Theory Recovery

Rupert Parson[1], Khalid Khan[1], and Stephen Muggleton[2]

[1] Oxford University Computing Laboratory, UK
{rupert.parson,khalid.khan}@comlab.ox.ac.uk
[2] Department of Computer Science, University of York, UK
stephen@cs.york.ac.uk

Abstract. In this paper we examine the problem of repairing incomplete background knowledge using *Theory Recovery*. Repeat Learning under ILP considers the problem of updating background knowledge in order to progressively increase the performance of an ILP algorithm as it tackles a sequence of related learning problems. Theory recovery is suggested as a suitable mechanism. A bound is derived for the performance of theory recovery in terms of the information content of the missing predicate definitions. Experiments are described that use the *logical back-propagation* ability of Progol 5.0 to perform theory recovery. The experimental results are consistent with the derived bound.

1 Introduction

In a previous paper [1], the authors described an extension of the standard machine learning framework, called *repeat learning*. In this framework, the learner is not trying to learn a single concept, but a series of related concepts, all drawn independently from the same distribution $\mathcal{D}$. A finite sequence of examples is provided for each concept in the series. The learner does not initially know $\mathcal{D}$, but progressively updates a posterior estimation of $\mathcal{D}$ as the series progresses.

Under Inductive Logic Programming (ILP) [6], the learner's estimation of $\mathcal{D}$ depends on the linguistic bias conveyed by his hypothesis language. The ILP learner can therefore alter the estimation of $\mathcal{D}$ by making changes to the hypothesis language. The previous paper [1] discussed a mechanism for this process that adjusted the background knowledge using predicate invention.

One can quantify the expected performance of an ILP algorithm by bounding the expected error of a hypothesis formed given the number of examples seen. Previous bounds for Progol [3, 4] have only considered the situation in which the learner knows the distribution $\mathcal{D}$. In this paper, we construct a bound for the case when the learner's estimate is incorrect. Significantly, this bound describes the difference between the estimate and the true distribution $\mathcal{D}$ in terms of the missing information content in the hypothesis language used by the learner.

Theory recovery is the process of adjusting, or completing background knowledge. In theory recovery, an incomplete logic program is reconstructed on the basis of examples. The examples indicate the desired behaviour of a particular predicate in the program, defined in terms of the background knowledge. A new

version of the ILP algorithm Progol, version 5.0, uses *logical back-propagation* to perform theory recovery.

The paper is structured as follows. Section 2 describes the formulation of the new expected error bound, with its proof given in Appendix C. Section 3 describes experiments that give results that are consistent with this bound. A complete set of results is given in Appendix A. Finally, our conclusions are drawn in Section 4. Appendix B briefly describes the mechanics of logical back-propagation in Progol 5.0.

2 A Theory Recovery Error Bound

It was shown in [5] that under suitable assumptions the class of all polynomial time-bounded logic programs is (U-)learnable. In [4] explicit upper bounds were given for the error of a Progol-like learning algorithm. The paper considered the case of positive-only learning compared to the more traditional positive and negative setting. In both cases upper bounds on expected error were derived showing that learning could be efficiently achieved. However under both models a strong assumption was made: that the learner knows the prior distribution (used by the teacher) over hypotheses. Clearly there is no reason why this should in general be true. In particular, in the following result we assume that the background knowledge of the learner is missing some predicate(s) contained in the target concept. In Subsection 2.1 we review the average case Bayesian model of learning used in [4] to analyse the expected error of a Progol-like learner. In Subsection 2.2 we modify the model by assuming that the learner is not in command of the correct prior distribution and derive adjusted upper bounds.

2.1 Known Prior

The following is a version of the U-learnability framework presented in [5] and restated in [4].

The Model X is taken to be a countable class of instances and $\mathcal{H} \subseteq 2^X$ to be a countable class of concepts. D_X and $D_{\mathcal{H}}$ are probability distributions over X and $\mathcal{H}$ respectively. The teacher randomly chooses a target theory T from $D_{\mathcal{H}}$ then randomly and independently chooses a series of examples $E = \langle x_1, \ldots, x_m \rangle$ from T according to D_X.

