Bayesian Inductive Logic Programming

Stephen Muggleton
Oxford University Computing Laboratory
Wolfson Building, Parks Road
Oxford, OX1 3QD.

Abstract

Inductive Logic Programming (ILP) involves the construction of first-order definite clause theories from examples and background knowledge. Unlike both traditional Machine Learning and Computational Learning Theory, ILP is based on lockstep development of Theory, Implementations and Applications. ILP systems have successful applications in the learning of structure-activity rules for drug design, semantic grammars rules, finite element mesh design rules and rules for prediction of protein structure and mutagenic molecules. The strong applications in ILP can be contrasted with relatively weak PAC-learning results (even highly-restricted forms of logic programs are known to be prediction-hard). It has been recently argued that the mismatch is due to distributional assumptions made in application domains. These assumptions can be modelled as a Bayesian prior probability representing subjective degrees of belief. Other authors have argued for the use of Bayesian prior distributions for reasons different to those here, though this has not lead to a new model of polynomial-time learnability. Incorporation of Bayesian prior distributions over time-bounded hypotheses in PAC leads to a new model called U-learnability. It is argued that U-learnability is more appropriate than PAC for Universal (Turing computable) languages. Time-bounded logic programs have been shown to be polynomially U-learnable under certain distributions. The use of time-bounded hypotheses enforces decidability and allows a unified characterisation of speed-up learning and inductive learning. U-learnability has as special cases PAC and Natarajan's model of speed-up learning.

COLT 94 - 7/94 New Brunswick, N..J. USA

1 INTRODUCTION

This paper is written for two separate audiences: 1) a Machine Learning audience, interested in implementations and applications and 2) a Computational Learning Theory audience interested in models and results in learnability. The paper should thus be taken as two loosely connected papers with a common theme involving the use of distributional information in Inductive Logic Programming (ILP).

In the last few years ILP [23, 24, 20, 31] has developed from a theoretical backwater to a rapidly growing area in Machine Learning. This is in part due to the fact that pure logic programs provide Machine Learning with a representation which is general purpose (Turing-computable), and has a simple, and rigorously defined, semantics. In addition, logic programs tend to be easy to comprehend as hypotheses in scientific domains.

ILP is based on interaction between Theory, Implementations and Applications. The author has argued [25] that the development of a formal semantics of ILP (see Section 2) allows the direct derivation of ILP algorithms. Such derivational techniques have been at the centre of specific-to-general ILP techniques such as Plotkin's least general generalisation [37, 36], Muggleton and Buntine's inverse resolution [27, 23] and Idestam-Almquist's inverse implication [15]. Two contending semantics of ILP have been developed [31]. These are the so-called 'open-world' semantics [31] and 'closed-world' semantics [13].

A number of efficient ILP systems have been developed, including FOIL [38], Golem [28], LINUS [21], CLAUDIEN [39] and Progol [26]. The use of a relational logic formalism has allowed successful application of ILP systems in a number of domains in which the concepts to be learned cannot easily be described in an attribute-value, propositional-level, language. These applications (see Section 3) include structure activity prediction for drug design [19, 44, 43], protein secondary-structure prediction [29], finite-element mesh design [9] and learning semantic grammars [45].

A variety of positive and negative PAC-learnability results exist for subsets of definite clause logic [11, 34, 10, 18, 7, 40]. However, in contrast to experimentally demonstrated abilities of ILP systems in applications, the positive PAC results are rather weak, and even highly restricted forms of logic

programs have been shown to be prediction hard [7]. Like many other results in PAC-learning, positive results are only achieved for ILP by setting language parameters to constant values (eg. k-clauses and l-literals). This provides 'hard' boundaries to the hypothesis space, which often reduces the representation to a propositional level. An alternative model, U-learnability [30], is better suited to Universal (Turing computable) representations, such as logic programs. U-learnability provides a 'soft' boundary to hypothesis spaces in the form of a prior probability distribution over the complete representation (eg. time-bounded logic programs). Other authors have argued for the use of Bayesian prior distributions for reasons different to those here, though this has not lead to a new model of polynomial-time learnability. U-learnability allows standard distributional assumptions, such as Occam's razor, to be used in a natural manner, without parameter settings. Although Occam algorithms have been studied in PAC-learning [3], this has not led to algorithms capable of learning in universal representations, as it has in Inductive Logic Programming. In the early 1980's a distribution assumption very different to Occam's razor was implicitly implemented by Arbab, Michie [1] and Bratko [4] in an algorithm for constructing decision trees. The algorithm constructs a decision list (linear decision tree) where one exists and otherwise returns the most linear decision tree which can be constructed from the data. This distributional assumption is based on the fact that decision lists are generally easier to comprehend than arbitrary decision trees. Other interesting kinds of distributions along these lines can be imagined; assigning higher prior probabilities to grammars that are regular or *almost* so, logic programs that are deterministic or *almost* so and logic programs that run in linear time or *almost* so. One can imagine very different kinds of prior distributions on hypotheses. For example, when learning concepts which mimic human performance in skill tasks [41] predictive accuracy is dependent on hypotheses being evaluable in time similar to that of human reaction, and so such hypotheses should be preferred *a priori*.

This paper is organised as follows. Section 2 gives a formal definition of ILP in terms of Bayesian inference. Section 3 provides an overview of ILP applications in molecular biology and discusses some of the distributional assumptions used in these applications. Section 4 defines U-learnability and describes some general results on U-learnable distributions.

