The Justification of Logical Theories based on Data Compression

Ashwin Srinivasan Stephen Muggleton Michael Bain

The Turing Institute,
36 North Hanover Street,
Glasgow G1 2AD,
UK

Abstract

Non-demonstrative or inductive reasoning is a crucial component in the skills of a learner. A leading candidate for this form of reasoning involves the automatic formation of hypotheses. Initial successes in the construction of propositional theories have now been followed by algorithms that attempt to generalise sentences in the predicate calculus. An important defect in these new-generation systems is the lack of a clear model for theory justification. In this paper we describe a method of evaluating the significance of a hypothesis based on the degree to which it allows compression of the observed data with respect to prior knowledge. This can be measured by comparing the lengths of the input and output tapes of a reference Turing machine which will generate the examples from the hypothesis and a set of derivational proofs. The model extends an earlier approach of Muggleton by allowing for noise. The truth values of noisy instances are switched by making use of correction codes. The utility of compression as a significance measure is evaluated empirically in three independent domains. In particular, the results show that the existence of compression distinguishes a larger number of significant clauses than other significance tests. The method also appears to distinguish noise as incompressible data.

1 Introduction

The ability to form inductive hypotheses is a fundamental requirement for a learner. This is evident even in the most rudimentary robots that plan actions based on continuity assumptions about the state of the world around them. Literature on the automated formation of hypotheses has largely concentrated on

their construction. Initial successes in the induction of propositional theories (Michalski 1983; Quinlan 1986; Clark and Niblett 1989) have been followed by algorithms that attempt to generalise sentences in the predicate calculus (DeRaedt and Bruynooghe 1992; Muggleton and Feng 1990; Quinlan 1990; Rouveirol 1991). However, while the need to justify inductive theories has long been recognised (Carnap 1952; Popper 1972) and is an important topic in the work by Plotkin (1971), it remains largely unexplored within machine learning. In particular, although propositional theories have been justified using statistical measures of significance (see for example, Clark and Niblett 1989), probability estimates (Cestnik 1991), or simplicity (Quinlan and Rivest 1989), there appears to be no clear method for evaluating first-order theories.

In this paper we describe a model that addresses the issue of hypothesis justification within the framework of learning first-order theories. There are two principal features in our approach. First, given a set of alternate first–order hypotheses for some data, the one chosen is that which is least likely to have explained the data by chance. Second, the model appears to be unique in that it accounts for the use of relevant prior knowledge by the hypotheses. Significance measures used in propositional learners are typically concerned only the accuracy of a theory and number of examples it explains. Simply adopting such a measure in a first-order learner would mean giving up the ability to judge if the explanation was based on chance coincidences in the prior knowledge.

2 Incremental hypothesis construction

We adopt the logical framework described by Shapiro (1983). In this, the task of the learning process is to infer some (unknown) target logic program P. There is an intended interpretation (model) for this program M_P. We assume the presence of some predicates whose interpretation is fixed and does not change in M_P. These predicates constitute the background knowledge B. At any given stage i of incremental hypothesis construction, a learner has accumulated examples known to be included in M_P (denoted ${E_i}^+$) and those known to be excluded from it (denoted ${E_i}^-$) Current learning systems attempt to construct hypotheses H_i under the following constraints:

$$E_i = {E_i}^+ \cup {E_i}^-$$

$${E_i}^+ \cap {E_i}^- = \emptyset$$

$$B \wedge H_i \vdash {E_i}^+$$

$$B \wedge H_i \wedge {E_i}^- \nvdash \square$$

There are in general, infinitely many consistent hypotheses that satisfy these requirements. A snapshot of the construction of one such hypothesis by a "learning" (induction) machine is shown in Figure 1. Despite satisfying the constraints

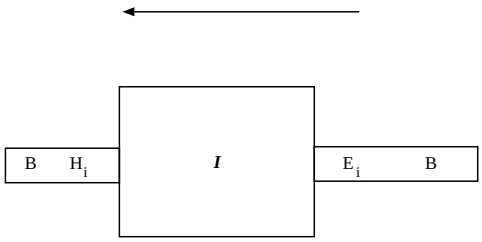

Figure 1: A stage in incremental hypothesis construction

listed above, a hypothesis H_i may be incorrect or incomplete with respect to M_P. This is remedied at subsequent stages when training examples exposing the problem become available to the learner.

There are some practical difficulties with this model of learning:

- Examples may arise from a noisy data source and at a given stage, we may have a theory that appears inconsistent. There is no mechanism for accepting such a theory.
- Even with noise-free data, there may several consistent hypotheses to explain the data. There is no method of evaluating the explanatory power of these hypotheses.
- There is no direction on how the examples for training are to be selected.

This paper is concerned with the first two issues. We note in passing that the normal approach adopted to address the third problem involves either random data–sampling or asking queries of an oracle. The former approach may require a large number of examples before converging on the target program. Recent efforts (Bain, 1991; Morales 1991) attempt to actively guide the generation of new examples.

3 Incremental hypothesis evaluation

In this section we consider evaluating competing hypotheses constructed at a single stage in the incremental learning process described earlier. Although in the 1950's Carnap (1952) and others suggested "confirmation theories" aimed at providing statistical significance tests for logic-based inductive inference, various difficulties and paradoxes encountered with these approaches meant that they were never applied within machine learning programs (Mortimer 1988). Instead machine learning researchers have for the most part made use of Occam's razor with various ad hoc definitions of complexity. However the lack of a clear model underlying such approaches makes it difficult to associate any independent meaning to the simplicity of a hypothesis.

We concentrate on an approach described by Muggleton in (Muggleton 1988). This addresses the theory evaluation question using ideas from algorithmic information theory (Chaitin 1987; Kolmogorov 1965; Solomonoff 1964). In this approach, the utility of a Horn clause program is measured on a reference Turing machine. Unlike the machine in Figure 1 this reference machine behaves like a *deduction* machine. The input to this machine is a program that has two distinct parts: a Horn clause theory and a proof specification. The machine uses the latter to output examples derivable by the theory (Figure 2).

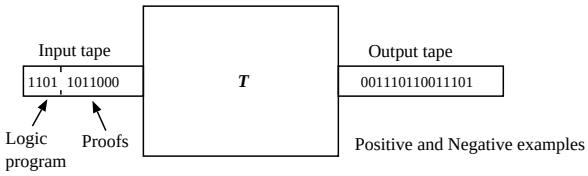

Figure 2: A Turing machine model for evaluating logic programs

Following algorithmic information theory, the Horn clause theory is said to be *compressive* if the length of an encoding of the input tape (in bits) is shorter than that of the output tape. The use of a reference machine (as opposed to a universal one) is motivated by demonstrating that the probability of obtaining a compressive theory by chance decreases exponentially with the amount of compression for any machine. This gives a clear meaning to the notion of compression. For completeness, we reproduce the proof of this result here.

