Type6),
        Atom6 \== Atom3.
```

```
ring5(Drug,[Atom1|List],[Atom1,Atom2,Atom4,Atom5,Atom3],
        [Type1,Type2,Type3,Type4,Type5]) :-
        bondd(Drug,Atom1,Atom2,Type1),
        memberchk(Atom2,[Atom1|List]),
        bondd(Drug,Atom1,Atom3,Type2),
        memberchk(Atom3,[Atom1|List]),
        Atom3 @> Atom2,
        bondd(Drug,Atom2,Atom4,Type3),
        Atom4 \== Atom1,
        memberchk(Atom4,[Atom1|List]),
        bondd(Drug,Atom3,Atom5,Type4),
        Atom5 \== Atom1,
        memberchk(Atom5,[Atom1|List]),
        bondd(Drug,Atom4,Atom5,Type5),
        Atom5 \== Atom2.

nitro(Drug,[Atom0,Atom1,Atom2,Atom3]) :-
        atm(Drug,Atom1,n,38,_),
        bondd(Drug,Atom0,Atom1,1),
        bondd(Drug,Atom1,Atom2,2),
        atm(Drug,Atom2,o,40,_),
        bondd(Drug,Atom1,Atom3,2),
        Atom3 @> Atom2,
        atm(Drug,Atom3,o,40,_).

methyl(Drug,[Atom0,Atom1,Atom2,Atom3,Atom4]) :-
        atm(Drug,Atom1,c,10,_),
        bondd(Drug,Atom0,Atom1,1),
        atm(Drug,Atom0,Type,_,_),
        Type \== h,
        bondd(Drug,Atom1,Atom2,1),
        atm(Drug,Atom2,h,3,_),
        bondd(Drug,Atom1,Atom3,1),
        Atom3 @> Atom2,
        atm(Drug,Atom3,h,3,_),
        bondd(Drug,Atom1,Atom4,1),
        Atom4 @> Atom3,
        atm(Drug,Atom4,h,3,_).


% intersection(+Set1, +Set2, ?Intersection)

interjoin(A,B,C) :-
        intersection(A,B,C),
        C \== [].

bondd(Drug,Atom1,Atom2,Type) :-  bond(Drug,Atom2,Atom1,Type).
```

```
member(X,[X|_]).
member(X,[_|T]):-
        member(X,T).

connected(Ring1,Ring2):-
        Ring1 \= Ring2,
        member(Atom,Ring1),
        member(Atom,Ring2), !.
```

B Using *Progol*

B.1 Defining a hypothesis language for Progol

The language $\mathcal{L}$ is defined in terms of

- Mode declarations which state the 'forms' that atoms in hypothesis can take in terms of
  - the places where variables are allowed an whether they are inputs or outputs (indicated by + or −);
  - the places where constants are allowed (indicated by #);
  - the types of these variables and constants; and
  - the degree of indeterminacy when making such a call to the background knowledge. This is either a number or $*$ meaning finite but unbounded recall of the goal.
- the maximum number of layers of variables introduced by atoms in the body of the clause from variables in the head of the clause;
- the acceptable level of consistency in terms of the maximum number of negatives that can be covered by any clause; and
- the maximum cardinality of any clause.

For the mutagenesis problem, the hypothesis language $\mathcal{L}$ for *Progol* is defined in Figure B.1.

In addition to the structural definitions, *Progol* was also provided the following information as background knowledge. These concern type-definitions and definitions for inequalities.

```
% compounds are named by an alphabet followed by a number
compound(D):-
        name(D,[_|X]), name(Num,X), int(Num),
```

Mode declarations	mode(∗,bond(+compound,−atomid,−atomid,#integer))
	mode(∗,bond(+compound,−atomid,+atomid,#integer))
	mode(∗,bond(+compound,+atomid,−atomid,#integer))
	mode(∗,bond(+compound,+atomid,+atomid,#integer))
	mode(∗,atm(∗,+compound,+atomid,#element,#integer,-charge))
	mode(∗,atm(∗,+compound,−atomid,#element,#integer,-charge))
	mode(1,(+charge)=(#charge))
	mode(1,lumo(+compound,−energy))
	mode(1,logp(+compound,−hydrophob))
	mode(1,gteq(+charge,#real))
	mode(1,gteq(+energy,#real))
	mode(1,gteq(+hydrophob,#real))
	mode(1,lteq(+charge,#real))
	mode(1,lteq(+energy,#real))
	mode(1,lteq(+hydrophob,#real))
Depth of variables	2
Maximum negatives	5
Maximum literals	4

Fig. B.1. Language specification for the mutagenesis problem.

```
        Num >= 1, Num =< 230, !.

% atoms are identified by the compound name, followed by an "_",
% followed by a unique number
atomid(A):-
        name(A,[_|X]),
        append(Z,[95|Y],X),
        name(N1,Y),
        name(N2,Z),
        int(N1), int(N2),
        N2 >= 1, N2 =< 230,
        N1 =< 500, !.

append([],A,A).
append([H|T],A,[H|T1]):-
        append(T,A,T1).

% charge, epsilon lumo and log P are all floating point numbers.
```

```
charge(X):-
        number(X).
energy(X):-
        number(X).
hydrophob(X):-
        number(X).


% chemical elements comprising the atoms
element(br).  element(c).  element(cl).  element(f).
element(h).  element(i).  element(n).  element(o).  element(s).

