Learning to Reason

Roni Khardon* **Dan Roth†**

Aiken Computation Laboratory,
Harvard University,
Cambridge, MA 02138.
{roni,danr}@das.harvard.edu

Abstract

We introduce a new framework for the study of reasoning. The Learning (in order) to Reason approach developed here combines the interfaces to the world used by known learning models with the reasoning task and a performance criterion suitable for it.

We show how previous results from learning theory and reasoning fit into this framework and illustrate the usefulness of the Learning to Reason approach by exhibiting new results that are not possible in the traditional setting. First, we give a Learning to Reason algorithm for a class of propositional languages for which there are no efficient reasoning algorithms, when represented as a traditional (formula-based) knowledge base. Second, we exhibit a Learning to Reason algorithm for a class of propositional languages that is not known to be learnable in the traditional sense.

Introduction

Consider a baby robot, starting out its life. If it were a human being, nature would have provided for the infant a safe environment to spend an initial time period in. In this period she adapts to her environment and learns about the structures, rules, meta-rules, superstitions and other information the environment provides for. In the meantime, the environment protects her from fatal events. Only after this "grace period", she is expected to have "full functionality" (whatever that means) in her environment, but it is expected that her performance depends on the world she grew up in and reflects the amount of interaction she has had with it.

Realizing that learning is a central aspect of cognition, computational learning theory, a subfield concerned with modeling and understanding learning phenomena (Valiant 1984), takes a similar view in that the performance of the learner is measured only after the learning period, and with respect to the world. Traditional theories of intelligent systems, however, have assumed that cognition (namely, computational processes like reasoning, language recognition, object identification and other "higher level" cognitive tasks) can be studied separately from learning (See (Kirsh 1991) for a discussion of this issue.). However, computational considerations render this self-contained reasoning approach as well as other variants of it (Selman 1990; Roth 1993), not adequate for common-sense reasoning (Levesque 1986; Shastri 1993).

We argue that the main difficulties in the traditional treatment of reasoning stem from its separation from the "world". The effect is twofolded: (1) a rigid representation language with which reasoning problems are presented to the reasoner (Brooks 1991) and (2) a performance criterion that is irrespective of the world.

In this paper we develop a new framework for the study of Reasoning. Our approach differs from other approaches to reasoning in that it views learning as an integral part of the process. The *Learning to Reason* theory developed here is concerned with studying the entire process of *learning* a knowledge base representation and *reasoning* with it.

In the new framework the intelligent agent is given access to her favorite learning interface, and is also given a grace period in which she can interact with this interface and construct her representation[1] KB of the world W. Her reasoning performance is measured only after this period, when she is presented with queries α from some query language, relevant to the world[2], and has to answer whether W implies α. We show that through this interaction with the world, the agents truly gains additional reasoning power. An inherent feature of the learning to reason approach, is a non-monotonic reasoning behavior it exhibits as a side effect; since reasoning mistakes can be used, incremen-

*Research supported by grant DAAL03-92-G-0164 (Center for Intelligent Control Systems).

†Research supported by NSF grant CCR-92-00884 and by DARPA AFOSR-F4962-92-J-0466.

[1] We stress that the assumption that an intelligent agent has to keep her knowledge in some representation and use it when she reasons is basic to this framework. We allow the reasoner, however, to choose her own representation and even to use different representations for different tasks.

[2] We take the view that a reasoner need not answer efficiently *all* possible queries, but only those that are "relevant", or "common", in a well defined sense.

tally, to improve the knowledge base. This desirable phenomenon is hard to formalize, when dealing with reasoning systems defined independent of learning.

This work is similar in nature to the Neuroidal model developed by Valiant (Valiant 1994). The model developed there provides a more comprehensive approach to cognition, and akin to our approach it views learning as an integral and crucial part of the process. There, the reasoner reasons from a learned knowledge base, a complex circuit, and thus can be modeled by our framework, and indeed reasoning in the Neuroidal model shares many properties with the Learning to Reason framework. Our approach is different in that in an effort to give a more formal treatment of a reasoner that has learned her knowledge base, we currently restrict our discussion to a fixed, consistent world.

Summary of Results

To motivate our approach we first develop a sampling approach to reasoning that exemplifies the power gained by giving the agent access to the "world" she is supposed to reason in later. We prove that Learning to reason is possible for arbitrary worlds and query languages under some technical restriction on the queries asked. However, for various considerations this approach alone is not sufficient as a model for reasoning, so we go on to define the Learning to Reason framework to capture those. We start by studying the relation of the Learning to Reason (L2R) framework to the two existing ones, the traditional reasoning, and the traditional learning (learning to classify, L2C), and discuss how existing results from learning and reasoning can be used in the new framework. We then consider detailed Learning to Reason algorithms that use models (satisfying assignments) as the knowledge representation language. A characterization of reasoning with models for Horn theories was given in (Kautz, Kearns, & Selman 1993) and for general theories in (Khardon & Roth 1994b). We build on these results to exemplify the usefulness of the new approach:

- Consider the reasoning problem $W \models \alpha$, where W is some CNF formula and α is a $\log n$CNF (i.e., a CNF formula with at most $\log n$ literals in each clause). Then, when W has a polynomial size DNF[3] there is an exact and efficient Learning to Reason algorithm for this problem, while the traditional reasoning problem (with a CNF representation as the input) is NP-Hard.
- Consider the reasoning problem $W \models \alpha$, where W is any boolean formula with a polynomial size DNF and α is a $\log n$CNF. Then, there is an exact and efficient Learning to Reason algorithm for this problem, while the class of boolean formulas with polynomial size DNF is not known to be learnable in the traditional (Learning to Classify) sense.