2 BAYES' ILP DEFINITIONS

Familiarity with standard definitions from Logic Programming [22, 14] is assumed in the following. A Bayesian version of the usual (open world semantics) setting for ILP is as follows. Suppose $\mathcal{P}$, $\mathcal{F}$ and $\mathcal{V}$ are sets of predicate symbols, function symbols and variables and $\mathcal{C}_d$ and $\mathcal{C}_h$ are the classes of definite and Horn clauses constructed from $\mathcal{P}$, $\mathcal{F}$ and $\mathcal{V}$. The symbol $\vdash_n$ denotes SLDNF derivation bounded by n resolution steps. An ILP learner is provided with background knowledge $B \subseteq \mathcal{C}_d$ and examples $E = \langle E^+, E^- \rangle$ in which $E^+ \subseteq \mathcal{C}_d$ are positive examples and $E^- \subseteq (\mathcal{C}_h - \mathcal{C}_d)$ are negative examples. Each hypothesis is a pair $H_n = \langle H, n \rangle$ where $H \subseteq \mathcal{C}_d$, $H \not\models B$ and n is a a natural number. H_n is said to hold for examples E, or $holds(H_n, E)$, when both the following are true[1].

Sufficiency: $H \vdash_n E^+$

Satisfiability: $H \wedge E^- \not\vdash_n \square$

The prior probability, $p(H_n)$, of an hypothesis is defined by a given distribution D. According to Bayes' theorem, the posterior probability is

$$p(H_n|E) = \frac{p(E|H_n).p(H_n)}{p(E)}$$

where $p(E|H_n)$ is 1 when $holds(H_n, E)$ and 0 otherwise and

$$p(E) = \sum_{holds(H'_n, E)} p(H'_n)$$

Well known strategies for making use of distributional information include a) maximisation of posterior probability b) class prediction based on posterior weighted sum of predictions of all hypotheses (Bayes optimal strategy) c) randomly sampling an hypothesis from the posterior probability (Gibbs algorithm). The sample complexities of strategies b) and c) are analysed in [12].

Although no existing ILP system takes an hypothesis distribution D as an input parameter, such distributions are implicit in FOIL's [38] information gain measure and Golem's [32] compression measure. Additionally, search strategies such as general-to-specific or specific-to-general, use an implicit prior distribution which favours more general hypotheses or, conversely, more specific ones.

Bayesian approaches to Machine Learning have been discussed previously in the literature. For instance, Buntine [6] develops a Bayesian framework for learning, which he applies to the problem of learning class probability trees. Also Haussler et al. [12] analyse the sample complexity of two Bayesian algorithms. This analysis focuses on average case, rather than worst case, accuracy. On the other hand U-learnability uses average case time complexity and worst case accuracy. Also unlike U-learnability (Section 4) neither Buntine nor Haussler et al. develop a new learning model which allows significantly larger polynomially-learnable representations, such as logic programs. The applications in the following section show that ILP systems *are* effective at learning logic programs in significant real-world domains. This is consistent with the fact that time-bounded logic programs are U-learnable under certain distributions (see Section 4).

3 APPLICATIONS IN MOLECULAR BIOLOGY

ILP applications in Molecular Biology extend a long tradition in approaches to scientific discovery that was initiated by Meta-Dendral's [5] application in mass spectrometry. Molecular biology domains are particularly appropriate for ILP due

[1] Though most ILP systems can deal with noise, this is ignored in the above definition for the sake of simplicity.

to the rich relational structure of the data. Such relationships make it hard to code feature-based representations for these domains. ILP's applications to protein secondary structure prediction [29], structure-activity prediction of drugs [19] and mutagenicity [42] of molecules have produced new knowledge, published in respected scientific journals in their area.

3.1 PROTEIN SECONDARY STRUCTURE PREDICTION

When a protein (amino acid sequence) shears from the RNA that codes it, the molecule convolves in a complex though deterministic manner. However, it is largely the final shape, known as secondary and tertiary structure, which determines the protein's function. Determination of the shape is a long and labour intensive procedure involving X-ray crystallography. On the other hand the complete amino acid sequence is relatively easy to determine and is known for a large number of naturally occurring proteins. It is therefore of great interest to be able to predict a protein's secondary and tertiary structure directly from its amino acid sequence.

Many attempts at secondary structure prediction have been made using hand-coded rules, statistical and pattern matching algorithms and neural networks. However, the best results for the overall prediction rate are still disappointing (65-70%). In [29] the ILP system Golem was applied to learning secondary structure prediction rules for the sub-domain of α/α proteins. The input to the program consisted of 12 non-homologous proteins (1612 amino acid residues) of known secondary structure, together with background knowledge describing the chemical and physical properties of the naturally occurring amino acids. Golem learned a set of 21 rules that predict which amino acids are part of the α-helix based on the positional relationships and chemical and physical properties. The rules were tested on four independent non-homologous proteins (416 amino acid residues) giving an accuracy of 81% (with 2% expected error). The best previously reported result in the literature was 76%, achieved using a neural net approach. Although higher accuracy is possible on sub-domains such as α/α than is possible in the general domain, ILP has a clear advantage in terms of comprehensibility over neural network and statistical techniques.

However, according to the domain expert, Mike Sternberg, Golem's secondary-structure prediction rules were hard to comprehend since they tended to be overly specific and highly complex. In many scientific domains, comprehensibility of new knowledge has higher priority than accuracy. Such a priori preferences can be modelled by a prior distribution on hypotheses.