% inequality definitions
gteq(X,Y):-
        not(var(X)), not(var(Y)), !,
        number(X), number(Y),
        X >= Y.
gteq(X,X):-
        not(var(X)),
        number(X).

lteq(X,Y):-
        not(var(X)), not(var(Y)), !,
        number(X), number(Y),
        X =< Y.
lteq(X,X):-
        not(var(X)),
        number(X).
```

C Methods of contingency tables analysis used for results

For reference, we first provide a brief description of the analysis tools used. Readers familiar with the calculations may wish to skip these details. For others, more information can be found in any standard statistical textbook on analysing contingency tables (for example, [7]).

Accuracy estimates in Figure 7 are obtained as follows. For each algorithm, a 2×2 tabulation of actual values against those predicted by an algorithm is obtained as in Figure C.1.

Before estimating the predictive accuracy, the first question to answer is whether there is any association between actual and predicted values. This is adequately catered for by obtaining the χ^2 value for the table. The relevant

		Actual: Active	Actual: Inactive	
Predicted	Active	n_1 (e_1)	n_2 (e_2)	n_a
	Inactive	n_3 (e_3)	n_4 (e_4)	n_b
		n_c	n_d	N

Fig. C.1. Tabulating the performance of an algorithm. n_1 is the number of compounds known to be active, and predicted as such. Similarly for entries $n_{2,3,4}$. e_1 is the expected value of for the Active/Active cell, under the hypothesis that the actual class is independent of the predicted one. It is calculated as $e_1 = (n_a n_c)/N$. Similarly for $e_{2,3,4}$.

formula for this is as follows:

$$\chi^2 = \sum_{i=1}^{4} \frac{(n_i - e_i)^2}{e_i}$$

A routine correction for non-continuous numbers (referred to as Yates' correction) is introduced by replacing the numbers $n_{1,\ldots,4}$ by numbers which are 0.5 of a unit less when they exceed the expected value, and 0.5 of a unit more when they lie below the expected value. The χ^2 probability is obtained from standard tables with $\nu = 1$ degree of freedom.

A high χ^2 value indicates a low probability of a chance association between predicted and actual values. In this study, we have stipulated that this probability is no more than 0.05. That is, χ^2 values must be at least 3.84. The predictive accuracy for theories that satisfy this constraint is then estimated as $p = (n_1 + n_4)/N$. The error in this estimate is $\pm\sqrt{pq/N}$ where $q = 1 - p$. For brevity, we do not reproduce the contingency tables for obtaining the accuracy and error estimates. These are available on request from Ashwin Srinivasan (electronic mail: ashwin@comlab.ox.ac.uk). Here we only tabulate the χ^2 values.

McNemar's test for changes is used to obtain the significance results in Figure 7. For a pair of algorithms, this is done by a cross-comparison of the compounds correctly and incorrectly classified as shown in Figure C.3.

The null hypothesis is that the proportions of examples correctly classified by both algorithms is the same. If there is no significant difference in the performance of the two algorithms, half of the $n_2 + n_3$ cases whose classifications disagree should be classified correctly by A_1 and A_2 respectively. Because of

Algorithm	Data set	
	188	42
Linear regression + NS + PS	106.2	0.2
Neural network + NS + PS	107.7	2.4
IndCART + NS + PS	102.1	11.7
Progol + NS + S1	66.7	12.1
Progol + NS + S2	94.1	12.1
Default class	0	0

Fig. C.2. χ^2 values for theories

		Predicted (A_1)		
		Correct	Incorrect	
Predicted (A_2)	Correct	n_1	n_2	n_a
	Incorrect	n_3	n_4	n_b
		n_c	n_d	N

Fig. C.3. Cross-comparison of the predictions of a pair of algorithms $A_{1,2}$ n_1 is the number of compounds whose class is correctly predicted by both algorithms. Similarly for the entries $n_{2,3,4}$.

small numbers, we directly estimate the probability of a chance classification using the binomial distribution, with probability of success at 0.5. In effect, this is likened to probability of obtaining at least n_2 (or n_3, if greater) heads in a sequence of $n_2 + n_3$ tosses of a fair coin.

McNemar's test calculations

These calculations refer to tabulations in Figure 7. The analysis here calls for a repeated cross-comparison of *Progol*, with different amounts of background knowledge, against 3 other feature-based learners. For reasons of space, we report only those numbers that are relevant to the procedure described in the preceding paragraphs (the numbers n_2 and n_3 in Figure C.3). We further adopt the convention of using n_2 to denote the situation where *Progol* classifies an example correctly and a feature-based learner classifies the same example incorrectly. Conversely n_3 will be used to denote the scenario that *Progol* misclassifies an example and the feature-based learner does not.

It is evident that repeated cross-comparisons of this form will yield occasions when *Progol*'s performance will apparently seem better than its propositional adversary. For repeated comparisons of a given pair of algorithms on different random samples of data, it is possible to apply a correction (known as the Bonferroni adjustment) for this problem. The situation of repeated comparisons of different pairs of algorithms on a given set of data (as is here) does not, on the surface, appear to be amenable to the same correction. However, adopting the spirit of the correction, we shall refrain from any quantitative interpretation of the binomial probabilities (P) obtained.

Feature-based learner	Progol					
	NS + S1			NS + S2		
	n_2	n_3	P	n_2	n_3	P
Regression	2	15	0.001	9	6	0.304
Neural net	6	19	0.007	9	12	0.332
IndCART	6	17	0.017	10	11	0.500

Fig. C.4. Cross-comparisons on “regression friendly” compounds.

Feature-based learner	Progol					
	NS + S1			NS + S2		
	n_2	n_3	P	n_2	n_3	P
Regression	7	0	0.008	7	0	0.008
Neural net	7	1	0.017	7	1	0.017
IndCART	1	1	0.750	1	1	0.750

Fig. C.5. Cross-comparisons on “regression unfriendly” compounds.