Statistical Aspects of Logic-Based Machine Learning

STEPHEN H. MUGGLETON
University of York.

Logic Programming (LP) is a subject which has influenced many of the activities of Computer Science. The subtopic of LP concerned with Machine Learning is known as "Inductive Logic Programming" (ILP). ILP involves the automatic construction of logic programs from examples and background knowledge. Most of the theory of ILP has been concerned with logical aspects of hypothesis formation. In contrast this paper surveys a growing body of literature concerned with probabilistic and statistical aspects of ILP. A Bayesian statistical framework is provided, within which probabilities are used to compare the degrees of belief of competing hypotheses. This approach has been used in the general learning framework of U-learnability. The paper surveys results of applying Bayesian analysis to the problem of learning from positive examples as well as that of learning in the presence of incomplete background knowledge and noise. More recently, probabilities have been used directly within the representation of first-order hypotheses. A survey of a number of such probabilistic representations is provided together with initial results on approaches to machine learning of such representations.

Categories and Subject Descriptors: [**Computational Logic**]: Logic and Machine Learning

General Terms: Inductive Logic Programming, Machine Learning
Additional Key Words and Phrases: U-learning, Computational Learning Theory, Probabilistic Logics

1. INTRODUCTION

The term "Inductive Logic Programming" (ILP) [Muggleton 1991] is now ten years old. ILP is a general form of Machine Learning which involves the construction of logic programs from examples and background knowledge. The subject has inherited its logical tradition from Logic Programming and its empirical orientation from Machine Learning. Robert Kowalski [Kowalski 1980] famously described Logic Programming (LP) with the following equation.

$$\text{LP} = \text{Logic} + \text{Control}$$

The equation emphasises the role of the programmer in providing sequencing control when writing Prolog programs. In a similar spirit we might describe ILP as follows.

$$\text{ILP} = \text{Logic} + \text{Statistics} + \text{Computational Control}$$

The logical part of ILP is related to the formation of hypotheses while the statistical part is related to evaluating their degrees of belief. As in LP the Computational

Author's address: S.H. Muggleton, Department of Computer Science, University of York, Heslington, York, YO10 5DD, United Kingdom.

Control part is related to the sequencing of search carried out when exploring the space of hypotheses.

While most descriptions of ILP have concentrated on either its logical foundations [Muggleton and Raedt 1994; Nienhuys-Cheng and de Wolf 1997] or its applications [Muggleton 1999a; 1999b], the present paper provides an overview of the state of various statistical aspects of the area. Three different applications of probability theory and statistics have emerged within ILP. These involve the use of

(1) probabilistic preference functions over competing hypotheses (Bayesian Inductive Logic Programming),
(2) probability theory to reason about the convergence of ILP algorithms (Computational Learning Theory and U-learnability) and
(3) probabilistic logics to explicitly represent degrees of uncertainty within examples, background knowledge and hypotheses (Learning Probabilistic Logical Representations).

Though previously distinct, these three statistical aspects of ILP are starting to merge. For instance, U-learning (Section 2.2) involves aspects 1 and 2, while the research described in Section 5.2 combines aspects 1 and 3. However, to date no approach combines all three.

The paper is organised as follows. Section 2 introduces and motivates Bayesian Learning Frameworks and Bayesian Inductive Logic Programming. The Bayesian framework of U-learnability is then described in Section 2.2, together with some general results concerning U-learnable families. Sometimes sample complexity analyses allow the use of mathematical techniques to design efficient algorithms. This is the case for the results given in Section 3.1 involving learning from positive-only examples and in Section 3.2 for learning from incomplete background knowledge. Section 4 discusses approaches to incorporating probabilistic information within logical representations. Various approaches are discussed including Probabilistic Relational Models (PRMs) and Stochastic Logic Programs (SLPs). Recent approaches to learning PRMs and SLPs are briefly reviewed in Section 5. The paper is summarised and future research directions are discussed in Section 6.

2. BAYESIAN LEARNING FRAMEWORKS

A variety of positive and negative PAC-learnability results exist for subsets of definite clause logic [Haussler 1990; Page and Frisch 1992; Džeroski et al. 1992; Kietz 1993; Cohen 1993; Raedt and Džeroski 1994; Cohen and Page 1995]. However, in contrast to experimentally demonstrated abilities of ILP systems in applications (see [Muggleton 1999a; 1999b]), the positive PAC results are rather weak, and even highly restricted forms of logic programs have been shown to be prediction hard [Cohen 1993; Nock and Jappy 1998]. 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 [Muggleton 1994; Muggleton and Page 1994], 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 (eg. [Haussler et al. 1991]) have argued for the use of Bayesian prior distributions, 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 [Board and Pitt 1989], 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 [Arbab and Michie 1985] and Bratko [Bratko 1983] 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 [Sammut et al. 1992] 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*.

2.1 Bayesian Inductive Logic Programming

Familiarity with standard definitions from Logic Programming [Lloyd 1987; Hogger 1990] 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 $H \subseteq \mathcal{C}_d$, $H \models B$ 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: . $H \wedge E^- \nvdash_n \square$

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

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

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

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 [Haussler et al. 1991].