3.2 DRUG STRUCTURE-ACTIVITY PREDICTION

Proteins are large macro-molecules involving thousands of atoms. Drugs tend to be small, rigid molecules, involving tens of atoms. Drugs work by stimulating or inhibiting the activity of proteins. They do so by binding to specific sites on the protein, often in competition with natural regulatory molecules, such as hormones. The 3-dimensional structure of the protein binding site is not known in over 90% of all

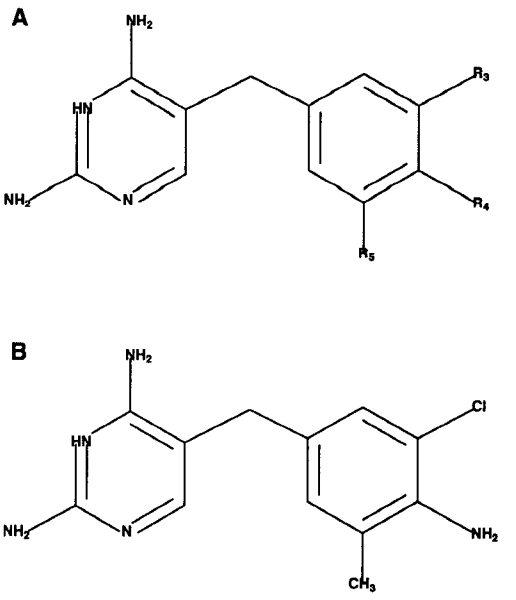

Figure 1: The family of analogues in the first ILP study. A) Template of 2,4-diamino-5(substituted-benzyl)pyrimidines R3, R4, and R5 are the three possible substitution positions. B) Example compound: $3-Cl, 4-NH_2, 5-CH_3$

drug development projects. However, the *in vitro* activity of members of a family of drugs can be determined experimentally in the laboratory. This data is usually analysed by drug designers using statistical techniques, such as regression, to predict the activity of untested members of the family of drugs. This in turn leads to directed testing of the drugs predicted to have high activity.

An application of ILP to structure-activity prediction was reported in [19]. The ILP program Golem [28] was applied to the problem of modelling the structure-activity relationships of trimethoprim analogues binding to dihydrofolate reductase. The training data consisted of 44 trimethoprim analogues and their observed inhibition of *Escherichia coli* dihydrofolate reductase. A further 11 compounds were used as unseen test data. Golem obtained rules that were statistically more accurate on the training data and also better on the test data than a previously published linear regression model. Figure 1 illustrates the family of analogues used in the study.

Owing to the lack of variation in the molecules, a highly deterministic subset of the class of logic programs was sufficient to learn in this domain. Unlike the protein domain, the domain expert found the rules to be highly compact and thus easy to comprehend.

3.3 MUTAGENESIS PREDICTION

In a third molecular biology study ILP has been applied to predicting mutagenicity of molecules [42]. Mutagenicity is highly correlated to carcinogenicity. Unlike activity of a drug, mutagenicity is a property which pharmaceutical companies would like to avoid. Also, unlike drug development projects, the data on mutagenicity forms a highly heteroge-

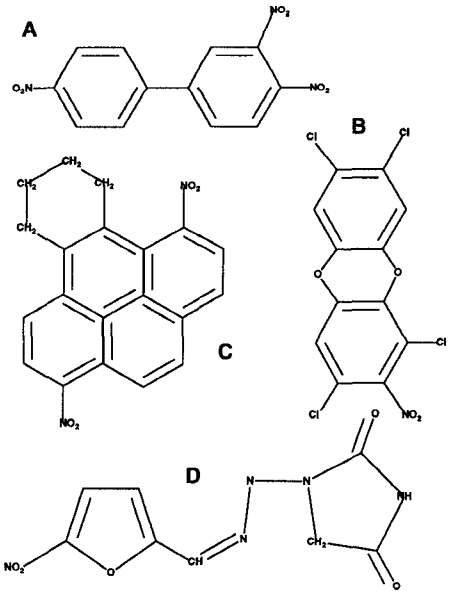

Figure 2: Examples of the diverse set of aromatic and heteroaromatic nitro compounds used in the mutagenesis study. A) 3,4,4'-trinitrobiphenyl B) 2-nitro-1,3,7,8-tetrachlorodibenzo-1,4-dioxin C) 1,6,-dinitro-9,10,11,12-tetrahydrobenzo[e]pyrene D) nitrofurantoin

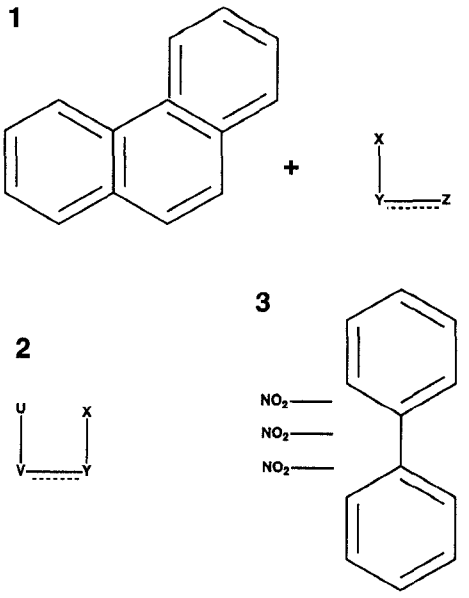

Figure 3: Structural properties discovered by Progol

neous set with large variations in molecular form, shape and charge distribution. The data comes from a wide number of sources, and the molecules involved are almost arbitrary in shape and bond connectivity (see Figure 2).