Given E, $D_{\mathcal{H}}$ and D_X a learner L outputs a hypothesis $H \in \mathcal{H}$. The error of the hypothesis is measured as $D_X(H \setminus T) + D_X(T \setminus H)$.

The hypotheses in $\mathcal{H}$ are assumed to be ordered according to decreasing prior probability as $H_1, H_2, \ldots$. The distribution $D_{\mathcal{H}}(H_i) = \frac{a}{i^2}$ is assumed, where $a = 6/\pi^2$ is a normalising constant. This is similar to the prior probability assumptions used in Progol 4.1 [3]. This distribution is a smoothed version of

the *universal distribution*[1] which assigns equal probability to the 2^b hypotheses describable in b bits, where the sum of the probabilities of such hypotheses is 2^{-b}.

Expected Error The following theorem (stated and proved in [4]) gives an upper bound on the expected error of an algorithm which learns by maximising the Bayes' posterior probability over the initial am hypotheses within the space.

Theorem 1. *Let X be a countable instance space and $\mathcal{H} \subseteq 2^X$ be a countable hypothesis space containing at least all finite subsets of X. Let $D_{\mathcal{H}}$, D_X be probability distributions over $\mathcal{H}$ and X. Assume that $\mathcal{H}$ has an ordering $H_1, H_2, \ldots$ such that $D_{\mathcal{H}}(H_i) \geq D_{\mathcal{H}}(H_j)$ for all $j > i$. Let $D_{\mathcal{H}}(H_i) = \frac{a}{i^2}$ where $\frac{1}{a} = \sum_{i=1}^{\infty} \frac{1}{i^2} = \frac{\pi^2}{6}$. Let $\mathcal{H}_n = \{H_i : H_i \in \mathcal{H} \text{ and } i \leq n\}$. T is chosen randomly from $D_{\mathcal{H}}$. Let $ex(x, H) = \langle x, v \rangle$ where $v = True$ if $x \in H$ and $v = False$ otherwise. Let $E = \langle ex(x_1, T), \ldots, ex(x_m, T) \rangle$ where each x_i is chosen randomly and independently from D_X. $H_E = \{x : \langle x, True \rangle \in E\}$. Hypothesis H is said to be consistent with E if and only if $x_i \in H$ for each $\langle x_i, True \rangle$ in E and $x_j \notin H$ for each $\langle x_j, False \rangle$ in E. Let $n = am$. L is the following learning algorithm. If there are no hypotheses $H \in \mathcal{H}_n$ consistent with E then $L(E) = H_E$. Otherwise $L(E) = H_n(E) = H$ only if $H \in \mathcal{H}_n$, H consistent with E and for all $H' \in \mathcal{H}_n$ consistent with E it is the case that $D_{\mathcal{H}}(H) \geq D_{\mathcal{H}}(H')$. The error of a hypothesis H is defined as $Error(H, T) = D_X(T \setminus H) + D_X(H \setminus T)$. The expected error of L after m examples, $EE(m)$, is at most:*

$$\frac{1.51 + 2 \ln m}{m} \tag{1}$$

2.2 Unknown Prior

We consider an extension of the above model. Previously it was assumed that the learner knew precisely the distribution $D_{\mathcal{H}}$ from which the target concepts were drawn. Clearly there is no reason why this should hold in practical machine learning situations. We now relax this assumption and consider what happens to the expected error of learning when the learner does not know the exact distribution $D_{\mathcal{H}}$. In particular we consider an incorrect prior over hypotheses induced by incomplete background knowledge.

The Modified Model We assume the existence of a universal linguistic bias generator G that, given a target space $\mathcal{H}$, and an hypothesis language B for it, returns a probability distribution $D_{\mathcal{H}} = G(\mathcal{H}, B)$ over the target space. Occam's razor can be taken as an example of such a generator, that always gives a distribution that assigns a higher probability to hypotheses that can be expressed more simply in the hypothesis language.