Although no existing ILP system takes an hypothesis distribution D as an input parameter, such distributions are implicit in FOIL's [Quinlan 1990] information gain measure as well as Golem's [Muggleton et al. 1992] and Progol's [Muggleton 1995] 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 are discussed elsewhere in the literature. For instance, Buntine [Buntine 1990] develops a Bayesian framework for learning, which he applies to the problem of learning class probability trees. Also Haussler et al. [Haussler et al. 1991] analyse the sample complexity of two Bayesian algorithms (the sample complexity of a learning algorithm relates the number of training examples to the out-of-sample accuracy). 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 2.2) 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 [Muggleton 1999a; 1999b] 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 2.2).

2.2 U-learning

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 2.1).

2.2.1 *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.

2.2.2 *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 $p_i \in P$.

2.2.3 *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'.

[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 $TIME_L(r,p,\langle x_1,...,x_m\rangle)$ 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 [Ben-David et al. 1992] for the motivation of this definition of average-case complexity.

2.2.4 *U-learnability results.* This section presents results about polynomial U-learnability and suggests directions for further work. To conserve space the proofs are omitted, but they appear in [Muggleton and Page 1994]. 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).*

The polynomial distribution family D_P is now introduced.

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 2.1.

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.

3. SAMPLE COMPLEXITY RESULTS

U-learnability is based on bounding both the sample complexity and the time complexity of a learning algorithm. It is often simplest to start the analysis by consideration of sample complexity alone. The cases in this section show that the use of mathematical techniques to minimise sample complexity sometimes leads naturally to the resultant algorithms having polynomial time-complexity. This section summarises results which first appeared in [Muggleton 2000a; Parson et al. 1999]. Proofs from the original papers are omitted due to lack of space.

3.1 Learning from positive-only examples

The following is a simplified version of the U-learnability framework presented in Section 2.2. X is taken to be a countable class of instances and $\mathcal{H} \subseteq 2^X$ to be a countable class of concepts. D_X and $D_{\mathcal{H}}$ are probability distributions over X and $\mathcal{H}$ respectively. For $H \in \mathcal{H}$, $D_X(H) = \sum_{x \in H} D_X(x)$ and the conditional distribution of D_X associated with H is as follows.

$$D_{X|H}(x) = D_X(x|H) = \frac{D_X(x \cap H)}{D_X(H)} = \begin{cases} 0 & \text{if } x \notin H \\ \frac{D_X(x)}{D_X(H)} & \text{otherwise} \end{cases}$$

The teacher randomly chooses a target theory T from $D_{\mathcal{H}}$ and randomly and independently chooses a series of examples $E = \langle x_1, .., x_m \rangle$ from T according to $D_{X|T}$. Given E, $D_{\mathcal{H}}$ and D_X a learner L chooses an hypothesis $H \in \mathcal{H}$ for which all elements of E are in H. The teacher then assesses $\text{Error}(H)$ as $D_X(H \backslash T) + D_X(T \backslash H)$.

Unlike the setting in U-learnability it is assumed in the present paper that L is given $D_{\mathcal{H}}$ and D_X.

3.1.1 *Bayes' posterior estimation.* Gold's negative result [Gold 1967] for identification of the regular languages over the symbol set Σ is based on the fact that for any sequence of positive examples E there will always be at least two possible candidate hypotheses, 1) Σ^*, the language containing all possible sentences and 2) the language corresponding to elements of E alone. It is clear that 1) is the most general hypothesis, and has a compact finite automaton description, while 2) is the least general hypothesis and has a complex finite state automaton description. Since these two extreme hypotheses will maximise errors of commission and omission respectively it would seem desirable to find a compromise between the size of the hypothesis description and its generality. Size and generality of an hypothesis can be defined within the Bayes' framework of the previous section as follows.

$$\begin{aligned} \text{sz}(H) &= -\ln D_{\mathcal{H}}(H) \\ \text{g}(H) &= D_X(H) \end{aligned}$$

Bayes' theorem allows us to derive a tradeoff between sz(H) and g(H). In its familiar form, Bayes' theorem is as follows.

$$p(H|E) = \frac{p(H)p(E|H)}{p(E)}$$

With respect to the Bayes' framework of the previous section $p(H|E)$ is interpreted from the learner's perspective as the probability that $H = T$ given the example sequence is E. Similarly, $p(H)$ is defined as the probability that $H = T$, which is

$$p(H) = D_{\mathcal{H}}(H).$$

Meanwhile $p(E|H)$ is the probability that the example sequence is E given that $H = T$. Since examples are chosen randomly and independently from $D_{X|H}$ then for any consistent hypothesis this is as follows.

$$\begin{aligned} p(E|H) &= \prod_{i=1}^{m} D_{X|H}(x_i) \\ &= \prod_{i=1}^{m} \frac{D_X(x_i)}{D_X(H)} \end{aligned}$$

The prior $p(E)$ is the probability that the example sequence is E irrespective of T. This is as follows.

$$p(E) = \sum_{T \in \mathcal{H}} D_{\mathcal{H}}(T) \prod_{j=1}^{m} D_{X|T}(x_j)$$

The Bayes' equation can now be rewritten as follows.

$$p(H|E) = \frac{D_{\mathcal{H}}(H) \prod_{i=1}^{m} \frac{D_X(x_i)}{D_X(H)}}{p(E)}$$

Since $\frac{\prod_{i=1}^{m} D_X(x_i)}{p(E)}$ is common to all consistent hypotheses, it will be treated as a normalising constant c_m in the following.

$$\begin{aligned} p(H|E) &= D_{\mathcal{H}}(H) \left(\frac{1}{D_X(H)}\right)^m c_m \\ \ln p(H|E) &= m \ln \left(\frac{1}{g(H)}\right) - sz(H) + d_m \end{aligned}$$

In the above $d_m = \ln c_m$. The tradeoff between size and generality of an hypothesis can be seen in the final equation above. The function $\ln p(H|E)$ decreases with increases in $sz(H)$ and $g(H)$. Additionally, as m grows, the requirements on generality of an hypothesis become stricter. A function f_m with similar properties was defined in [Muggleton 1995] and it was shown there that for every hypothesis H except T there is a value of m such that for all $m' > m$ it is the case that $f_{m'}(H) < f_m(T)$. This result indicates a form of convergence, somewhat different from Gold's identification in the limit.