Theorem 1 *Let Σ_n be the set of all binary strings of length n, T be an arbitrarily chosen reference Turing machine and the k-bit-compressible strings of length n, $K_{n,k}$, be defined as $\{y : y \in \Sigma_n, x \in \Sigma_{n-k}, T(x) = y\}$. The set $K_{n,k}$ has at most 2^{n-k}elements.*

Proof Since Turing machines are deterministic T either induces a partial one-to-one or many-to-one mapping from the elements of Σ_{n-k} to the elements of $K_{n,k}$. Thus $|K_{n,k}| \leq |\Sigma_{n-k}| = 2^{n-k}$. $\square$

Corollary 2 *The probability of a binary string generated by tossing an unbiased coin being compressible by k bits using any Turing machine T as a decoding mechanism is at most 2^{-k}.*

These results provide the theoretical justification for the material in this paper. Hypotheses are evaluated on the basis of the compression they produce. Theories with higher compression are preferred as they are less likely to explain the data by chance. This deductive evaluation of hypotheses nicely complements the induction process depicted in Figure 1. Evaluating the hypothesis constructed at any stage requires us to check the compression produced by the hypothesis at

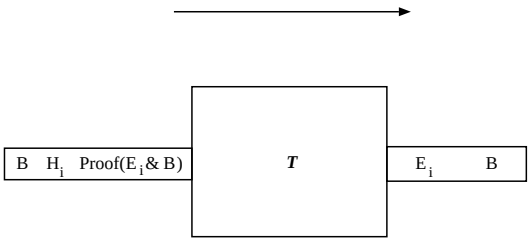

Figure 3: A Turing machine model for incremental hypothesis evaluation

that stage. As seen in Figure 3, this is, in some sense, like *reversing* the theory construction stage.

In Figure 3, the machine T has the following input-output behaviour:

$$T(\mathrm{I}(B \wedge H_i, \mathrm{Proof}(B \wedge E_i | B \wedge H_i))) = \mathrm{O}(B \wedge E_i)$$

where I, O and $Proof$ are input, output and proof encodings for T. The k-bit compression achieved by the theory on the input tape is then:

$$k \;=\; |\mathrm{O}(B \wedge E_i)| - |\mathrm{I}(B \wedge H_i, \mathrm{Proof}(B \wedge E_i | B \wedge H_i))|$$

To be of practical value, we also have to specify the encodings I, O and $Proof$ (this aspect was left unexplored in Muggleton 1988). Clearly, we would like an encoding that is guaranteed to be optimal. However, results from algorithmic information theory have shown that finding an optimal encoding is equivalent to the halting problem. Consequently, we have to be satisfied with a sufficient test for significance based on the encoding scheme adopted. In the next section we describe an efficient coding scheme that can be used to evaluate Horn clause theories for the setting described by Figure 3.

4 An encoding scheme for the evaluation of logical theories

4.1 Input tape encoding

The basic premise of the compression model is that efficient (ideally optimal) encodings are found for input and output tapes of a machine. The components of the input tape for the machine in Figure 3 are shown in Figure 4. In this figure, the first three sections (upto the encoding of B) constitute header information.

The reference machine T interprets the input tape as follows:

- The size of the background knowledge (number of atoms and/or clauses in B) allows the machine to distinguish between it and the hypothesis constructed. The number of clauses is used for two purposes. It states how

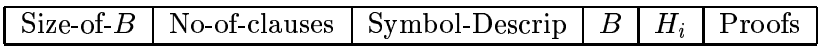

Figure 4: Sections of the input tape for machine T

many clauses to expect on the input tape and it is also used to construct a special clause separator symbol. Although the background knowledge can consist of clauses, it is common practice with current first-order learning systems to represent it by a ground model (Muggleton and Feng 1990; Quinlan 1990). The need to specify a symbol description in the header is elaborated shortly.

- A header is generated for the output tape using B and the examples specified by the proof encoding. This header has the same form as that on the input tape. The difference is that the clause count and symbol description refer to the output tape.

- Each example on the output tape is generated by its proof encoding. The machine acts as a logic program interpreter. For each example, the proof encoding specifies the clauses in the hypothesis and background knowledge that are used to derive the example.

- The machine outputs the atoms and/or clauses in B without interpreting them onto the output tape.

4.1.1 Background knowledge and hypothesis encoding

A logic program can be viewed as a sequence of symbols. A near optimal choice for encoding these symbols involves the use of prefix codes. We assume a vocabulary S of symbols where each symbol $s \in S$ appears with relative frequency p_s. A prefix code is a function

$$Prefix : S \rightarrow \{0,1\}^*$$

which has the property that no code is a prefix of any other code. This property ensures that codes are self-delimiting. Information Theory (Shannon and Weaver 1963) tells us that the optimal code length for symbol s is $-log_2 p_s$ bits. Huffman coding (Gallager 1968) is a prefix coding which achieves approximately this code length for each symbol.

In order for the machine to "understand" the encoding of symbols in B and the hypothesis, it is necessary to define the frequencies of the different symbols used. This can then be used to construct a code-book for the message on the input tape. The components of this symbol description header are shown in Figure 5.

PSyms	Zero	FSyms	Zero	Vars	Zero	PSym-arity	FSym-arity

Figure 5: Sections of the symbol description header

Predicate, function and variable symbols have different codes. A prefix table, such as that of the predicate symbols, consists of the individual symbol counts in order of their appearance. This sequence of natural numbers is sufficient for a unique reconstruction of the codes used in the theory. The clause separator symbol (constructed using the clause count) is treated as though it were a predicate symbol. The arities for predicate and function symbols are also number sequences whose orders correspond to those in the prefix tables. Clearly the clause separator "predicate" symbol has no arity. "Zero" is defined to be the encoding of the natural number 0 and acts as a separator for different sections of the header. Separators are not necessary to delimit the arities since their number is determined by the predicate and function symbol counts. In order to avoid infinite regress we must find a universal coding for the natural numbers that appear in the header. Natural numbers can be encoded using prefix codes given an appropriate prior distribution. Rissanen (1982) shows that an optimal distribution can be defined for which the code length $L(n)$ is bounded as follows

$$log_2 n < L(n) < log_2 n + r(n)$$

where $r(n)/log_2 n \to 0$ and $r(n) \to \infty$ as $n \to \infty$. For example, a universal code for numbers could be constructed using N 0's followed by a 1 followed by $2^N - 1$ binary encoded digits $N = 0, 1, 2....$ This gives the following codes for natural numbers.

$0 \to 1$
$1 \to 010$
$2 \to 011$
$3 \to 001000$
..
$11 \to 00010000000$

The approximate code-length for the natural number n using this code is $\lceil log_2(n+2) \rceil + \lceil log_2 \lceil log_2(n+2) \rceil \rceil$. The prior probability distribution assumed for these numbers is $p(n) = (n \log_2{}^2 n)^{-1}$. In general, the lengths of all such universal codes for natural numbers have $log_2 n$ as the dominant term and to first approximation can be taken as bounded by $log_2(n+2) + 2log_2 log_2(n+2)$.