The 2D bond-and-atom molecular descriptions of 229 aromatic and heteroaromatic nitro compounds taken from [8] were given to the ILP system Progol [26]. The study was confined to the problem of obtaining structural descriptions that discriminate molecules with positive mutagenicity from those which have zero or negative mutagenicity.

A set of 8 compact rules were discovered by Progol. These rules suggested 3 previously unknown features leading to mutagenicity. Figure 3 shows the structural properties described by the rules in the Progol theory. The descriptions of these patterns are as follows. 1) Almost half of all mutagenic molecules contain 3 fused benzyl rings. Mutagenicity is enhanced if such a molecule also contains a pair of atoms connected by a single bond where one of the atoms is connected to a third atom by an aromatic bond. 2) Another strong mutagenic indicator is 2 pairs of atoms connected by a single bond, where the 2 pairs are connected by an aromatic bond as shown. 3) The third mutagenic indicator discovered by Progol was an aromatic carbon with a partial charge of +0.191 in which the molecule has three NO_2 groups substituted onto a biphenyl template (two benzine rings connected by a single bond).

An interesting feature is that in the original regression analysis of the molecules by [8], a special 'indicator variable' was provided to flag the existence of three or more fused rings (this variable was then seen to play a vital role in the regression equation). The first rule (pattern 1 in Figure 3), expressing precisely this fact, was discovered by Progol without access to any specialist chemical knowledge. Further, the regression analysis suggested that electron-attracting elements conjugated with nitro groups enhance mutagenicity. A particular form of this pattern was also discovered by Progol (pattern 3 in Figure 3).

Previous published results for the same data [8] used linear discriminant techniques and hand-crafted features. The ILP technique allowed prediction of 49 cases which were statistical outliers not predictable by the linear discriminant algorithm. In addition, the ILP rules are small and simple enough to provide insight into the molecular basis of mutagenicity. Such a low-level representation opens up a range of arbitrary chemical structures for analysis. This is of particular interest for drug discovery, since all compounds within a database of compounds have known atom and bond descriptions, while very few have computed features available.

Owing to the large range of molecules the rules were highly non-deterministic (of the form "there exists somewhere in molecule M a structure S with property P"). Owing to simplicity, the mutagenicty rules were the most highly favoured by the expert among the Molecular Biology domains studied. For all the domains in Molecular Biology comprehensibility, rather than accuracy, provides the strongest incentive for an Occam distribution (ie. shorter theories preferred a priori).

4 U-LEARNABILITY

The learnability model defined in this section allows for the incorporation of a priori distributions over hypotheses. The definition of this model, U-learnability, is motivated by the general problems associated with learning Universal (Turing computable) representations, such as logic programs. Nevertheless U-learnability can be easily applied to other representations, such as decision trees. U-learnability is defined using

the traditional notation from computational learning theory rather than that typically used in ILP (see Section 2).

4.1 BACKGROUND FOR U-LEARNING

Let $(\Sigma_E, X, \Sigma_C, R, c)$ be the representation of a learning problem (as in PAC-learning), where X is the domain of examples (finite strings over the alphabet Σ_E), R is the class of concept representations (finite strings over the alphabet Σ_C), and $c : R \rightarrow 2^X$ maps concept representations to concepts (subsets of X). Hence the concept class being learned is the actual range of c (a subset of 2^X). Let P be a subset of the class of polynomial functions in one variable. From this point on, a concept representation will be a pair (r, p), for $r \in R$ and $p \in P$.

Let $A(r, p, x)$ be an algorithm that maps any tuple (r, p, x), for $r \in R$, $p \in P$, and $x \in X$, to 0 or 1. Let A run in time bounded by $q(|r|, p(|x|))$, where q is a polynomial in two variables. Furthermore, let A have the following properties. First, for all $r \in R$ and $x \in X$, if $x \notin c(r)$ then for every $p \in P$: $A(r, p, x) = 0$. Second, for all $r \in R$ and $x \in X$, if $x \in c(r)$ then there exists $p \in P$ such that for every $p' \in P$ with $p'(|x|) \geq p(|x|)$: $A(r, p', x) = 1$. Intuitively, p specifies some form of time bound that is polynomially related to the size of the example. It might specify the maximum derivation length (number of resolution steps) for logic programs or the maximum number of steps in a Turing Machine computation. Greater derivation lengths or more computation steps are allowed for larger inputs.

Let F be any family of probability distributions over $R \times P$, where the distributions in F have an associated parameter $n \geq 0$. Let G be any family of probability distributions over X.

4.2 PROTOCOL FOR U-LEARNING

In U-learning, a Teacher randomly chooses a pair (r, p), for $r \in R$ and $p \in P$, according to some distribution D_1 in F. The Teacher presents to a learning algorithm L an infinite stream of labeled examples $\langle (x_1, l_1), (x_2, l_2), ... \rangle$ where: each example x_i is drawn randomly, independently of the preceding examples, from X according to some fixed, unknown distribution D_2 in G and labeled by $l_i = A(r, p, x_i)$. After each example x_i in the stream of examples, L outputs a hypothesis $H_i = (r_i, p_i)$, where $r_i \in R$ and $p_i \in P$.