3.1.2 *Analysis of expected error.* Haussler et al. [Haussler et al. 1994] argue the advantages of analysing expected error over VC dimension analysis. Analysis of expected error is the approach taken below.

It is assumed that class membership of instances is decidable for all hypotheses. Also the hypotheses in $\mathcal{H}$ are assumed to be ordered according to decreasing prior probability as $H_1, H_2, \ldots$. For the purposes of analysis the distribution $D_{\mathcal{H}}(H_i) = \frac{a}{i^2}$ is assumed, where a is a normalising constant. This is similar to the prior distribution assumptions used in Progol4.1 [Muggleton 1995] and is a smoothed version of a distribution which assigns equal probability to the 2^b hypotheses describable in b bits, where the sum of the probabilities of such hypotheses is 2^{-b}. Within this distribution i has infinite mean and variance. It is also assumed that the hypothesis space contains only targets T for which $D_X(T) \leq \frac{1}{2}$. This assumption, which holds for most target concepts used in Inductive Logic Programming, is not a particularly strong restriction on the hypothesis space since if $\overline{T}$ is the complement of T and $D_X(T) > \frac{1}{2}$ then clearly $D_X(\overline{T}) \leq \frac{1}{2}$.

The following theorem gives an upper bound on the expected error of an algorithm which learns from positive examples only by maximising the Bayes' posterior probability function over the initial am hypotheses within the space.

THEOREM 6. **Expected error for positive examples only.** *Let X be a countable instance space and D_X be a probability distribution over X. Let $\mathcal{H} \subseteq 2^X$ be a countable hypothesis space containing at least all finite subsets of X, and for which all $H \in \mathcal{H}$ have $D_X(H) \leq \frac{1}{2}$. Let $D_{\mathcal{H}}$ be a probability distribution over $\mathcal{H}$. Assume that $\mathcal{H}$ has an ordering $H_1, H_2, \ldots$ such that $D_{\mathcal{H}}(H_i) \geq D_{\mathcal{H}}(H_j)$ for all $j > i$. Let $D_{\mathcal{H}}(H_i) = \frac{a}{i^2}$ where $\frac{1}{a} = \sum_{i=1}^{\infty} \frac{1}{i^2} \approx \frac{1}{0.608}$. Let $\mathcal{H}_n = \{H_i : H_i \in \mathcal{H} \text{ and } i \leq n\}$. T is chosen randomly from $D_{\mathcal{H}}$ and the x_i in $E = \langle x_1, .., x_m \rangle$ are chosen randomly and independently from $D_{X|T}$. Let $f_m(H) = D_{\mathcal{H}}(H)(\frac{1}{D_X(H)})^m$ and let $n = am$. L is the following learning algorithm. If there are no hypotheses $H \in \mathcal{H}_n$ such that $H \supseteq H_E = \{x_1, .., x_m\}$ then $L(E) = H_E$. Otherwise $L(E) = H_n(E) = H$ only if $H \in \mathcal{H}_n$, $H \supseteq H_E$ and for all $H' \in \mathcal{H}_n$ for which $H' \supseteq H_E$ it is the case that $f_m(H) \geq f_m(H')$. The error of an hypothesis H is defined as $Error(H,T) = D_X(T \setminus H) + D_X(H \setminus T)$. The expected error of L after m examples, $EE(m)$, is at most $\frac{2.33+2\ln m}{m}$.*

Note that this result is independent of the choice of D_X and that L considers only $O(m)$ hypotheses to achieve an expected error of $O(\frac{\ln m}{m})$. For comparison a similar algorithm which learns from a mixture of positive and negative examples has a corresponding bound of $EE(m) \leq \frac{1.51+2\ln m}{m}$, which is within a small additive term of the bound for learning from positive examples only.

3.2 Learning with incomplete background knowledge

The results in the previous section assume the learner knows the teacher's prior. In ILP this corresponds to a situation in which complete and correct background knowledge has been provided. We now consider the case in which background knowledge is incomplete.

We consider an extension of the model in Section 3.1. Previously it was assumed that the learner knew precisely the teacher's distribution $D_{\mathcal{H}}$ over hypotheses. Clearly there is no reason why this should hold in practical machine learning

situations. We now relax this assumption and consider what happens to the expected error of learning when the learner does not know the exact form of the teacher's prior. In particular we consider an incorrect prior over hypotheses induced by an incomplete background theory.

3.2.1 *The Modified Model.* The learner's background theory B_L is assumed to be missing certain predicate definitions $P = B_T \setminus B_L$ which occur in the target predicate. The teacher's prior is now $D_{\mathcal{T}}$ and the learner's (incorrect) prior is $D_{\mathcal{L}}$.