We assume that logical theories are expressed as a set of Prolog clauses. The following grammar gives the syntax of our encoding of theories.

Theory ::= { Clause }$^{\text{No of clauses}}$
Clause ::= Atom Clause | Stop
Atom ::= PredSym [Negated] {Term}$^{\text{Arity(PredSym)}}$
Term ::= ["0"] FuncSym {Term}$^{\text{Arity(FuncSym)}}$ | ["1"] VarSym
Negated:= "0" | "1"

Example 3 *Consider encoding a theory that consists of the following clauses*

normal(Year) :- year(Year), not leap4(Year).
leap4(Year) :- mod(Year,4,0).

*The predicate symbols in order of appearance are normal/1, year/1, leap4/1 and mod/3. In addition, there is a clause separator "predicate". The function symbols in order of appearance are 4/0 and 0/0. There is a single variable symbol that occurs 5 times. The encodings for the symbol counts and arities follow (***P***, **D** and **B** stand for Parse, Decimal and Binary). For the example, we shall use the natural number coding described earlier.*

P	P-count					F-count			V-count	
D	1	1	2	1	0	1	1	0	5	0
B	010	010	011	010	1	010	010	1	001010	1
P	P-arity					F-arity				
D	1	1	1	3		0	0			
B	010	010	010	001000		1	1			

This is a total of 44 bits. Huffman codes for the clause separator and the predicate symbols are 00, 10, 110, 01 and 111 respectively. The function symbols have the codes 0 and 1. Since there is only one variable it does not need a code. Thus the coding for the theory itself is as follows.

P	normal(...)	year(...)	not leap4(...)	Sep	leap4(...)	mod(...)	Sep
P	Atom	...					
B	10 1	110 0 1	01 1 1	00	01 1	111 1 0	00

This is a further 21 bits. The encoding illustrates several points. First, since we are only concerned with Prolog clauses, the head of the clause does not have to be flagged for negation. Further, codes are not needed once the identity of the symbol is determined from the count and arity information. This is evident in the case of encoding the variable: all that is needed is to indicate where Year occurs (using the flag "1"). It is further illustrated in the encoding for mod/3. From the header, it is clear that the predicate has 3 arguments. From the symbol counts, these have to be single occurences of Year, 4/0 and 0/0. Thus once the position of the variable is known (from the "1" flag), there is no need to flag the remaining arguments as being function symbols. From the encoding of the second argument (by "0"), the third is determined and does not have to be stated.

Although for longer theories we would expect that the header information would be considerably shorter than the statement of the theory, prefix coding may not be very efficient for small theories. Clearly, each symbol type (predicate, function, variable) can be coded differently with bits at the front of the input tape indicating the type of coding adopted for each symbol. This will change the contents of the header. Within our implementation, we can select the most efficient amongst three different coding schemes for a symbol: universal natural number code, a fixed-length code or a prefix code (listed in order of increasing header information). For each scheme, we use the non-integral code length as an optimal estimate. The assumption here is that this value can be reached when sufficiently long messages are encoded.

4.1.2 Proof encoding

The reference machine T takes the theory and a proof encoding and generates the examples. Derivational proofs are represented as sequences of choices to be taken by a Prolog interpreter. We illustrate this with a simple example.

Example 4 *Consider a hypothesis that consists of the following clause:*

p(X,Y,Z) :- q(X,Y), r(Y,Z).

Background knowledge consists of the following ground atoms:

q(1,2). *r(2,3).*
q(2,3). *r(2,4).*
q(3,4). *r(3,4).*
q(4,5). *r(3,6).*

Consider deriving p(1,2,3) using this hypothesis and background knowledge. In deriving the atom, the interpreter has to first choose which clause of p/1 *to execute. In this case, there is only one such clause and thus, no choice. The first atom in the body of the chosen clause q(X,Y) can be matched against any one of the set of 4 ground unit clauses in the background. Specifying the choice for the proof (that is q(1,2)) requires 2 bits. This choice determines the choice of the variable Y and constrains the interpreter to one of 2 possible choices for r(Y,Z). Specifying the atom r(2,3) requires 1 more bit. The complete choice-point encoding for the atom p(1,2,3) is specified by the string 011.*

Note that the choice-point encoding is procedural: hypotheses that are more efficient to execute have lower choice complexity. This provides a natural bias towards learning efficient clauses. We want to be able to encode a sequence of proofs; one for each example on the output tape. This can be achieved by preceding the series of proofs by an encoding of the number of examples. This

encoding of proofs is sufficient for examples which are derivable from range-restricted (generative) theories. However, it has to be extended to accomodate for the following:

1. For non-generative clauses, substitutions have to be provided for variables that do not occur in the body of the clause (since these will never be bound by any choice specification). The function codes for any substitutions needed appear after the choice specifications.

2. Incorrect or incomplete theories can still be used for compressing data to a certain degree. The theory in Example 3 is an example of a useful, though incorrect, theory for distinguishing leap years from normal ones.

To address the second issue we distinguish three categories of results obtained from the theory:

1. Correct. In this case the truth-value of the derived fact agrees with the intended interpretation.

2. Error of commission. The truth-value of the derived fact is the opposite of the intended interpretation.

3. Error of omission. The fact cannot be derived.

Each choice-point encoding is preceded by a prefix code indicating its category. The prefix codes for the categories are constructed using three numbers indicating the counts in each category. These numbers are coded using the universal coding scheme and precede the proof encoding on the input tape. Clearly, the total number of examples no longer have to be specified. The encoding of the errors of omission is the same as their explicit encoding.

4.2 Output tape encoding

The output tape encoding is almost the same as that of the input tape (see Section 4.1). The difference is that instead of the hypothesis and proofs, examples are explicitly encoded as atoms.

4.3 A comparison with FOIL's encoding scheme

The selection criterion used by the model in previous sections is one of minimising description length. Within a first-order framework, the learning system FOIL (Quinlan 1990) uses an encoding length criterion motivated by the Minimum Description Length principle described in (Rissanen 1978). Viewed as a

Turing machine model, the "output" tape length is computed using the following function:

$$log_2(|E|) + log_2\left(\left(\begin{array}{c}|E|\\|E^+|\end{array}\right)\right)$$

where $|E|$ is the number of examples in the training set, and $|E^+|$ is the number of positive instances covered by the hypothesis. The "input" tape encoding is simply the encoding of the hypothesised clause. This is given by

$$\sum(1 + log_2(|PredSyms|) + log_2(|Args|)) - log_2(n!)$$

where n is the number of literals in the clause, $|PredSyms|$ is the number of predicate symbols in the background knowledge, $|Args|$ is the number of possible arguments. The number of bits for the input tape must be less than the number of bits for the output tape. Comparing this to our encoding method we note the following:

1. There is no notion of proof encoding and consequently no apparent instructions on how to reproduce the examples on the output tape.