[1] If we take the universal distribution to be $u(\mathcal{H}_n) = 2^{-2\lfloor \log_2(n+1) \rfloor}$ then the probability of the 2^nth hypothesis is $u(\mathcal{H}_{2^n}) = 2^{-2n} = \frac{1}{(2^n)^2}$. So $\frac{a}{i^2}$ is a smoothed and renormalised version of u.

The teacher selects the target concept from $\mathcal{H}$ according to the distribution $D_{\mathcal{T}}$. The learner's imperfect hypothesis language $B_{\mathcal{L}}$ induces an incorrect probability distribution $D_{\mathcal{L}} = G(\mathcal{H}, B_{\mathcal{L}})$. We assume the existence of some set of predicate definitions P such that $G(\mathcal{H}, (B_{\mathcal{L}} \cup P)) = D_{\mathcal{T}}$. In other words a set of "missing" predicates, that, if added to the background knowledge of the learning algorithm, would mean that the the learner's induced distribution $D_{\mathcal{L}}$ would be the correct one.

The hypotheses in $\mathcal{H}$ are ordered by the teacher according to decreasing prior probability $D_{\mathcal{T}}(H_i) = \frac{a}{i^2}$ as $H_1, H_2, \ldots, H_i, \ldots$. The learner only has partial information about this ordering in that its prior is a corrupted version of the teachers distribution. In particular, there is a set of hypotheses $\mathcal{H}_P \subseteq \mathcal{H}$ for which $H \in \mathcal{H}_P \Leftrightarrow D_{\mathcal{T}}(H) < D_{\mathcal{L}}(H)$. Let the information content in bits of an hypothesis relative to a distribution D be given by $\text{info}(D, H) = -\log_2(D(H))$. The information content of $H \in \mathcal{H}_P$ under the learner's distribution $D_{\mathcal{L}}$ is more (in bits) than the information that would be assigned under the teacher's distribution $D_{\mathcal{T}}$.

Lemma 1. *$H \in \mathcal{H}_P$ is given different indices H_i and H_j under the orderings induced by $D_{\mathcal{T}}$ and $D_{\mathcal{L}}$. If the information content of the "missing" predicates in P is at most k bits, then for any hypothesis $H \in \mathcal{H}$ the indices i and j satisfy $j \leq 2^{k/2}\, i$.*

Proof.

$$\begin{aligned} k &\geq \text{info}_{D_{\mathcal{L}}}(H) - \text{info}_{D_{\mathcal{T}}}(H) = -\log_2 D_{\mathcal{L}}(H) + \log_2 D_{\mathcal{T}}(H) \\ &= -\log_2 \frac{a}{j^2} + \log_2 \frac{a}{i^2} = 2\log_2 j - 2\log_2 i = 2\log_2 \frac{j}{i} \end{aligned}$$

Therefore $j \leq 2^{k/2}\, i$.

Expected Error

Theorem 2. *Let X be a countable instance space and $\mathcal{H} \subseteq 2^X$ be a countable hypothesis space containing at least all finite subsets of X. Assume the existence of a universal linguistic bias generator G that, given a target space $\mathcal{H}$, and an hypothesis language B for it, returns a probability distribution $D_{\mathcal{H}} = G(\mathcal{H}, B)$ over the target space. Let D_X be a probability distribution over X. Let $D_{\mathcal{T}}$, $D_{\mathcal{L}}$ be probability distributions over $\mathcal{H}$, where $D_{\mathcal{L}} = G(\mathcal{H}, B_{\mathcal{L}})$ is the probability distribution induced by the learners hypothesis language. Assume the existence of some set of predicate definitions P such that $G(\mathcal{H}, (B_{\mathcal{L}} \cup P)) = D_{\mathcal{T}}$. Let the information content of the predicate definitions P be at most k bits. Assume that $\mathcal{H}$ has an ordering $H_1, H_2, \ldots, H_i, \ldots$ such that $D_{\mathcal{T}}(H_i) \geq D_{\mathcal{T}}(H_{i+1})$ for all i and an ordering $H'_1, H'_2, \ldots, H'_j, \ldots$ such that $D_{\mathcal{L}}(H'_j) \geq D_{\mathcal{L}}(H'_{j+1})$ for all j. Let $D_{\mathcal{T}}(H_i) = \frac{a}{i^2}$ where $\frac{1}{a} = \sum_{i=1}^{\infty} \frac{1}{i^2} = \pi^2/6$. Let $\mathcal{H}_n = \{H_i : H_i \in \mathcal{H} \text{ and } i \leq n\}$. T is chosen randomly from $D_{\mathcal{T}}$. Let $ex(x, H) = \langle x, v \rangle$ where $v = True$ if $x \in H$ and $v = False$ otherwise. Let $E = \langle ex(x_1, T), \ldots, ex(x_m, T) \rangle$ where each x_i is chosen randomly and independently from D_X. $H_E = \{x : \langle x, True \rangle \in E\}$. Hypothesis H*