The hypotheses in $\langle$ are ordered by the teacher according to decreasing prior probability $D_{\mathcal{T}}(H_i) = \frac{a}{i^2}$ as $H_1, H_2, \ldots, H_i, \ldots$. The learner only has partial information about this ordering in that its prior is a corrupted version of the teachers distribution. In particular, those hypotheses $\mathcal{H}_P \subseteq \mathcal{H}$ which reference predicates in P are assigned different information content under the two distributions. This results in $H \in \mathcal{H}_P$ having differing indices H_i and H_j in the orderings of $D_{\mathcal{T}}$ and $D_{\mathcal{L}}$. In fact if we assume that the information content of the missing predicates P is at most k bits then for any hypothesis $H_i \in \mathcal{H}$:

$$\begin{aligned} k &\leq \left|H^{E(B_L)}\right| - \left|H^{E(B_T)}\right| \\ &= -\log_2 D_{\mathcal{L}}(H_j) + \log_2 D_{\mathcal{T}}(H_i) \\ &= -\log_2 \frac{a}{j^2} + \log_2 \frac{a}{i^2} \\ &= 2\log_2 j - 2\log_2 i \\ &= 2\log_2 \frac{j}{i} \end{aligned}$$

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

3.2.2 *Expected Error*

THEOREM 7. *Let X be a countable instance space and $\mathcal{H} \subseteq 2^X$ be a countable hypothesis space containing at least all finite subsets of X. Let $D_{\mathcal{T}}$, $D_{\mathcal{L}}$ be probability distributions over $\mathcal{H}$. Let D_X be a probability distribution over X. Assume that $\mathcal{H}$ has an ordering $H_1, H_2, \ldots, H_i, \ldots$ such that $D_{\mathcal{T}}(H_i) \geq D_{\mathcal{T}}(H_{i+1})$ for all i and an ordering $H'_1, H'_2, \ldots, H'_j, \ldots$ such that $D_{\mathcal{L}}(H'_j) \geq D_{\mathcal{L}}(H'_{j+1})$ for all j. Let $D_{\mathcal{T}}(H_i) = \frac{a}{i^2}$ where $\frac{1}{a} = \sum_{i=1}^{\infty} \frac{1}{i^2} = \pi^2/6$. Let $\mathcal{H}_n = \{H_i : H_i \in \mathcal{H}$ and $i \leq n$. T is chosen randomly from $D_{\mathcal{T}}$. Let $ex(x, H) = \langle x, v\rangle$ where $v =$ True if $x \in H$ and $v =$ False otherwise. Let $E = \langle ex(x_1, T), \ldots, ex(x_m, T)\rangle$ where each x_i is chosen randomly and independently from D_X. $H_E = \{x : \langle x, True\rangle \in E\}$. Hypothesis H is said to be consistent with E if and only if $x_i \in H$ for each $\langle x_i, True\rangle$ in E and $x_j \notin H$ for each $\langle x_j, False\rangle$ in E. Let $n = am$. L is the following learning algorithm. If there are no hypotheses $H \in \mathcal{H}_n$ consistent with E then $L(E) = H_E$. Otherwise $L(E) = H_n(E) = H$ only if $H \in \mathcal{H}_n$, H consistent with E and for all $H' \in \mathcal{H}_n$ consistent with E it is the case that $D_{\mathcal{L}}(H) \geq D_{\mathcal{L}}(H')$. The error of an hypothesis H is defined as $Error(H, T) = D_X(T \setminus H) + D_X(H \setminus T)$. The expected error of L after m examples, $EE(m)$, is at most:*

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

3.3 Noise handling

In [McCreath and Sharma 1997] McCreath and Sharma consider a variant of the positive-only model (Section 3.1) in which a degree of noise is assumed. However, they did not develop sample complexity results for this model.

4. PROBABILISTIC REPRESENTATIONS

Previous sections have concentrated on the use of probabilistic preference functions and probability theory as a means of reasoning about convergence of learning algorithms. However, there are good reasons to explicitly represent probabilities in the inputs and outputs of ILP systems. Since the output of inductive inference has only a limited degree of certainty, it is preferable that degrees of belief are directly represented as part of the hypotheses. These can then be taken into account in any further inductive and deductive reasoning. This implies that probabilities should be allowed in both hypotheses and background knowledge. The inevitability of noisy data implies that probabilities should also be allowed in the representation of individual examples. This argument implies the need for the use of probabilistic first-order logics within ILP.

Efforts to combine logic and probability can be traced back at least as far as George Boole [Boole 1854]. However, Carnap [Carnap 1962] was the first to attempt to do so at the level of first-order logic. In the 1970's and 1980's further efforts along these lines were encouraged by the success of expert systems such as MYCIN [Shortliffe and Buchanan 1975] and Prospector [Duda et al. 1979]. Bayesian networks [Pearl 1988] can be viewed as a sound approach to doing so at a propositional logic level. However, arguably Halpern [Halpern 1990] was the first to develop a sound computational semantics for the combination of first-order logic and probability. Subsequently a number of efficient representations have been developed [Poole 1993; Ng and Subrahmanian 1992; Koller and Pfeffer 1998; Muggleton 1996] which are compatible with Halpern's semantics. These approaches are briefly surveyed below.

4.1 Halpern's analysis

The following two approaches for a semantics of first-order logics of probability are considered by Halpern [Halpern 1990].

(1) Probabilities are placed on objects in the domain. This is appropriate for modeling sentences such as "The probability that a randomly chosen bird flies is greater than 0.9".

(2) Probabilities are placed on possible worlds. This is appropriate for modeling the sentence "The probability that Tweety flies (a particular bird) is greater than 0.9".