2. The hypothesis length grows logarithmically with the number of predicate symbols in the background knowledge. A particular clause with given coverage can be invalidated by adding predicates to the background knowledge. There is thus no distinction between relevant and irrelevant background knowledge. In contrast, in our model, the effect of irrelevant background knowledge is equal on input and output tapes.

3. Negative examples are treated asymmetrically.

4. The factor of $log_2(n!)$ is used to correct for the ordering of literals in clauses. The assumption is that any literal ordering has equivalent information content. However in our model the literal ordering affects the choice complexity for proving examples, and thus the corresponding lengths of proofs.

5 Compression as a significance measure

In this section, we illustrate the utility of using compression as a measure of confidence in clauses learned for three different problems:

1. Prediction of protein secondary structure. The prediction of protein secondary structure from primary sequence is an important unsolved problem in molecular biology. Recently it has been shown that the use of relational learning algorithms (see King and Sternberg 1990; Muggleton *et al.* 1991) can lead to improved performance.

2. Modelling drug structure-activity relationships. The design of a pharmaceutical drug often requires an understanding of the relationship between its structure and chemical activity. Rules learned to model this relationship have been recently been shown to perform better than existing numerical methods (King *et al.* 1992).

3. Learning rules of illegality for the KRK chess end-game. Despite its simplicity, the KRK problem remains the test-bed for ILP techniques. We evaluate the compression measure with different levels of noise. We use a simple noise model that randomly changes the sign of a fixed percentage of the data-set. This is similar to the Classification Noise Process introduced by Angluin and Laird in (Angluin and Laird 1988). [1] Training and testing is done with 1000 examples in each set.

The error rate of a clause on training data (that is, the *reclassification* or *resubstitution* error rate) can give a highly misleading estimate of the accuracy of the clause on the whole domain. The protein folding domain provides a striking example of this. Figure 6 shows a correlation diagram of the accuracies of clauses on training and unseen (test) data for the protein structure prediction problem. Each point in this figure represents a first-order clause constructed for predicting the position of an α-helix. [2] The rank correlation between the training and test set accuracies in the figure is 0.3.

In this section we evaluate compression as a method of identifying "stable" clauses in the following sense. In general, we find a difference between the accuracy of a clause over training and test data. Consider the variation of this difference over a set of clauses judged to be significant by some measure. Small variations indicate that the significance measure has largely avoided selecting clauses that over-fitted the training data.

We compare the compression measure against the following alternatives:

1. Training set accuracy. The motivation for this is that the observed sample accuracy is a consistent estimate of the domain accuracy of clauses. In the presence of noise, this measure is not very efficient (as seen from Figure 6).

2. Training set coverage. These are the number of examples that can be derived using a clause. The motivation for this comes from the expectation that the greater the cover, the less likely that sample accuracy will be far

[1] In this, a noise of η implies that (independently) for each example, the sign of the example is reversed with probability η. This is not the only random noise process possible. For example, a noise of η here corresponds to a class-value noise of 2η in that adopted by Quinlan (1986) and Donald Michie (private communication) advocates a process that preserves the underlying distribution of positive and negative examples.

[2] The clauses were learnt by the learning program Golem (Muggleton and Feng 1990). Note that in this domain the accuracy of a rule which predicted all positions to be part of an α-helix is 0.5.

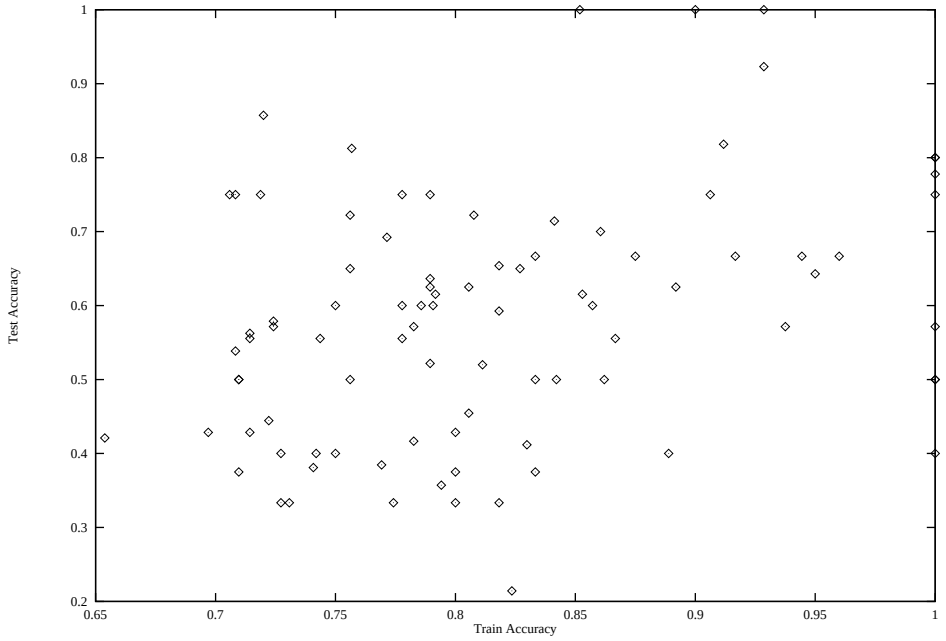

Figure 6: Training and Test set accuracies of clauses predicting protein structure

from domain accuracy. We consider the coverage of a clause as a percentage of the total training set.

3. Bayesian posterior variance. This measure is described in (Cussens 1992). It concerns the variance of the posterior distribution for the domain accuracy. This distribution is obtained by using Bayes rule to update a uniform prior. If a clause covers n examples, t_p of which are true positives then the value of this measure is:

$$VarianceMeasure = 1 - 12\frac{(t_p+1)(n-t_p+1)}{(n+2)^2(n+3)}$$

The comparison is done as follows.

- For each domain, we first find the compressive clauses. These are the clauses that the compression measure reports as having avoided over-fitting (that is, significant).
- The standard deviation of the difference in training and test accuracies s of compressive clauses is determined.
- For each of the other measures, we order clauses in increasing value of the measure. We want to select the clauses reported significant by the measure. Unlike the compression measure, there is no obvious threshold to determine

this clause-set. Consequently, we chose a threshold that selects the largest clause-set with an s value guaranteed to be less than or equal to that of compressive clauses. This ensures that the set of significant clauses is at least as "stable" as the compressive one.

- We compare the measures on the basis of the number of clauses reported significant by each measure.

Figure 7 shows the number of clauses selected as significant by each measure. The thresholds required to obtain these clauses are shown in Figure 8.