4.3 DEFINITION OF U-LEARNABILITY

Definition 1 Polynomial U-learnability. *Let F be a family of distributions (with associated parameters) over $R \times P$, and let G be a family of distributions over X. The pair (F,G) is* polynomially U-learnable *just if there exist polynomial functions $p_1(y) = y^c$, for some constant c, $p_2(y)$, $p_3(y_1, y_2)$, and a learning algorithm L, such that for every distribution D_1 (with parameter n) in F, and every distribution D_2 in G, the following hold.*

- *The average-case time complexity of L at any point in a run is bounded by $p_1(M)$, where M is the sum of $q(|r|, p(|x_i|))$ over the examples x_i seen to that point.*[2] *(Recall that (r, p) is the target concept chosen randomly according to D_1, and that q describes the time complexity of the evaluation algorithm A.)*
- *For all $m \geq p_2(n)$, there exist values ϵ and δ, $0 < \epsilon, \delta < 1$, such that: $p_3(\frac{1}{\epsilon}, \frac{1}{\delta}) \geq m$, and with probability at least $1 - \delta$ the hypothesis $H_m = (r_m, p_m)$ is $(1 - \epsilon)$-accurate. By $(1 - \epsilon)$-accurate, we mean that the probability (according to D_2) that $A(r_m, p_m, x_{m+1}) \neq l_{m+1}$ is less than ϵ.*

As with PAC-learning, we can also consider a *prediction* variant, in which the hypotheses H_m need not be built from R (in addition to P), but can come from a different class R' and can have a different associated evaluation algorithm A'.

4.4 INITIAL RESULTS AND PROPOSED DIRECTIONS

This section presents initial results about polynomial U-learnability and suggests directions for further work. To conserve space the proofs are omitted, but they appear in a full paper on U-learnability [30]. In each of these results, let $(\Sigma_E, X, \Sigma_C, R, c)$ be the representation of any given learning problem. The following theorem shows that PAC-learnability is a *special case* of U-learnability.

Theorem 2 U-learnability generalises PAC. *Let p be the polynomial function $p(x) = x$. Let F be a family of distributions that contains one distribution D_i for each $r_i \in R$, such that D_i assigns probability 1 to (r_i, p). Let $n = 0$ be the parameter for each member of F. Let G be the family of all distributions over X. Then the representation $(\Sigma_E, X, \Sigma_C, R, c)$ is PAC-learnable (PAC-predictable) if and only if (F, G) is polynomially U-learnable (U-predictable).*

This observation raises the question of how, exactly, polynomial U-learnability differs from PAC-learnability. There is a cosmetic difference that polynomial U-learnability has been defined to look more similar to *identification in the limit* than does the definition of PAC-learnability. Ignoring this leaves the following distinguishing features of polynomial U-learnability.

[2] To be more precise, let m be any positive integer. Let $Pr_{D_1}(r, p)$ be the probability assigned to (r, p) by D_1, and (with a slight abuse of notation) let $Pr_{D_1,D_2}(r, p, \langle x_1, ..., x_m \rangle)$ be the product of $Pr_{D_1}(r, p)$ and the probability of obtaining the sequence $\langle x_1, ..., x_m \rangle$ when drawing m examples randomly and independently according to D_2. Let $\text{TIME}_L(r, p, \langle x_1, ..., x_m \rangle)$ be the time spent by L until output of its hypothesis H_m, when the target is (r, p) and the initial sequence of m examples is $\langle x_1, ..., x_m \rangle$. Then we say that the average-case time complexity of L to output H_m is bounded by $p_1(y) = y^c$ just if the sum (or integral), over all tuples $(r, p, \langle x_1, ..., x_m \rangle)$, of

$$[Pr_{D_1,D_2}(r, p, \langle x_1, ..., x_m \rangle)] \frac{(\text{TIME}_L(r, p, \langle x_1, ..., x_m \rangle))^{\frac{1}{c}}}{M}$$

is less than infinity, where M is the sum of $q(|r|, p(|x_i|))$ over all x_i in $x_1, ..., x_m$. See [2] for the motivation of this definition of average-case complexity.

- Assumption of a prior distribution on target concept representations.
- Sample size is related to ϵ, δ, and (by means of the parameter n) the *reciprocal of the probability of the target representation* rather than the target representation size.
- Use of average case time complexity, relative to the prior distribution on hypotheses and the distribution on examples, in place of worst case time complexity.
- Concept representations need not be polynomially evaluable in the traditional sense, but only in the weaker sense that some polynomial p must exist for each concept representation such that the classification of an example x can be computed in time $p(|x|)$. This distinction and the next are irrelevant for concept representations that are already polynomially evaluable and hence do not need time bounds.
- The learning algorithm runs in (expected) time polynomial in the (worst-case) time required to compute the classification, by the target, of the examples seen thus far, rather than in the sum of the sizes of the examples seen thus far.