Halpern shows that these two approaches can be combined and provides an axiomatic system which is sound and complete for a substantial class of probabilistic statements.

4.2 Probabilistic Horn abduction

In [Poole 1993] Poole presents a framework for Horn-clause abduction, with probabilities being associated with hypotheses. It should be noted that within an abduc-

tive framework such hypotheses are associated with individual predictions, rather than with universally quantified beliefs, as they are in ILP.

Poole shows that the probabilistic dependencies in a Bayesian belief network can be represented in his framework. Moreover, the framework allows Bayesian networks to be extended beyond a propositional language.

4.3 Probabilistic Logic Programs

In [Ng and Subrahmanian 1992] Ng and Subrahmanian have developed Probabilistic Logic Programs (PLPs) where literals are annotated with probability intervals. The approach is consistent with a subset of the semantics developed by Halpern. A fixed probabilistic strategy is used which assumes independence of events. Evaluation is carried out using linear programming. Later in [Dekhtyar and Subrahmanian 2000] Dekhtyar and Subrahmanian consider hybrid probabilistic programs in which the user can select a set of probabilistic strategies which define conjunctive and disjunctive connectives.

4.4 Probabilistic Relational Models

Koller and Pfeffer [Koller and Pfeffer 1998] recently introduced Probabilistic Relational Models (PRMs) as an alternative representation for efficiently combining first-order logic and probability theory. Whereas Poole and Subrahmanian et al. approach the problem from the direction of extending logical representations to deal with probabilistic inference, Koller and Pfeffer's approach aims at lifting Bayesian networks by the addition of logical relations. The end result lays greater emphasis on the network of relations and less on the underlying logical sentences.

4.5 Stochastic Logic Programs

Whereas Probabilistic Horn abduction and PRMs were directly motivated by an attempt to generalise Bayesian belief networks, Stochastic Logic Programs (SLPs) [Muggleton 1996] were originally developed as generalisations of Hidden Markov Models (HMMs) [Rabiner 1989] and Stochastic Context Free Grammars (SCFGs) [Lari and Young 1990]. The author viewed them as a compact approach to representing a probabilistic preference function to provide as a parameter to ILP algorithms.

HMMs and SCFGs have been extremely successful in sequence-oriented applications in Natural Language and Bioinformatics. They provide a compact representation of a probability distribution over sequences. This contrasts with Bayesian networks, which represent conditional independences between a set of propositions. It is natural to think of HMMs and SCFGs as representing probabilities over objects in the domain (Halpern's type 1 approach from Section 4.1) and Bayesian networks as representing probabilities over possible worlds (Halpern's type 2 approach). However, Cussens [Cussens 1999] has shown that directed Bayesian networks are also special cases of SLPs.

4.5.1 *Stochastic automata.* Stochastic automata, otherwise called Hidden Markov Models [Rabiner 1989], have found many applications in speech recognition. An example is shown in Figure 1. Stochastic automata are defined by a 5-tuple $A = \langle Q, \Sigma, q_0, F, \delta \rangle$. Q is a set of states. Σ is an alphabet of symbols. q_0 is

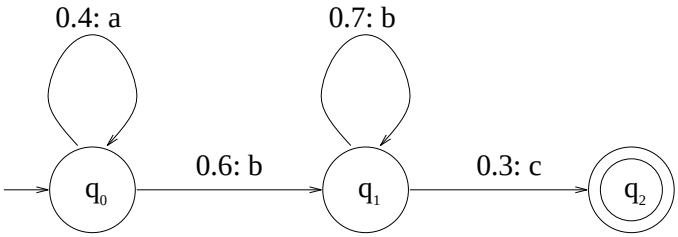

Fig. 1. Stochastic automaton.

$$
\begin{aligned}
&0.4 : q_0 \to aq_0\\
&0.6 : q_0 \to bq_1\\
&\\
&0.7 : q_1 \to bq_1\\
&0.3 : q_1 \to cq_2\\
&\\
&1.0 : q_2 \to \lambda
\end{aligned}
$$

Fig. 2. Labelled production rule representation of stochastic automaton.

the initial state and $F \subseteq Q$ ($F = \{q_2\}$ in Figure 1) is the set of final states. $\delta : (Q \setminus F) \times \Sigma \to Q \times [0,1]$ is a stochastic transition function which associates probabilities with labelled transitions between states. The sum of probabilities associated with transitions from any state $q \in (Q \setminus F)$ is 1.

In the following λ represents the empty string. The transition function $\delta^* : (Q \setminus F) \times \Sigma^* \to Q \times [0,1]$ is defined as follows. $\delta^*(q,\lambda) = \langle q, 1\rangle$. $\delta^*(q,au) = \langle q_{au}, p_a p_u\rangle$ if and only if $\delta(q,a) = \langle q_a, p_a\rangle$ and $\delta^*(q_a,u) = \langle q_{au}, p_u\rangle$. The probability of u being accepted from state q in A is defined as follows. $Pr(u|q,A) = p$ if $\delta^*(q,u) = \langle q', p\rangle$ and $q' \in F$. $Pr(u|q,A) = 0$ otherwise.

The following example illustrates the calculation of probabilities of strings.

EXAMPLE 8. **Probabilities associated with strings.** *For the automaton A in Figure 1 we have $Pr(abbc|A) = 0.4 \times 0.6 \times 0.7 \times 0.3 = 0.0504$. $Pr(abac|A) = 0$.*

A can also be viewed as expressing a probability distribution over the language $L(A) = \{u : \delta^*(q_0,u) = \langle q,p\rangle \text{ and } q \in F\}$.