is said to be consistent with E if and only if $x_i \in H$ for each $\langle x_i, True\rangle$ in E and $x_j \notin H$ for each $\langle x_j, False\rangle$ in E. Let $n = am$. L is the following learning algorithm. If there are no hypotheses $H \in \mathcal{H}_n$ consistent with E then $L(E) = H_E$. Otherwise $L(E) = H_n(E) = H$ only if $H \in \mathcal{H}_n$, H consistent with E and for all $H' \in \mathcal{H}_n$ consistent with E it is the case that $D_{\mathcal{L}}(H) \geq D_{\mathcal{L}}(H')$. The error of an hypothesis H is defined as $Error(H,T) = D_X(T \setminus H) + D_X(H \setminus T)$. The expected error of L after m examples, $EE(m)$, is at most:

$$\frac{1.51 + 2\ln m + k\ln 2}{m} \tag{2}$$

Proof. Given in Appendix C

3 Experiments

To confirm the assumptions given to derive the bound given in Equation 2, the following experiments were devised and run. The experiments made use of the *logical back-propagation* abilities of Progol 5.0. This ILP algorithm uses an augmented version of Inverse entailment that includes the completion of background knowledge in the generalisation process. The mechanism is described in Appendix B.

The aim of the experiments were to determine how the coverage accuracy of the logic program would be affected by having a percentage of the clauses of the program removed, and then using Progol 5.0 to repair the program, given a varying number of examples.

3.1 The Experimental Domain

The experiments are conducted in an artificial domain, called the *base-n-string* domain. The elements of the domain are strings in base n, where $n \in \{2,3,4,5\}$, of length up to a maximum value l.

The target program has a distinct predicate for each distinct length of string. The predicate that defines strings of length m is defined in terms of the predicate that defines strings of length $m-1$. This means that a missing clause definition for a predicate defining strings of length m will affect the definition of all strings of length greater than m.

Table 1.1 shows the program for the binary case - that is, $n = 2$ where $l = 10$.

The maximum string length l was determined by restricting the total number of clauses in the logic program to be 20. For the cases $n = 2, 4, 5$, this gave values $l = 10, 5, 4$ respectively. For the case $n = 3$, the value $l = 7$ was chosen, and then one of the definitions for `p7/1` was excluded from the program.

3.2 Method

In order to be able to encapsulate the entire success set of the logic program, a *meta*-predicate `ss/1` was defined. For every predicate `pred(X)` defined, the

```
p1(zero).
p1(one).
p2(zero(A)) :- p1(A).
p2(one(A)) :- p1(A).
. . .
p10(zero(A)) :- p9(A).
p10(one(A)) :- p9(A).
```

Table 1.1: The program for base $n = 2$, $l = 10$.

Base n	l	Clause information
2	10	7.64
3	7	7.20
4	5	6.64
5	4	6.32

Table 1.2: Information content (in bits, to 2 d.p.) of a single clause in each base.

clause `ss(pred(X)) :- pred(X).` was added to the program. This meant that calling `ss/1` would then return all the ground facts provable in the original logic program.