Definition 3 The distribution family D_P. *Let R be countable, let $p'(y)$ be a polynomial function, and let $E_{p'} = \langle E_1, E_2, ... \rangle$ be an enumeration of subsets of R such that: (1) for all $i \geq 1$, the cardinality of E_i is at most $p'(i)$, (2) every $r \in R$ is in some E_i, and (3) the members of E_i, for each i, can be determined efficiently (by an algorithm that runs in time polynomial in the sum of the sizes of the members of E_i). Let F' be the family of all distributions D', with finite variance, over the positive integers, and let the parameter n' of D' be the maximum of the mean and standard deviation of D'. For each such distribution D' in F', define a corresponding distribution D'' over R as follows: for each $i \geq 1$, let the probabilities of the $r \in E_i$ be uniformly distributed with sum $Pr_{D'}(i)$. Let the parameter associated with D'' also be n'. Let P be the set of all linear functions of the form $p(x) = cx$, where c is a positive integer. Let D'_L be any probability distribution over the positive integers that has a probability density function (p.d.f.) $Pr_{D'_L}(x) = \frac{1}{x^k}$, for some $k > 3$ (appropriately normalised so that it sums to 1). For each distribution D'_L, define a distribution D_L over P to assign to the function $p(x) = cx$ the probability assigned to c by D'_L. Finally, for each pair of distributions D'' and D_L as defined above, define a distribution D over $R \times P$ as follows: for each pair (r, p), where $r \in R$ and $p \in P$, $Pr_D(r, p) = [Pr_{D''}(r)][Pr_{D_L}(p)]$. Let the parameter n of D be n'. D_P is the family of all such distributions D, each with associated parameter n.*

Theorem 4 Polynomial U-learnability under D_P. *Let F be the distribution family D_P defined by a particular enumeration of subsets of R. Let G be any family of distributions over X. The pair (F, G) is polynomially U-learnable.*

The following example relates U-learnability to the ILP definitions given in Section 2.

Example 5 Time-bounded logic programs are U-learnable under D_P. *Let R be the set of all logic programs that can be built from a given alphabet of predicate symbols $\mathcal{P}$, function symbols $\mathcal{F}$, and variables $\mathcal{V}$. Notice that R is countable. Let P be the set of all bounds $p(x)$ on derivation length of the form $p(x) = cx$, where c is a positive integer and x is the size of (number of terms in) the goal. Let the domain X of examples be the ground atomic subset of $\mathcal{C}_d$, which is the set of definite clauses that can be built from the given alphabet. Notice also that a concept (r, p) classifies as positive just the definite clauses $d \in X$ such that $r \vdash_{p(|d|)} d$, where $|d|$ is the number of terms in d. Let F be the family of distributions D_P built using some particular enumeration of polynomially-growing subsets of R, and let G be the family of all distributions over examples. From Theorem 4, it follows that (F, G) is polynomially U-learnable.*

Definition 6 The distribution family D_{E_k}. *Let R be countable, let k be a positive integer, and let $E_e = \langle E_1, E_2, ... \rangle$ be an enumeration of subsets of R such that: (1) for all $i \geq 1$, the cardinality of E_i is at most k^i, (2) every (r, p), for $r \in R$ and $p \in P$, is in some E_i, and (3) the members of E_i, for each i, can be determined efficiently (by an algorithm that runs in time polynomial in the sum of the sizes of the members of E_i). Let D'_k be the discrete exponential distribution with probability density function*

$$Pr(x) = (k-1)k^{-x} \text{ for all integers } x \geq 1$$

and let the parameter n' of D'_k be its mean ($\frac{k}{k-1}$). From D'_k, define a corresponding distribution D_k over R as follows: for each $i \geq 1$, let the probabilities of the $r \in E_i$ be uniformly distributed with sum $Pr_{D'_k}(i)$. Let the parameter associated with D_k also be n'. Let P be the set of all linear functions of the form $p(x) = cx$, where c is a positive integer. Let the distributions D_L over P be as defined earlier (see definition of D_P). Finally, for the distribution D_k (with parameter n') and each distribution D_L, define a distribution D over $R \times P$ as follows: for each pair (r, p), for $r \in R$ and $p \in P$, $Pr_D(r, p) = [Pr_{D_k}(r)][Pr_{D_L}(p)]$. Let the parameter n of D be the parameter n' of D_k. D_{E_k} is the family of distributions consisting of each such distribution D.

Theorem 7 Polynomial U-learnability under D_{E_k}. *Let F be the distribution family D_{E_k} defined by a particular enumeration of subsets of R and a particular choice of k. Let G be any family of distributions over X. The pair of distributions (F, G) is polynomially U-learnable.*

Example 8 Time-bounded logic programs are U-learnable under D_{E_k}. *Again let R be the set of all logic programs that can be built from a given finite alphabet of predicate symbols $\mathcal{P}$, function symbols $\mathcal{F}$, and variables $\mathcal{V}$. And again let P be the set of all bounds $p(x)$ on derivation length of the form $p(x) = cx$, where c is a positive integer and x is the size of (number of terms in) the goal. Let the domain X of examples be the ground atomic subset of $\mathcal{C}_d$. Let $\langle S_1, S_2, ... \rangle$ be an enumeration of subsets of R such that S_i contains all logic programs of length i, for all $i \geq 1$. Let F be the distribution family D_{E_k} built from this enumeration and from the positive integer k chosen to be the sum of the sizes of $\mathcal{P}$, $\mathcal{F}$, and $\mathcal{V}$. Let G be the family of all distributions over examples. From Theorem 7, it follows that (F, G) is polynomially U-learnable.*