4.5.2 *Labelled productions.* Stochastic automata can be equivalently represented as a set of labelled production rules. Each state in the automaton is represented by a non-terminal symbol and each δ transition $\langle q,a\rangle \to \langle q',p\rangle$ is represented by a production rule of the form $p : q \to aq'$. Figure 2 is the set of labelled production rules corresponding to the stochastic automaton of Figure 1. Strings can now be generated from this stochastic grammar by starting with the string q_0 and progressively choosing productions to rewrite the leftmost non-terminal randomly in proportion to their probability labels. The process terminates once the string contains no non-terminals. The probability of the generated string is the product of the labels of rewrite rules used.

$$0.5 : S \rightarrow \lambda$$
$$0.5 : S \rightarrow aSb$$

Fig. 3. Stochastic context free grammar

LP	**SLP**	eg.
Definite Clause	Labelled Definite Clause	0.3: like(X,Y) ← pet(Y,X), ..
Definition	Labelled Definition	0.5: coin(head) ← 0.5: coin(tail) ←
Program	Labelled Program	0.3: proteinfold1(X) ← .. .. 0.3: proteinfold2(X) ← ..

Fig. 4. Comparison of Syntax of LPs and SLPs

4.5.3 *Stochastic context-free grammars.* Stochastic context-free grammars [Lari and Young 1990] can be treated in the same way as the labelled productions of the last section. However, the following differences exist between the regular and context-free cases.

—To allow for the expression of context-free grammars the left-hand sides of the production rules are allowed to consist of arbitrary strings of terminals and non-terminals.

—Since context-free grammars can have more than one derivation of a particular string u, the probability of u is the sum of the probabilities of the individual derivations of u.

—The analogue of Theorem **??** holds only in relation to the length of the derivation, not the length of the generated string.

EXAMPLE 9. **The language $a^n b^n$.** *Figure 3 shows a stochastic context-free grammar G expressed over the language $a^n b^n$. The probabilities of generated strings are as follows.* $Pr(\lambda|G) = 0.5$, $Pr(ab|G) = 0.25$, $Pr(aabb|G) = 0.125$.

4.5.4 *Syntax for SLPs.* Every context-free grammar can be expressed as a definite clause grammar [Clocksin and Mellish 1981]. For this reason the generalisation of stochastic context-free grammars to stochastic logic programs (SLPs) is reasonably straightforward.

The syntax of SLPs is illustrated in Figure 4 by comparison with the more familiar syntax of Logic Programs. SLPs consists of a set of labelled clauses $p : C$ where p is from the interval $[0, 1]$ and C is a range-restricted definite clause. The probability labels on any predicate definition must sum to no more than 1.

4.5.5 *Semantics for SLPs.* The semantics of SLPs is illustrated in Figure 5 by comparison with that of Logic Programs. SLPs have a distributional semantics, that is one which assigns a probability distribution to the atoms of each predicate in the Herbrand base of the underlying (unlabelled) logic program. An interpretation M is a model of an SLP S if all the atoms a have a probability assigned by M which

LP	**SLP**	eg.
Interpretation	Distributional Interpretation	0.3: p(a) .. 0.4: q(a) ..
Model	Distributional Model	0.3: p(a) .. 0.3: q(a) ..
$P \models Q$	$P \models Q$	0.3: q(a), 1.0: p(X) ← q(X) $\models$ 0.3: p(a)

Fig. 5. Comparison of Semantics of LPs and SLPs

LP	**SLP**	eg.
SLD derivation	SSLD derivation	{0.3: q(a), 1.0: p(X) ← q(X) } refutes goal 0.3: ← p(a)

Fig. 6. Comparison of Proof for LPs and SLPs

is at least the sum of the probabilities of derivations of a with respect to S.

4.5.6 *Proof for SLPs.* Proof for SLPs is illustrated in Figure 5. Probabilities are assigned to atoms according to an SLD-resolution strategy which employs a stochastic selection rule. Derivations can be viewed as Markov chains in which each stochastic selection is made randomly and independently. Thus the probability of deriving any particular atom a is the sum of products of the probability labels on the derivations of a.

5. LEARNING PROBABILISTIC LOGIC REPRESENTATIONS

The section describes some of the initial approaches which have been taken to learning probabilistic logic representations.

5.1 Learning PRMS

PRMs share the underlying probabilistic semantics and local independence assumptions of Bayesian Networks. This has allowed many of the Bayesian net learning techniques to be extended to PRMs. For instance, Koller and Pfeffer [Koller and Pfeffer 1997] have used EM (Expectation Maximisation [Dempster et al. 1977]) to estimate the parameters θ_S of a PRM for which the dependency structure is known. More recently Friedman et al. [Friedman et al. 1999] have also attacked the more difficult problem of learning the dependency structure S directly from data.

5.2 Learning SLPs

The task of learning SLPs, like that of learning PRMs, can also be divided into that of parameter estimation and structure learning. Cussens [Cussens 2000] presents a new algorithm called Failure-Adjusted Maximisation (FAM) which estimates from data the parameters of an SLP for which the underlying logic program is given. FAM is an instance of the EM algorithm that applies specifically to normalised SLPs. Recently the author [Muggleton 2000b] has presented a preliminary algo-

rithm for learning both the parameters of an SLP and the underlying logic program from data. The algorithm is based on maximising Bayes' posterior probability, and has been demonstrated on problems involving learning an animal taxonomy and a simple English grammar.