Domain	Total clauses	Sd of signif. clauses	Number of significant clauses			
			Compress.	Acc.	% Cover	Var.
Proteins	89	0.09	16	0	7	9
Drugs	107	0.05	103	52	79	76
KRK(5%)	19	0.08	19	19	19	19
KRK(10%)	36	0.13	31	0	12	23
KRK(20%)	40	0.15	33	17	36	33

Figure 7: A comparison of significance measures

Domain	Selection thresholds for significant clauses			
	Compression	Accuracy	% Cover	Variance
Proteins	>0	>0.65	>2.9	>0.97
Drugs	>0	=1.0	>2.1	> 0.99
KRK(5%)	>0	>0.50	>0	>0.86
KRK(10%)	>0	>0.54	>11.1	>0.92
KRK(20%)	>0	>0.7	>0.59	>0.83

Figure 8: Selection thresholds for significance measures

The figures illustrate the following points:

- Training accuracy appears to be clearly the worst judge of clauses. Compression usually distinguishes a larger number of significant clauses than the other measures. Although there is probably not enough data to claim that it does significantly better, we expect it to perform well in domains such as the one predicting protein structure. A characteristic of this domain is that the clauses are highly relational and the background knowledge is quite complex. In these cases, simple measures relying on coverage and accuracy may not be enough.

- In practice, the selection threshold for a significance measure is pre-determined. A compression-based choice is easy: all that is required is to select clauses that are compressive. On the other hand, the threshold for the other measures is not obvious.

6 Compression as a noise-meter

The results in the previous section suggest that compressive hypotheses have good "accuracy-stability". We now consider extending this to estimate the amount of noise in the domain. Besides being an interesting quantity in itself, this estimate can be used to provide loose bounds on the expected accuracy of hypotheses on unseen (test) data. Since we are dealing with noisy domains, it is necessary to distinguish between the *real* sample accuracy (on noise-free data) of a hypothesis and its *observed* one (on noisy data).

Consider a noise-free training set classified by an oracle. Let p be the fraction of examples in this data whose classification by a hypothesis agrees with the oracle's classification (this is the real sample accuracy). Clearly, $p \leq 1$. Suppose this data is then subject to noise that changes the class values of a fraction $1-q$ of the examples. That is, a fraction q of the examples is not affected by the noise process. For noisy domains $q \leq 1$. Then, on unseen data from the same source the observed accuracy of the hypothesis can at best be $pq+(1-p)(1-q) = 1-p-q+2pq$. If the hypothesis does not fit any noise at all, then its observed accuracy on the training set will be q. Its corresponding real accuracy p can at best be 1. On the other hand, if the hypothesis fits all the noise, the observed accuracy on the training set will be 1. The corresponding real accuracy will be q. Therefore, a rough estimate is that the observed accuracy of any hypothesis on independent test data from the same distribution lies in the interval $[2q^2 - 2q + 1, q]$.

The problem with this analysis is that in general, the value of q is unknown. All that is known is that q will be the observed training accuracy of a hypothesis that exactly avoids fitting any noise. One possible method of identifying such a hypothesis is to exploit the feature of noisy data as being incompressible by a hypothesis. This suggests the following procedure for determining the amount of noise in the domain. Progressively consider hypotheses with higher training accuracy. Determine the compression of each hypothesis. Estimate q as the training accuracy of the most compressive hypothesis. Clearly, given that we can never prove that our input tape encoding is the shortest one possible, adopting this definition may result in incorrectly condemning some data as noise. Thus, the value of q obtained using this procedure will be an under-estimate (alternatively, the amount of noise will be over-estimated). We test this on the KRK domain. The experiments involved introducing different levels of noise (using the classification noise model) into a data set of 10,000 examples. The noise level was then estimated using the "most compressive" hypothesis. The results tabulated in Figure

% Noise introduced	% Noise estimated
0	0
5	5.4
10	10.3
15	15.3
20	20.2
30	30.1
40	40.1

Figure 9: Noise estimation using compression

9 clearly show the tendency to over-estimate the amount of noise. The relatively large size of the training set is also important to obtain reliable estimates using the compression model. This issue is explored further in the next section.

7 Compression-guided learning: a case study

Based on the positive empirical results of the previous sections, we now describe a case study that involves incorporating the compression model in the learning process. We concentrate on the non-monotonic learning framework called closed-world specialisation (Bain and Muggleton 1991). This method progressively corrects a first-order theory by inventing (and possibly generalising) new abnormality predicates. The process has recently been used to construct a complete and correct solution for the standard KRK illegality problem (Bain 1991). However, a key issue remains to be addressed: there is no mechanism by which a non-monotonic learning strategy can reliably distinguish true exceptions from noise. For example, a strategy based on closed-world-specialisation would continue specialising until a correct theory is obtained. In noisy domains, this will necessarily result in fitting the noise. To address this, we incorporate the Turing machine compression model in Figure 3 within the non-monotonic learning framework. A simple search procedure is developed to find as compressive an explanation as possible for the data. The result of this compression-guided non-monotonic learning is evaluated on the KRK domain.

7.1 Closed-World Specialisation

Figure 10 shows an algorithm that performs the alternate operations of specialisation and generalisation characteristic of closed-world specialisation.

It is worth noting here that:

1. As in (McCarthy 1986), there is an assumption that the exceptions to a

```
start:
  PosE = positive examples of target concept
  NegE = negative examples of target concept
  return learn(PosE,NegE)

learn(Pos,Neg):
  ClauseList = []
  repeat
    C = generalise(Pos,Neg)
    if C ≠ []
      PosC = positive examples covered by C
      NegC = negative examples covered by C
      Pos = Pos - PosC
      Neg = Neg - NegC
      ClauseList = ClauseList + (C,PosC,NegC)
  until C = []

  Theory = []
  foreach (Clause,PosC,NegC) in ClauseList
    if |NegC| ≠ 0
      Theory = Theory + specialise(Clause,PosC,NegC)
    else
      Theory = Theory + Clause
  return Theory

specialise(HornClause,Pos,Neg):
  hd(V_1,...,V_n) = head of HornClause
  Body = body of HornClause
  ab = a new predicate symbol
  SpecialisedClause = hd(V_1,...,V_n) ← Body, not ab(V_1,...,V_n)
  PosE = positive examples of ab formed from Neg
  NegE = negative examples of ab formed from Pos
  return SpecialisedClause + learn(PosE,NegE)
```

Figure 10: Non-monotonic inductive inference using closed-world-specialisation (CWS)

rule are fewer than the examples that satisfy it.

2. The call to *generalise* results in an attempt to induce a (possibly over-general) rule by a learning algorithm.

3. All rules are added to the theory. Further, all negative examples covered by an over-general clause are taken to be exceptions and the clause is specialised with a (new) abnormality predicate.