In many cases it may be incorrect to assume that the probabilities of target representations vanish as quickly as they do in the distributions considered in the preceding theorems. In particular, a more appropriate distribution on target representations may be defined by combining an enumeration E_e, as in Definition 6, with some distribution in which the probabilities do not decrease exponentially, as in Definition 3. Polynomial U-learnability should be considered with such distributions, perhaps combined with restricted families of distributions on examples. It can be expected that obtaining positive results for "natural" such distributions will be theoretically challenging, and that such results will have a major impact on practical applications of machine learning. It is also expected that it will be challenging and useful to obtain negative results for some such distributions. Whereas negative results for PAC-learnability can be obtained by showing that the *consistency* problem for a given concept representation is NP-hard (assuming $RP \neq NP$) [35], negative results for polynomial U-learnability in some cases can be obtained by showing that the consistency problem is *NP-hard-on-average* (or *DistNP hard*) [2, 16] relative to particular distributions on hypotheses and examples. In addition, just as negative results for PAC-predictability can be obtained (based on certain assumptions) using hard problems from cryptography [17], negative results for polynomial U-predictability might possibly be obtained in this same way, but would require additional effort. Specifically, obtaining negative results for polynomial U-predictability will require specifying a particular distribution (or family of distributions) to be used with a hard problem from cryptography, such as inverting RSA, and it must be verified, in some way, that the problem does not become substantially easier under this distribution (or every distribution in this family). For example, inverting RSA is assumed hard if encryption/decryption is based on a choice of two large random prime numbers according to a *uniform* distribution over primes of some large size, but it becomes easy if the distribution over primes of a given size assigns probability 1 to one particular prime number; what about for distributions "in between", where some primes of a given size are more likely than others? The distributions used in polynomial U-predictability would correspond to such distributions for RSA.

4.5 INCORPORATING BACKGROUND KNOWLEDGE

Recall that background knowledge is often used in ILP. This paper has not provided for the use of background knowledge in polynomial U-learnability. The full paper on U-learnability includes a definition of polynomial U-learnability that allows the teacher to provide the learner with a background theory. A surprising aspect of this definition is that it not only captures the notion of inductive learning relative to background knowledge, as desired, but it also provides a generalisation of Natarajan's PAC-style formalisation of speed-up learning [33]. Although Natarajan's formalisation is quite appealing, few positive have been proven within it because it is also very demanding. The use of a family of prior distributions on hypotheses, as specified in the definition of polynomial U-learnability, can in some cases effectively ease these demands and lead to additional positive results.

5 DISCUSSION

As the ILP applications in Section 3 show, not only is Horn clause logic a powerful representation language for scientific concepts, but also one which is tractable in real-world domains. This apparently flies in the face of negative PAC-learnability results for logic programs. This may be explained by the fact that despite logic programs being used as the representation language throughout the applications in Section 3, very different distributional assumptions were necessary in each application.

ILP has a clear model-theoretic semantics which is inherited from Logic Programming. However, without consideration of the probabilistic nature of inductive inference, a purely logic-based definition of ILP is always incomplete. For this reason the Bayes' oriented definitions of ILP found in Section 2 provide a more complete semantics for ILP than has previously been suggested. This introduces prior distributional information into the definition of ILP. The introduction of such information is also motivated both by the requirements of practical experience and the ability to define a model of learnability which is more appropriate for learning logic programs.

All ILP algorithms make implicit use of distributional information. One promising line of research would involve explicit provision of distributions to general-purpose ILP systems.

The choice of representation is at the centre of focus in PAC-learning. By contrast, the choice of distribution is the central theme in U-learnability. The reason for the refocusing of attention is that once one is committed to a Universal representation, as in the case of ILP, differences in distributional assumptions become paramount.

Some general initial results of U-learnability are stated in this paper. Much more effort is required to clearly define the boundaries of what is and is not U-learnable. It is believed that U-learnability will not only be of interest to researchers in ILP but also to the wider community within Machine Learning.

Acknowledgements

The author would like to thank David Page and Ashwin Srinivasan of Oxford University Computing Laboratory and Michael Sternberg and Ross King of Imperial Cancer Research Fund for discussions and input to this paper. This work was supported partly by the Esprit Basic Research Action ILP (project 6020), the SERC project GR/J46623 and an SERC Advanced Research Fellowship held by the author. The author is also supported by a Research Fellowship at Wolfson College, Oxford.

References

[1] B. Arbab and D. Michie. Generating rules from examples. In *IJCAI-85*, pages 631–633, Los Altos, CA, 1985. Kaufmann.

[2] S. Ben-David, B. Chor, and O. Goldreich. On the theory of average case complexity. *Journal of Information and System Sciences*, 44:193–219, 1992.

[3] R. Board and L. Pitt. On the necessity of Occam algorithms. UIUCDCS-R-89-1544, University of Illinois at Urbana-Champaign, 1989.

[4] I. Bratko. Generating human-understandable decision rules. Working paper, E. Kardelj University Ljubljana, Ljubljana, Yugoslavia, 1983.

[5] B. Buchanan, E. Feigenbaum, and N. Sridharan. Heuristic theory formation: data interpretation and rule formation. In B. Meltzer and D. Michie, editors, *Machine intelligence 7*, pages 267–290. Edinburgh University Press, 1972.

[6] W. Buntine. *A Theory of Learning Classification Rules*. PhD thesis, School of Computing Science, University of Technology, Sydney, 1990.

[7] W. Cohen. Learnability of restricted logic programs. In S. Muggleton, editor, *Proceedings of the 3rd International Workshop on Inductive Logic Programming (Technical report IJS-DP-6707 of the Josef Stefan Institute, Ljubljana, Slovenia)*, pages 41–72, 1993.

[8] 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.

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

[10] S. Dzeroski, S. Muggleton, and S. Russell. PAC-learnability of determinate logic programs. In *Proceedings of the 5th ACM Workshop on Computational Learning Theory*, Pittsburg, PA, 1992.