In the learning sessions under Progol 5.0, `ss/1` was the target to be learned, and the incomplete logic program is given as background knowledge for `ss/1`. The examples given were of the form `ss(fact)` where `fact` is a ground fact that should be provable by the original complete logic program. Logical back-propagation in Progol 5.0 uses these examples to complete any missing predicates in the background knowledge. Notice that only positive examples were used.

A complete program can be used to generate the set of all base-n-strings up to a certain length. An incomplete or partially repaired program will generate only a subset of these strings. Therefore at each stage we were measuring the *coverage* accuracy of the program.

Each run of the experiment was parameterised by two parameters: p, the percentage of the logic program that was deleted, and m, the number of examples seen by Progol 5.0 in order to reconstruct the missing predicates.

A run proceeded as follows:

- The original program (defined in the background knowledge) has p percent of its clauses deleted.
- Measure the accuracy of the program with depleted background knowledge.
- Generate m random examples of the success set of the complete program.
- Run Progol 5.0 with the incomplete program as background knowledge using the generated examples.
- Measure the accuracy of the repaired program.

Each run was repeated 10 times, for each of the possible combinations of values of $p \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$ and $m \in \{10, 20, 40, 80, 120, 160\}$, for each of the original logic programs in base $n \in \{2, 3, 4, 5\}$.

3.3 Theoretical Results for the Domain

The theoretical bound 2 requires that one estimates the size in bits of the information content of the missing predicates.

In reconstructing a background clause in base n, Progol 5.0 has a choice of a certain number, l, of predicate symbols for the head, and a choice of the same l

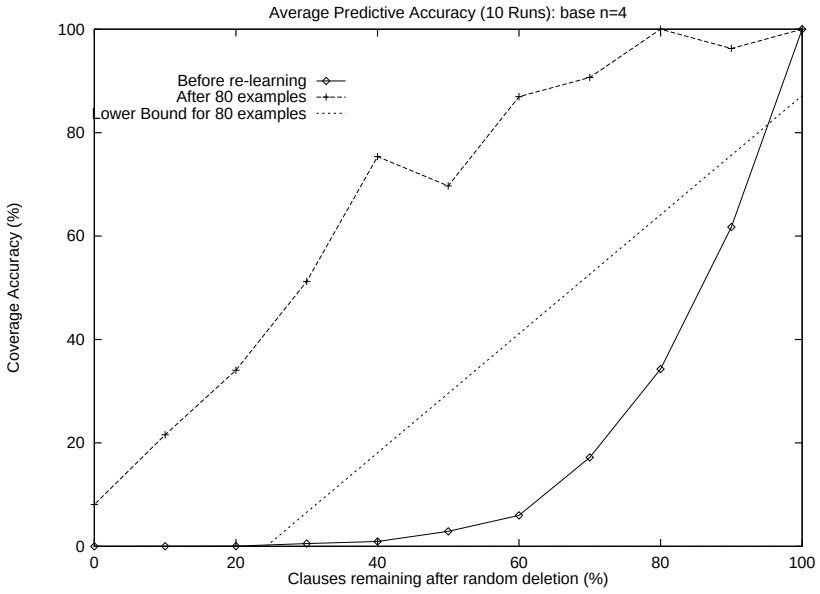


Fig. 1. Experimental result for the case $n = 4, m = 80$. Complete results are shown in appendix A.

predicate symbols for the one atom in the body. It also has a choice of n function symbols to add to the string. (`one`, `two`, etc).

If it is assumed that all of the possible choices are assigned equal probability, then the information content of a single clause of this kind is therefore info(n) = $\log_2(nl^2)$. The values of this function for the different possible values of n are given in Table 1.2.
The bound on the expected experimental error is then:

$$\mathrm{EE}(m) \leq \frac{1.51 + 2\ln(m) + c.\mathrm{info}(n)\ln(2)}{m}$$

where c is the number of missing clauses.

3.4 Results

A typical result is shown in Figure 1. This graph shows the case that $n = 4, m = 80$. The lower curve is the accuracy of the incomplete logic program before theory recovery. The upper curve is the accuracy of the logic program after theory recovery using m examples. The straight line is the theoretical bound.