7.2 Compression-based clause selection

Each correction performed by the CWS algorithm is an attempt to improve the accuracy of the theory, at the expense of increasing its size. Clearly, if the correction was worthwhile, the gain in accuracy should outweigh the penalty incurred in increasing the theory size. In encoding terms, each correction increases the theory encoding on the input tape and decreases the proof encoding. In the model in Figure 3, a net decrease in the length of the input tape occurs when the correction succeeds in identifying some pattern in the errors (that is, the errors are not noise). The new theory consequently compresses the data further by exploiting this pattern. Using this feature, we evaluate the utility of updating a theory by checking for an increase in compression. We note the following consequences of using the compression model within the non-monotonic framework adopted:

1. We can be confident of not having fitted the noise only if the theory itself is compressive.

2. With the closed-world assumption, all examples are covered. Consequently, the output tape has to be encoded only once. Input tapes for alternate theories are compared against this encoding.

3. Consider an over-general clause in the current theory. The proof encoding described ensures all variables of the clause are bound to ground terms. Specialising this clause involves adding a negated literal to its body. By appending this literal to the body, we are guaranteed that it will be ground. This ensures safety of the standard Prolog computation rule used by the Turing machine.

4. Recall that the proof encoding for each example has two parts: a choice-point specification and a proof tag. Since the negative literal appended to a clause can never create bindings, the choice-point specification remains unaltered. The size-accuracy trade-off referred to earlier therefore reduces to a trade-off between increasing theory size and decreasing tag size. Not having to recalculate the choice-point encoding for each specialisation is a major benefit as this is an extremely costly exercise.

While the aim is to obtain the most compressive subset of the clauses produced by the CWS algorithm, it is unnecessary to examine all subsets since clauses constructed as generalisations of an abnormality predicate cannot be considered independent of the parent over-general clause. For example, it makes no sense to consider the following set of clauses for explaining leap years:

normal(Year) :- year(Year), not leap4(Year).
leap400(Year) :- modulo(Year,400,0).

Despite this, there may still be an intractably large number of clause-sets to consider. Consequently, we adopt a greedy strategy of selecting clauses in order of those that give the most gain (in compression). This strategy has to confront two important issues: devising a reliable method of deciding on the "best" clause to add to the theory and the fact that adding this clause may not produce an immediate increase in compression.

A simple way to address the first problem is to select the clause that corrects the most errors. Since decreasing errors is the only way to shorten the input tape, the gains are larger for theories that make fewer errors. This works well if all clauses are of approximately the same descriptional complexity. A better estimate would account for the complexity of individual clauses as well. This can be done using average estimates of the cost of encoding predicates, functions and variables. In the experiments in the next section, this more sophisticated estimate has proved unnecessary. This is because the clauses fitting noisy data tend to correct fewer errors and therefore, considered later using the simpler estimate. For the other clauses, the gain from correcting errors dominates the loss from increased theory size.

To address the problem of local minima, it is clearly desirable to have a method of looking ahead to see if a (currently non-compressive) clause will be part of the final theory. To decide this, we calculate an estimate of the compression produced by the most accurate theory containing the clause. The clause is retained if this expected compression is better than the maximum achieved so far. Each time an actual increase in compression is produced, the theory is updated with all clauses that have been retained. Figure 11 shows how the estimate is calculated. The estimated compression will usually be optimistic because it it assumes that all errors can be compressed. Of course, one way to guarantee an optimistic estimate is to assume that there will be no increase in theory size (as opposed to the scaled estimate in Figure 11). However, this gives no heuristic power and usually only prolongs a futile search for a correct theory. Figure 12 summarises the main steps in the compression-based selection of clauses as described here. The following points deserve attention:

1. At any given stage, only some clauses produced by CWS are candidates to be added to the theory (recall the earlier statement that over-general clauses have to be considered before their specialisations).

```
estimate(Theory):
  Ncorrect = number of examples correctly classified by Theory
  Nmaximum = number of examples that the learner can classify correctly
  Outbits = length of output tape (in bits)
  OldTheory = length of Theory (in bits)
  OldTags = current length of correction tags (in bits)
  Choices = length of choice-point encoding (in bits)

  NewTheory = OldTheory × Nmaximum / Ncorrect
  NewTags = length of error tags to correctly classify Nmaximum examples
  EstInbits = NewTheory + Choices + NewTags
  return (Outbits - EstInbits)
```

Figure 11: Estimating the compression from a theory

```
start:
  ClauseList = clauses produced by CWS
  return select_clauses(ClauseList)

select_clauses(ClauseList):
  Theory = PartialTheory = []
  Compression = 0
  repeat
    PotentialClauses = clauses in ClauseList that can be added to theory
    C = "best" clause in PotentialClauses
    if C ≠ []
      PartialTheory = PartialTheory + C
      NewCompression = compression of PartialTheory
      if NewCompression > Compression
        Theory = PartialTheory
        Compression = NewCompression
      else
        EstCompression = estimate(PartialTheory)
        if EstCompression ≤ Compression return Theory
  until C = []
  return Theory
```

Figure 12: Compression-based selection of clauses produced by CWS

```
% legal(WK_file,WK_rank,WR_file,WR_rank,BK_file,BK_rank)
legal(A,B,C,D,E,F) :- not ab00(A,B,C,D,E,F).
ab00(A,B,C,D,C,E) :- not ab11(A,B,C,D,C,E).
ab00(A,B,C,D,E,D) :- not ab12(A,B,C,D,E,D).
ab00(A,B,C,D,E,F) :- adj(A,E), adj(B,F).
ab00(A,B,A,B,C,D).
ab12(A,B,C,B,D,B) :- lt(A,D), lt(C,A).
ab12(A,B,C,B,D,B) :- lt(A,C), lt(D,A).
ab11(A,B,A,C,A,D) :- lt(B,D), lt(C,B).
ab11(A,B,A,C,A,D) :- lt(B,C), lt(D,B).
```

Figure 13: A complete and correct theory for KRK-legality

2. The "best" clause refers to the clause selected using the simple error-count measure, or the more sophisticated one that accounts for the estimated theory increase. To obtain the latter requires a knowledge of the number of predicate, function and variable symbols in the clause.

3. Consider the situation when the estimated compression from adding the "best" clause is no better than the compression already obtained. Figure 12 does not acknowledge the possibility that some of the other clauses can do better. It is possible to rectify this by progressively trying the "next best" clause until all clauses have been tried.

4. The procedure in Figure 12 is reminiscent of post-pruning in decision–tree algorithms (the clauses are constructed first and then possibly discarded). A natural question that arises is whether it is possible to incorporate the compression measure within the specialisation process The analogy to zero-order learning algorithms is whether tree pre-pruning is feasible. The answer is yes, and in practice it is the mechanism preferred as it avoids inducing all clauses. The procedure in Figure 12 however serves to bring out the main features succintly.

7.3 Empirical Evaluation

We evaluate the compression-guided closed-world specialisation procedure on the KRK domain. However, contrary to normal practice, we chose to learn rules for KRK-legality (as opposed to KRK-illegality). This provides an extra level of exceptions for the specialisation method. Given background knowledge of the predicates *lt/2* and *adj/2*, Figure 13 shows the target theory. It is possible to

```
legal(A,B,C,D,E,F) :- not ab00(A,B,C,D,E,F).
ab00(A,B,C,D,C,E).
ab00(A,B,C,D,E,D).
ab00(A,B,C,D,E,F) :- adj(A,E), adj(B,F).
```

Figure 14: An "approximately correct" theory for KRK-legality

achieve an accuracy of about 99.6% without accounting for the second level of exceptions. In fact, the theory shown in Figure 14 is about 98% correct.