6. CONCLUSION

This paper reviews three distinct applications of Statistics within Logic-Based Machine Learning. These approaches involve 1) the use of Bayesian probabilistic preference functions (Bayesian Inductive Logic programming), 2) the analysis of convergence of Machine Learning algorithms (Computational Learning Theory and U-learnability) and 3) the learning of probabilistic representations.

Machine Learning and Uncertainty Reasoning are presently dealt with by a variety of research communities using a wide array of approaches. These approaches are separated largely by differences in the underlying representation of the knowledge being learned and reasoned about (eg. Hidden Markov Models, Bayes' nets, decision trees and Logic Programs). Each formalism has representational advantages for particular tasks. However, large-scale applications, such as those found in Bioinformatics require a broad mix of these representations. This is not supplied by any one approach. Amalgams of representations can be used successfully. However, such amalgams do not scale well because of the lack of uniformity of reasoning both about and within such systems. A radically new approach to unifying the underlying representations is required. The author believes research into learning of probabilistic logic representations has an important long-term potential in providing a unifying framework for Machine Learning and Uncertainty Reasoning.

ACKNOWLEDGMENTS

The author would like to thank the following people who strongly influenced the research in this paper: David Page, David Haussler, Wray Buntine, James Cussens, Rupert Parson, Khalid Khan and Tony Hoare. Thanks are due to the funders and researchers who helped develop the technology and applications of ILP on the Esprit projects ECOLES (1989-1992), ILP I (1992-1995), ILPnet (1992-1995), ILP II (1996-1999), ILPnet2 (1998-2001), ALADIN (1998-2001) and the EPSRC Rule-based system project (1990-1993), Experimental Application and Developments of ILP (1993-1996), Distribution-based Machine Learning (1995-1998), Closed Loop Machine Learning (1992-2002). The author would also like to acknowledge the personal support he received in carrying out research into ILP under an SERC Post-doctoral Fellowship (1990-1992), an EPSRC Advanced Research Fellowship (1993-1998) and Research Fellowship from Wolfson College (1993-1997). Warm thanks are offered to the author's wife Thirza and daughter Clare for their continuous cheerful support.

REFERENCES

ARBAB, B. AND MICHIE, D. 1985. Generating rules from examples. In *IJCAI-85*. Kaufmann, Los Altos, CA, 631–633.

BEN-DAVID, S., CHOR, B., AND GOLDREICH, O. 1992. On the theory of average case complexity. *Journal of Information and System Sciences 44*, 193–219.

BOARD, R. AND PITT, L. 1989. On the necessity of Occam algorithms. UIUCDCS-R-89-1544, University of Illinois at Urbana-Champaign.

BOOLE, G. 1854. *The Laws of Thought.* MacMillan & Co., London.

BRATKO, I. 1983. Generating human-understandable decision rules. Working paper, E. Kardelj University Ljubljana, Ljubljana, Yugoslavia.

BUNTINE, W. 1990. A theory of learning classification rules. Ph.D. thesis, School of Computing Science, University of Technology, Sydney.

CARNAP, R. 1962. *The Logical Foundations of Probability.* University of Chicago Press, Chicago.

CLOCKSIN, W. AND MELLISH, C. 1981. *Programming in Prolog.* Springer-Verlag, Berlin.

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

COHEN, W. AND PAGE, C. 1995. Polynomial learnability and Inductive Logic Programming: methods and results. *New Generation Computing 13*, 369–409.

CUSSENS, J. 1999. Loglinear models for first-order probabilistic reasoning. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence.* Kaufmann, San Francisco, 126–133.

CUSSENS, J. 2000. Parameter estimation in stochastic logic programs. *Machine Learning.* In press.

DEKHTYAR, A. AND SUBRAHMANIAN, V. 2000. Hybrid probabilistic programs. *Journal of Logic Programming 43,* 3, 187–250.

DEMPSTER, A., LAIRD, N., AND RUBIN, D. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B 39*, 1–38.

DUDA, R., GASHNIG, J., AND HART, P. 1979. Model design in the prospector consultant program for mineral exploration. In *Expert Systems in the Microelectronic Age*, D. Michie, Ed. Edinburgh University Press, Edinburgh, 153–167.

DŽEROSKI, S., MUGGLETON, S., AND RUSSELL, S. 1992. PAC-learnability of determinate logic programs. In *Proceedings of the 5th ACM Workshop on Computational Learning Theory.* Pittsburg, PA.

FRIEDMAN, N., GETOOR, L., KOLLER, D., AND PFEFFER, A. 1999. Learning probabilistic relational models. In *IJCAI-99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence.* Morgan-Kaufmann, San Mateo, CA:, 1300–1309.

GOLD, E. 1967. Language identification in the limit. *Information and Control 10*, 447–474.

HALPERN, J. Y. 1990. An analysis of first-order logics of probability. *Artificial Intelligence* 46, 311–350.

HAUSSLER, D. 1990. Applying Valiant's learning framework to AI concept-learning problems. In *Machine learning: an artificial intelligence approach*, Y. Kodratoff and R. Michalski, Eds. Vol. 3. Morgan Kaufman, San Mateo, CA, 641–669.

HAUSSLER, D., KEARNS, M., AND SHAPIRE, R. 1991. 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.* Morgan Kauffmann, San Mateo, CA, 61–74.