As the graph shows, the bound fits the results well, and runs parallel to the observed experimental accuracy. The results for other values of n (the base), and m (the number of examples) are similar. Results for all the values of n and m are given in Appendix A.

The experiments are consistent with the theoretical bound, and the calculation of the information content of clauses in this domain (see Table 1.2).

4 Conclusions

This paper has introduced a theoretical model for analysing learning when performing theory recovery. We derived an average case error bound for the error of a Progol-like learner in such a situation and showed that the bound held and was reasonably tight under experiment. The experiments used logical backpropagation, a feature of Progol 5.0, to perform theory recovery.

This work is part of a wider programme to analyse multiple-task learning within a relational (ILP) setting. In particular, we analyse a particular issue raised by the *repeat learning* framework introduced in [1], that of learning under a prior distribution that is incorrect. The experiments differ from those conducted in the previous paper in that theory recovery, rather than predicate invention, is used to alter background knowledge. However the repeat learning framework does not specify the particulars of how the linguistic bias of the learner is to be altered. Although there is no multiple-task learning in this work - we are only ever learning one concept - the link with repeat learning comes in the form of the updating of background knowledge and hence the updating of the linguistic bias of the learner. In both models the learner is missing predicates in the background knowledge. The analysis could easily be extended to the case when one is learning more than one task. This would be the natural direction in which to extend the research.

Acknowledgements

Rupert Parson would like to thank the William Esson Bequest Fund, by whom he is partially funded. Khalid Khan would like to thank his parents. This work was supported partly by the Esprit Long Term Research Action ILP II (project 20237) and EPSRC grant GR/K57985 on Experiments with Distribution-based Machine Learning.