We experiment with learning the legality theory with artificially introduced noise on different sized training sets. For the experiments, we again adopt the classification noise model referred earlier. Finally, although the procedure described in Figure 12 is not dependent on any particular induction algorithm, the results quoted here use Golem (Muggleton and Feng 1990).

Figure 15 tabulates the percentage accuracy of the most compressive theory for different noise levels. Here "accuracy" refers to accuracy on an independent (noise-free) test set of 10000 examples. Since the compression model only guarantees reliability for compressive theories, nothing can be said about those for which compression is less than 0 (irrespective of their accuracy on the test set). In Figure 15, an entry of "_" denotes that the theory obtained is non-compressive on the training data and consequently, no claim is made regarding its accuracy on the test set. The results highlight some important points. Compressive theories do appear to avoid fitting the noise to a large extent. The price for this reliability is reflected in the amount of data required. In comparison, it is possible that other techniques may require fewer examples. However, they either require various parameters to be set (Dzeroski 1991), use *ad hoc* constraints (Quinlan 1990) or need an additional data set for pruning (Brunk and Pazzani 1991). Further, most of them are unable to offer any guarantee of reliability (the approach followed in Dzeroski 1991 can select clauses above a user-set significance threshold). In this respect, our empirical results mirror PAC (Valiant 1984) results for learning with noisy data in propositional domains (Angluin and Laird 1988): with increasing noise, more examples are needed to obtain a good theory. It is also worth noting that the conditions covered by the second level of exceptions (the cases in which the White King is in between the White Rook and Black King) occur less than 4 times in every 1000 examples. This is only picked up in the noise-free data set of 10000 examples (in which there were 38 examples where the rules applied).

Extending the PAC analogy further, Figure 16 shows the results from a different perspective. For different levels of noise, this figure shows the number of training examples required for the "approximately correct" theory of Figure 14 to be compressive. For example, at least 170 examples are required to obtain a

	Training Set Size					
Noise (%)	100	250	500	1000	5000	10000
0	_	99.7	99.7	99.7	99.7	100
5	_	98.1	98.1	99.7	99.7	99.7
10	_	_	98.1	98.1	99.7	99.7
15	_	_	98.1	98.1	99.7	99.7
20	_	_	_	98.1	99.7	99.7
30	_	_	_	_	98.1	98.1
40	_	_	_	_	_	98.1

Figure 15: Test-set accuracy for the "most compressive" theory

compressive theory that is 98% accurate on noise-free data. While these numbers are approximate (they are obtained by extrapolating the compression produced by the theory for the different training sets in Figure 15) they do indicate the general trend of requiring larger example sets for increasing noise levels.

8 Conclusion

In this paper we have developed a general encoding scheme for deciding the significance of first-order hypotheses by refining the approach found in (Muggleton 1988). The requirement to encode both hypotheses and proofs results in some unique advantages

1. The resulting compression measure appears to be the first significance measure that accounts for the relevance and utility of background knowledge. This issue has been avoided to date by relational learning systems.

2. The measure appears to reliably distinguish noisy data by finding them to be incompressible with the background knowledge.

3. The encoding incorporates notions of efficiency in the same units (bits) as the program description. Michie (1977) discusses in detail the time-space tradeoff for encoded knowledge using the standard Machine Intelligence problem of perfect play in chess. Within Machine Learning various methods including EBL (Mitchell *et al.* 1986) have been developed to learn the time-space tradeoff. EBL systems take a set of axioms and add redundant theorems in order to reduce proofs lengths for specific types of problems. These systems do not contain any sound criterion of whether theorems being added are significant. A method of estimating the utility of adding these theorems is described in (Subramanian and Hunter 1992). The method does

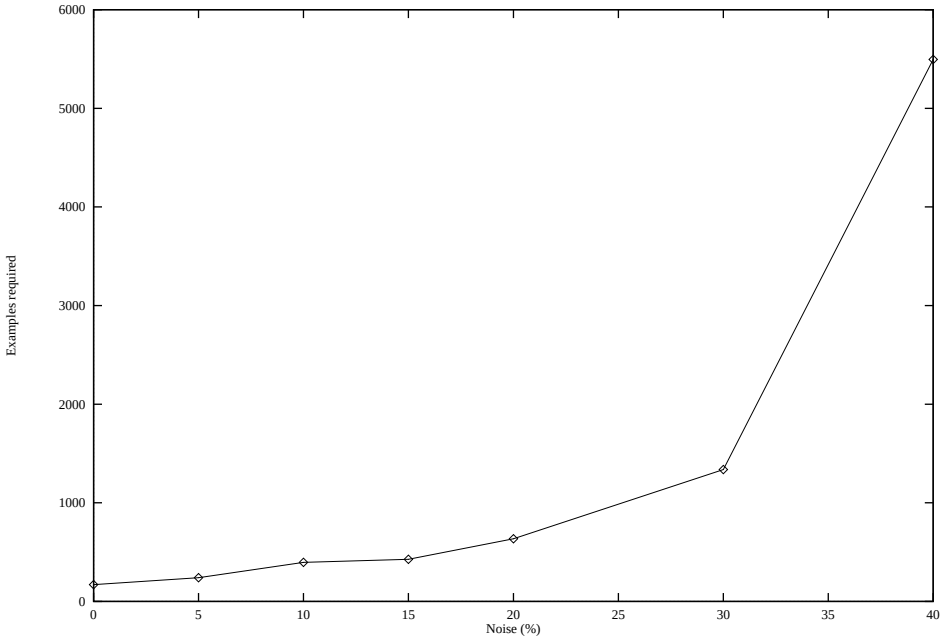

Figure 16: Examples required for a 98% correct and compressive theory

not appear to directly account for the space dimension. The compression model described in this paper should be able to provide a criterion that accounts for this as well.

The question of how well hypotheses perform on unseen data was dealt with first by Gold (1967) and more recently within the PAC (Probably-Approximately-Correct) framework (Valiant 1984). The Gold and PAC frameworks describe the conditions under which a class of concepts can be said to be learnable. In the PAC framework it is explicitly assumed that the distribution of examples in the training and test sets are the same. Blumer *et al.* (1986), Board and Pitt (1989) and Li and Vitanyi (1989) have in various ways shown that a class of concepts is PAC-learnable if and only if it can be guaranteed that a learning algorithm is able to find a hypothesis which is smaller than the data. It remains to be shown that our concept of hypothesis size (that is, hypothesis and proofs) is equivalent to that adopted in these theoretical results.