[11] D. Haussler. Applying Valiant's learning framework to AI concept-learning problems. In Y. Kodratoff and R. Michalski, editors, *Machine learning: an artificial intelligence approach*, volume 3, pages 641–669. Morgan Kaufman, San Mateo, CA, 1990.

[12] D. Haussler, M Kearns, and R. Shapire. Bounds on the sample complexity of bayesian learning using information theory and the vc dimension. In *COLT-91: Proceedings of the 4th Annual Workshop on Computational Learning Theory*, pages 61–74, San Mateo, CA, 1991. Morgan Kauffmann.

[13] N. Helft. Induction as nonmonotonic inference. In *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning*, pages 149–156. Kaufmann, 1989.

[14] C.J. Hogger. *Essentials of logic programming*. Oxford University Press, Oxford, 1990.

[15] P. Idestam-Almquist. *Generalisation of Clauses*. PhD thesis, Stockholm University, 1993.

[16] D. Johnson. The NP-completeness column—an ongoing guide. *Journal of Algorithms*, 4:284–299, 1984.

[17] M. Kearns and L. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. In *Proceedings of the 21st ACM Symposium on Theory of Computing*, pages 433–444. ACM, 1989.

[18] J.U Kietz. Some lower bounds on the computational complexity of inductive logic programming. In P. Brazdil, editor, *Proceedings of the 6th European Conference on Machine Learning*, volume 667 of *Lecture Notes in Artificial Intelligence*, pages 115–123. Springer-Verlag, 1993.

[19] R. King, S. Muggleton R. Lewis, and M. 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. *Proceedings of the National Academy of Sciences*, 89(23), 1992.

[20] N. Lavrač and S. Džeroski. *Inductive Logic Programming : Techniques and Applications*. Ellis Horwood, 1993.

[21] N. Lavrač, S. Džeroski, and M. Grobelnik. Learning non-recursive definitions of relations with LINUS. In Yves Kodratoff, editor, *Proceedings of the 5th European Working Session on Learning, volume 482 of Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1991.

[22] J.W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, Berlin, 1984.

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

[24] S. Muggleton, editor. *Inductive Logic Programming*. Academic Press, 1992.

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

[26] S. Muggleton. Mode-directed inverse resolution. In K. Furukawa, D. Michie, and S. Muggleton, editors, *Machine Intelligence 14*. Oxford University Press, (to appear).

[27] S. Muggleton and W. Buntine. Machine invention of first-order predicates by inverting resolution. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 339–352. Kaufmann, 1988.

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

[29] S. Muggleton, R. King, and M. Sternberg. Protein secondary structure prediction using logic-based machine learning. *Protein Engineering*, 5(7):647–657, 1992.

[30] S. Muggleton and D. Page. A learnability model for universal representations. Technical Report PRG-TR-3-94, Oxford University Computing Laboratory, Oxford, 1994.

[31] S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 12, 1994. (to appear).

[32] S. Muggleton, A. Srinivasan, and M. Bain. Compression, significance and accuracy. In *Proceedings of the Ninth International Machine Learning Conference*, San Mateo, CA, 1992. Morgan-Kaufmann.

[33] B. K. Natarajan. Learning from exercises. In *Proceedings of the 1989 Workshop on Computational Learning Theory*, pages 72–86, San Mateo, CA, 1989. Morgan Kaufmann.

[34] D. Page and A. Frisch. Generalization and learnability: A study of constrained atoms. In S. Muggleton, editor, *Inductive Logic Programming*. Academic Press, London, 1992.

[35] L. Pitt and L. G. Valiant. Computational limitations on learning from examples. *Journal of the ACM*, 35(4):965–984, 1988.

[36] G. Plotkin. A further note on inductive generalization. In *Machine Intelligence*, volume 6. Edinburgh University Press, 1971.

[37] G.D. Plotkin. A note on inductive generalisation. In B. Meltzer and D. Michie, editors, *Machine Intelligence 5*, pages 153–163. Elsevier North Holland, New York, 1970.

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

[39] L. De Raedt and M. Bruynooghe. A theory of clausal discovery. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 1993.

[40] L. De Raedt and S. Dzeroski. First-order jk-clause theories are pac-learnable. Technical report, Department of Computer Science, Katholieke Universieit Leuven, Heverlee, Belgium, 1994.

[41] C. Sammut, S. Hurst, D. Kedzier, and D. Michie. Learning to fly. In D. Sleeman and P. Edwards, editors, *Proceedings of the Ninth International Workshop on Machine Learning*, pages 385–393, San Mateo, CA, 1992. Morgan Kaufmann.

[42] A. Srinivasan and S. Muggleton. Discovering rules for mutagenesis. Technical Report PRG-TR-4-94, Oxford University Computing Laboratory, Oxford, 1994.

[43] M. Sternberg, R. King, R. Lewis, and S. Muggleton. Application of machine learning to structural molecular biology. *Philosophical Transactions of the Royal Society B*, 1994 (to appear).

[44] M. Sternberg, R. Lewis, R. King, and S. Muggleton. Modelling the structure and function of enzymes by machine learning. *Proceedings of the Royal Society of Chemistry: Faraday Discussions*, 93:269–280, 1992.

[45] J. Zelle and R. Mooney. Learning semantic grammars with constructive inductive logic programming. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 817–822, San Mateo, CA, 1993. Morgan Kaufmann.