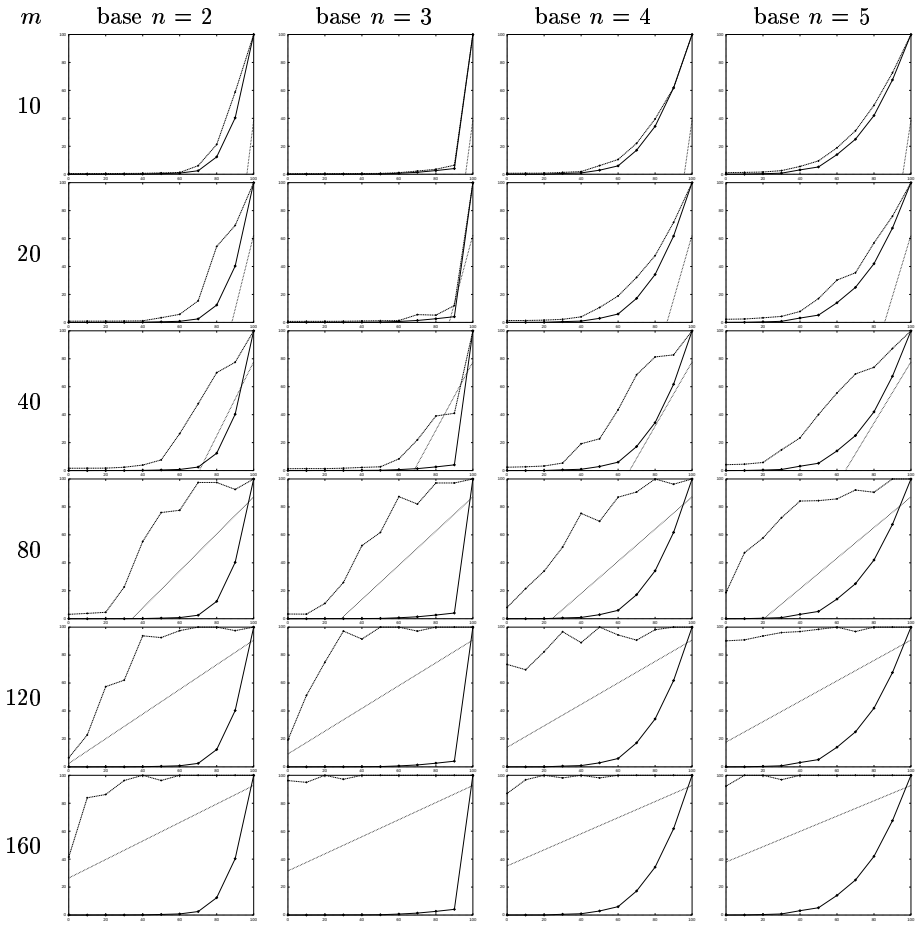
References

1. K. M. Khan, S. H. Muggleton, and R. D. G. Parson. Repeat learning using predicate invention. In David Page, editor, *Proceedings of the Eighth International Conference on Inductive Logic Programming*, volume 1446 of *LNAI*, pages 165–174, Madison, Wisconsin, USA, 1998. Springer.
2. S. Moyle and S. H. Muggleton. Learning programs in the event calculus. In Nada Lavrač and Sašo Džeroski, editors, *Proceedings of the Seventh International Conference on Inductive Logic Programming*, volume 1297 of *LNAI*, pages 205–212, Prague, Czech Republic, September 1997. Springer.
3. S. H. Muggleton. Inverse entailment and progol. *New Generation Computing*, 13:245–286, 1995.
4. S. H. Muggleton. Learning from positive data. In *Proceedings of the Sixth Workshop on Inductive Logic Programming*, Stockholm, 1996.
5. S. H. Muggleton and C. D. Page. A learnability model for universal representations. Technical report, Oxford University Computing Laboratory, Oxford, UK, 1997.

6. S. H. Muggleton and L. De Raedt. Inductive logic programming: theory and methods. *Journal of Logic Programming*, 19,20:629–679, 1994.

7. M. E. Stickel. A prolog technology theorem prover: a new exposition and implementation in prolo. *Theoretical Computer Science*, 104(1):109–128, 1992.

A Experimental Results



B Progol 5.0: Theory recovery by Logical Back-propagation

The theory recovery mechanism that is applied to the experiments in the paper is provided by *logical back-propagation* [2], a form of generalised inverse entailment [3]. The problem specification for ILP is: *given* background knowledge B and examples E; *find* the simplest consistent hypothesis H s.t. $B \wedge H \models E$. In the case that both E and H are single Horn clauses then by inverse entailment it is possible to generate the conjunction of all ground literals that are true in all models of $B \wedge \overline{E}$, denoted by $\overline{\bot}$ (i.e. $B \wedge \overline{E} \models \overline{\bot} \models \overline{H}$). In logical back-propagation examples of an observational predicate are used to augment the definition of a related theoretical predicate. In the following, examples of sentences (predicates for s) are used to augment a definition for noun phrase (np).

Example 1. Natural language processing. Given background knowledge

$$B = \begin{cases} s(A,B) \leftarrow np(A,C), vp(C,D), np(D,B) \\ np(A,B) \leftarrow det(A,C), noun(C,B) \end{cases}$$

example $E = s([the, nasty, man, hit, the, dog], [])$, and a prior hypothesis $H = np(A,D) \leftarrow det(A,B), adj(B,C), noun(C,D)$ then by inverse entailment

$$\begin{aligned} \overline{\bot} = & \neg s([the, nasty, man, hit, the, dog], []) \\ & \wedge \neg np([the, nasty, man, hit, the, dog], [hit, the, dog]) \\ & \wedge det([the, nasty, man, hit, the, dog], [nasty, man, hit, the, dog]) \\ & \wedge \ldots \wedge np([the, dog], []) \end{aligned}$$

The most specific (non-definite) clause that results from variablising terms (guided y mode declarations) is

$$\bot = s(A,B); np(A,C) \leftarrow det(A,D), adj(D,E), noun(E,C), vp(C,F), np(F,B)$$

The generation of $\overline{\bot}$ in the above example requires derivation of $\neg s$ and $\neg vp$, which leads to obvious difficulties when using a Horn clause theorem prover. To overcome this the implementation of logical back-propagation makes use of mechanisms from Stickel's Prolog Technology Theorem Prover [7]. Clauses are constructed to provide definitions for negated literals. For example, for the clause $p \leftarrow q$, the *contrapositive* clause $\neg q \leftarrow \neg p$ is added, allowing the possibility of $\neg q$ being derived using a Prolog interpreter.