The task of distinguishing between exceptions and noise is an issue that is typically ignored in the literature on non-monotonic reasoning. It is, however, of fundamental importance for a learning program that has to construct theories using real-world data. One way to approach the problem is to see if the exceptions to the current theory exhibit a pattern. The compression model we have used in this paper does precisely this. Our empirical results suggest that by selecting the most compressive theory, it is possible (given enough data) to reliably avoid fitting most of the noise. Clearly, it would be desirable to confirm these results

with controlled experiments in other domains. In practice, the method has found interesting rules on an independent problem of pharmaceutical drug design (King *et al.* 1992). Finally, the results also lend support to the link between compressive theories for first-order concepts and their PAC-learnability.

Acknowledgements. The authors would like to thank Donald Michie and the ILP group at the Turing Institute for their helpful discussions and advice. This work was carried out at the Turing Institute and was supported by the Esprit Basic Research Action project ECOLES, the IED's Temporal Databases and Planning project and the SERC Rule-Base Systems Project. Stephen Muggleton is supported by an SERC post-doctoral fellowship.

References

[1] Angluin, D. and Laird, P. (1988). Learning from noisy examples. *Machine Learning*, 2(4):343–370.

[2] Bain, M. (1991). Experiments in non-monotonic learning. In *Proceedings of the Eighth International Workshop on Machine Learning*, pages 380–384, San Mateo, CA. Morgan Kaufmann.

[3] Bain, M. and Muggleton, S. (1991). Non-monotonic learning. In Michie, D., editor, *Machine Intelligence 12*. Oxford University Press.

[4] Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. (1986). Classifying learnable geometric concepts with the Vapnik-Chervonenkis dimension. In *Proceedings of the 18th ACM Symposium on Theory of Computing*, pages 273–282.

[5] Board, R. and Pitt, L. (1989). On the necessity of occam algorithms. Uiucdcs-r-89-1544, University of Illinois at Urbana-Champaign.

[6] Brunk, C. and Pazzani, M. (1991). An investigation of noise-tolerant relational concept learning algorithms. In Birnbaum, L. and Collins, G., editors, *Proceedings of the Eighth International Workshop on Machine Learning*, San Mateo. Morgan Kaufmann.

[7] Carnap, R. (1952). *The Continuum of Inductive Methods*. Chicago University, Chicago.

[8] Cestnik, B. (1991). Estimating probabilities: A crucial task in machine learning. In *Proceedings of the European Conference on Aritficial Intelligence*, Stockholm, Sweden.

[9] Chaitin, G. (1987). *Information, Randomness and Incompleteness - Papers on Algorithmic Information Theory*. World Scientific Press, Singapore.

[10] Clark, P. and Niblett, T. (1989). The CN2 algorithm. *Machine Learning*, 3(4):261–283.

[11] Cussens, J. (1992). Estimating rule accuracies from training data. In *ECAI Workshop on Logical Approaches to Learning*.

[12] DeRaedt, L. and Bruynooghe, M. (1992). Interactive concept-learning and constructive induction by analogy. *Machine Learning*, 8(2):107–150.

[13] Dzeroski, S. (1991). *Handling Noise in Inductive Logic Programming*. University of Ljubljana, (M.Sc. Thesis), Ljubljana.

[14] Gallager, R. G. (1968). *Information theory and Reliable Communication*. Wiley, New York.

[15] Gold, E. (1967). Language identification in the limit. *Information and Control*, 10:447–474.

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

[17] King, R. and Sternberg, M. (1990). A machine learning approach for the prediction of protein secondary structure. *Journal of Molecular Biology*, 216:441–457.

[18] Kolmogorov, A. (1965). Three approaches to the quantitative definition of information. *Prob. Inf. Trans.*, 1:1–7.

[19] Li, M. and Vitanyi, P. (1989). Inductive reasoning and Kolmogorov complexity. In *Proceedings of the Fourth Annual IEEE Structure in Complexity Theory Conference*, pages 165–185.

[20] McCarthy, J. (1986). Applications of circumscription to formalizing common sense knowledge. *Artificial Intelligence*, 28:89–116.

[21] Michalski, R. (1983). A theory and methodology of inductive learning. In Michalski, R., Carbonnel, J., and Mitchell, T., editors, *Machine Learning: An Artificial Intelligence Approach*, pages 83–134. Tioga, Palo Alto, CA.

[22] Michie, D. (1977). A theory of advice. In Elcock, E. and Michie, D., editors, *Machine Intelligence 8*, pages 151–168. Horwood.

[23] Mitchell, T., Keller, R., and Kedar-Cabelli, S. (1986). Explanation-based generalization: A unifying view. *Machine Learning*, 1(1):47–80.

[24] Morales, E. (1991). Learning features by experimentation in chess. In Kodratoff, Y., editor, *EWSL '91*, pages 494–511, Berlin. Springer-Verlag.

[25] Mortimer, H. (1988). *The Logic of Induction*. Ellis Horwood, Chichester, England.

[26] Muggleton, S. (1988). A strategy for constructing new predicates in first order logic. In *Proceedings of the Third European Working Session on Learning*, pages 123–130. Pitman.

[27] Muggleton, S. and Feng, C. (1990). Efficient induction of logic programs. In *Proceedings of the First Conference on Algorithmic Learning Theory*, Tokyo. Ohmsha.

[28] Muggleton, S., King, R., and Sternberg, M. (1991). Predicting protein secondary structure using inductive logic programming. *submitted to Proteins*.

[29] Plotkin, G. (1971). *Automatic Methods of Inductive Inference*. PhD thesis, Edinburgh University.

[30]

[31] Popper, K. (1972). *Conjectures and Refutations: The Growth of Scientific Knowledge*. Routledge and Kegan Paul, London.

[32] Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, 1:81–106.

[33] Quinlan, J. (1990). Learning logical definitions from relations. *Machine Learning*, 5:239–266.

[34] Quinlan, J. and Rivest, R. (1989). Inferring decision trees using the Minimum Description Length principle. *Information and Computation*, 80:227–248.

[35] Rissanen, J. (1978). Modeling by Shortest Data Description. *Automatica*, 14:465–471.

[36] Rissanen, J. (1982). A universal prior for integers and estimation by Minimum Description Length. *Annals of Statistics*, 11:416–431.

[37] Rouveirol, C. (1991). Itou: Induction of first-order theories. In *First International Workshop on Inductive Logic Programming*, Porto, Portugal.

[38] Shannon, C. and Weaver, W. (1963). *The Mathematical Theory of Communication*. University of Illinois Press, Urbana.

[39] Shapiro, E. (1983). *Algorithmic program debugging*. MIT Press.

[40] Solomonoff, R. (1964). A formal theory of inductive inference. *Information and Control*, 7:376–388.

[41] Subramanian, D. and Hunter, S. (1992). Measuring utility and the design of provably good ebl algorithms. *Ninth International Machine Learning Conference*.

[42] Valiant, L. (1984). A theory of the learnable. *Communications of the ACM*, 27:1134–1142.