HAUSSLER, D., KEARNS, M., AND SHAPIRE, R. 1994. Bounds on the sample complexity of Bayesian learning using information theory and the VC dimension. *Machine Learning Journal 14,* 1, 83–113.

HOGGER, C. 1990. *Essentials of logic programming.* Oxford University Press, Oxford.

KIETZ, J. 1993. Some lower bounds on the computational complexity of inductive logic programming. In *Proceedings of the 6th European Conference on Machine Learning*, P. Brazdil, Ed. Lecture Notes in Artificial Intelligence, vol. 667. Springer-Verlag, 115–123.

KOLLER, D. AND PFEFFER, A. 1997. Learning probabilities for noisy first-order rules. In *IJCAI-97: Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence.* Morgan-Kaufmann, San Mateo, CA:, 1316–1321.

KOLLER, D. AND PFEFFER, A. 1998. Probabilistic frame-based systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence, AAAI98*. AAAI Press / The MIT Press, 157–164.

KOWALSKI, R. 1980. *Logic for Problem Solving*. North Holland.

LARI, K. AND YOUNG, S. J. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language 4*, 35–56.

LLOYD, J. 1987. *Foundations of Logic Programming*. Springer-Verlag, Berlin. Second edition.

MCCREATH, E. AND SHARMA, A. 1997. ILP with noise and fixed example size: A Bayesian approach. In *IJCAI-97: Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*. Morgan-Kaufmann, San Mateo, CA:, 1310–1315.

MUGGLETON, S. 1991. Inductive Logic Programming. *New Generation Computing 8,* 4, 295–318.

MUGGLETON, S. 1994. Bayesian Inductive Logic Programming. In *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory*, M. Warmuth, Ed. ACM Press, New York, 3–11. Keynote presentation.

MUGGLETON, S. 1995. Inverse entailment and Progol. *New Generation Computing 13*, 245–286.

MUGGLETON, S. 1996. Stochastic logic programs. In *Advances in Inductive Logic Programming*, L. de Raedt, Ed. IOS Press, 254–264.

MUGGLETON, S. 1999a. Inductive logic programming: issues, results and the LLL challenge. *Artificial Intelligence 114,* 1–2 (December), 283–296.

MUGGLETON, S. 1999b. Scientific knowledge discovery using Inductive Logic Programming. *Communications of the ACM 42,* 11 (November), 42–46.

MUGGLETON, S. 2000a. Learning from positive data. *Machine Learning*. Accepted subject to revision.

MUGGLETON, S. 2000b. Learning stochastic logic programs. In *Proceedings of the AAAI2000 workshop on Learning Statistical Models from Relational Data*, L. Getoor and D. Jensen, Eds. AAAI.

MUGGLETON, S. AND PAGE, C. 1994. A learnability model for universal representations. Tech. Rep. PRG-TR-3-94, Oxford University Computing Laboratory, Oxford. ftp://ftp.cs.york.ac.uk/pub/ML_GROUP/Papers/ulearn10.ps.gz.

MUGGLETON, S. AND RAEDT, L. D. 1994. Inductive logic programming: Theory and methods. *Journal of Logic Programming 19,20*, 629–679.

MUGGLETON, S., SRINIVASAN, A., AND BAIN, M. 1992. Compression, significance and accuracy. In *Proceedings of the Ninth International Machine Learning Conference*, D. Sleeman and P. Edwards, Eds. Morgan-Kaufmann, San Mateo, CA, 338–347.

NG, R. AND SUBRAHMANIAN, V. 1992. Probabilistic logic programming. *Information and Computation 101,* 2, 150–201.

NIENHUYS-CHENG, S.-H. AND DE WOLF, R. 1997. *Foundations of Inductive Logic Programming*. Springer-Verlag, Berlin. LNAI 1228.

NOCK, R. AND JAPPY, P. 1998. Function-free Horn clauses are hard to approximate. In *Proceedings of the Eighth Inductive Logic Programming Workshop (ILP98)*, D. Page, Ed. Springer-Verlag, Berlin. LNAI 1446.

PAGE, D. AND FRISCH, A. 1992. Generalization and learnability: A study of constrained atoms. In *Inductive Logic Programming*, S. Muggleton, Ed. Academic Press, London.

PARSON, R., KHAN, K., AND MUGGLETON, S. 1999. Theory recovery. In *Proc. of the 9th International Workshop on Inductive Logic Programming (ILP-99)*. Springer-Verlag, Berlin.

PEARL, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, Los Altos.

POOLE, D. 1993. Probabilistic Horn abduction and Bayesian networks. *Artificial Intelligence 64,* 1, 81–129.

QUINLAN, J. 1990. Learning logical definitions from relations. *Machine Learning 5*, 239–266.

RABINER, L. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE 77,* 2, 257–286.

RAEDT, L. D. AND DŽEROSKI, S. 1994. First-order jk-clause theories are pac-learnable. Tech. rep., Department of Computer Science, Katholieke Universieit Leuven, Heverlee, Belgium.

SAMMUT, C., HURST, S., KEDZIER, D., AND MICHIE, D. 1992. Learning to fly. In *Proceedings of the Ninth International Workshop on Machine Learning*, D. Sleeman and P. Edwards, Eds. Morgan Kaufmann, San Mateo, CA, 385–393.

SHORTLIFFE, E. AND BUCHANAN, B. 1975. A model of inexact reasoning in medicine. *Mathematical Biosciences 23*, 351–379.