Not all clauses of a theory need have their contrapositive added when implementing generalised inverse entailment. A relevance map based on the calling diagram among predicates is used to determine the additional contrapositives required. The contrapositive required to generate $\bot$ for the example above is $\neg np(A,C) \leftarrow \neg s(A,B), vp(C,D), np(D,B)$. This enables the derivation of $\neg np$ for the generalisation of $\bot$. The theoretical and observational predicates involved in the generalisations are communicated by the user to Progol 5.0 by way of mode declarations.

C Proof of Theorem 2

Proof. For all $H_i \in \mathcal{H}_P$,

$$D_{\mathcal{T}}(H_i) = \frac{a}{i^2} \leq \frac{a}{(j^2/2^k)} = \frac{a2^k}{j^2} \text{ Since } j \leq 2^{k/2}\, i.$$

So for the learner's ordering over hypotheses :

$$\sum_{j \geq n+1}^{\infty} D_{\mathcal{T}}(H_j) \leq \sum_{j \geq n+1}^{\infty} \frac{a2^k}{j^2} \leq \int_{j \geq n}^{\infty} \frac{a2^k}{j^2}\, dj = \left[-\frac{a2^k}{j} \right]_{j=n}^{\infty} = \frac{a2^k}{n}$$

The proof now proceeds in a similar manner to the proof of Theorem 1 given in [4].

$$\begin{aligned} EE(m) &= \sum_{T \in \mathcal{H}} D_{\mathcal{T}}(T) \sum_{E \in T^m} D_X(E|T) Error(L(E), T) \\ &\leq \sum_{T \in \mathcal{H}_n} D_{\mathcal{T}}(T) \sum_{E \in T^m} D_X(E|T) Error(\mathcal{H}_n(E), T) \; + \sum_{T \in \mathcal{H} \setminus \mathcal{H}_n} D_{\mathcal{T}}(T).1 \\ &\leq \sum_{T \in \mathcal{H}_n} D_{\mathcal{T}}(T) \sum_{E \in T^m} D_X(E|T) Error(\mathcal{H}_n(E), T) \; + \frac{a2^k}{n} \end{aligned}$$

Let $\tau_{mn}(\epsilon) = \{E' : E' \in T^m \text{ and } Error(\mathcal{H}_n(E'), T) \leq \epsilon\}$. Then:

$$\begin{aligned} &\sum_{T \in \mathcal{H}_n} D_{\mathcal{T}}(T) \sum_{E \in T^m} D_X(E|T) Error(\mathcal{H}_n(E), T) \\ &= \sum_{T \in \mathcal{H}_n} D_{\mathcal{T}}(T) \sum_{E \in \tau_{mn}(\epsilon)} D_X(E|T) Error(\mathcal{H}_n(E), T) \\ &+ \sum_{T \in \mathcal{H}_n} D_{\mathcal{T}}(T) \sum_{E \in T^m \setminus \tau_{mn}(\epsilon)} D_X(E|T) Error(\mathcal{H}_n(E), T) \\ &\leq \epsilon + Pr(\exists H \in \mathcal{H}_n : Error(H, T) > \epsilon \text{ and } x_1, \ldots, x_m \in T \cap H) \\ &\leq \epsilon + n e^{-\epsilon m} \end{aligned}$$

Thus,

$$EE(m) = \epsilon + n e^{-\epsilon m} + \frac{a2^k}{n}$$

Optimal values of n and ϵ are found by successively setting to zero the partial derivatives of n,ϵ and solving. This gives $\epsilon = \frac{\ln mn}{m}$ and $n = 2^k am$. Substituting gives

$$EE(m) \leq \frac{2 + k \ln 2 + 2 \ln m + \ln a}{m} < \frac{1.51 + 2 \ln m + k \ln 2}{m}$$