These results show that neither a traditional reasoning algorithm (from the CNF representation) nor a traditional learning algorithm (that can "classify" the world) is necessary to Learn to Reason. Moreover, the results exemplify and aid in formalizing the notion of "intelligence is in the eye of the beholder" (Brooks 1991), since our agent seems to behave logically, even though her knowledge representation need not be a logical formula and she does not use any logic or "theorem proving".

Due to the limited space we omit some details and most of the proofs. These can be found in the full version of this paper (Khardon & Roth 1994a).

Preliminaries

Reasoning

A widely accepted framework for reasoning in intelligent systems is the knowledge-based system approach (McCarthy 1958). Knowledge, in some *representation language* is stored in a *Knowledge Base* (KB) that is combined with a reasoning mechanism. Reasoning is abstracted as a deduction task[4] of determining whether a *query* α, assumed to capture the situation at hand, is implied from KB (denoted $KB \models \alpha$). The discussion in this paper is restricted to propositional knowledge bases[5].

Let $\mathcal{F}, \mathcal{Q}$ be two arbitrary classes of representations for boolean functions. All the functions we discuss are boolean functions over $\{0,1\}^n$, where n is the number of variables in the domain. Throughout this paper we assume that an exact description of the real world W is in $\mathcal{F}$, that all the queries α are restricted to be in the class $\mathcal{Q}$ and that the functions have a polynomial size representation in the respective class. In particular, the class $\mathcal{F} = \text{CNF}$ denotes those boolean functions with a polynomial size CNF and the class $\mathcal{F} = \text{CNF} \cap \text{DNF}$ denotes those boolean functions with a polynomial size CNF and a polynomial size DNF.

We refer to boolean functions as either functions or subsets of $\{0,1\}^n$: the boolean function g is identified with its set of models, $g^{-1}(1)$ (That is, $f \models g$ if and only if $f \subseteq g$).

Definition 1 *An algorithm A is an* exact reasoning *algorithm for the reasoning problem $(\mathcal{F}, \mathcal{Q})$ if for all $f \in \mathcal{F}$ and all $\alpha \in \mathcal{Q}$, when A is presented with input (f, α), A runs in time polynomial in n and the size of f and α, and answers "yes" if and only if $f \models \alpha$.*

[3] The DNF representation is not given to the reasoner. Its existence is essential, since the algorithm is polynomial in its size.

[4] We restrict ourselves here to deduction although the approach developed is applicable for other reasoning tasks, e.g., Bayesian networks.

[5] A propositional expression is just a boolean function, and a propositional language is a class of boolean functions. These terms are used in the reasoning and learning literature accordingly, and we use them interchangeably.

Learning to Classify

The formal study of learning, (studied in computational learning theory (Valiant 1984; Haussler 1987; Angluin 1992)), abstracts the problem of inductively learning a concept as the problem of learning a boolean function, given some access to an oracle that is familiar to some degree with the function. The interpretation is that the function's value is 1 when the input belongs to the target concept and 0 otherwise. The oracles are used to model the type of interface the learner may have to the world and they vary between learning models according to the amount of information we assume the learner receives about the concept. Next we describe some standard oracles, introduce a new one that is especially suited for Reasoning and define the learning problem.

Definition 2 *A* Membership Query Oracle *for a function f, denoted $MQ(f)$, is an oracle that when given an input $x \in \{0,1\}^n$ returns $f(x)$.*

Definition 3 *An* Equivalence Query Oracle *for a function f, denoted $EQ(f)$, is an oracle that when given as input a function g, answer "yes" if and only if $f \equiv g$. If it answers "no" it supplies a counterexample, namely, an $x \in \{0,1\}^n$ such that $f(x) \neq g(x)$. A counterexample x satisfying $f(x) = 1$ ($f(x) = 0$) is called a positive (negative) counterexample.*

Definition 4 *An* Example Oracle *for a function f, with respect to the probability distribution D, denoted $EX_D(f)$, is an oracle that when accessed, returns $(x, f(x))$, where x is drawn at random according to D.*

Definition 5 *A* Reasoning Query Oracle *for a function f and a query language $\mathcal{Q}$, denoted $RQ(f, \mathcal{Q})$, is an oracle that when accessed performs the following protocol with a learning agent A. (1) The oracle picks an arbitrary query $\alpha \in \mathcal{Q}$ and returns it to A. (2) The agent A answers "yes" or "no" according to her belief with regard to the truth of the statement $f \models \alpha$. (3) If A's answer is correct then the oracle says "Correct". If the answer is wrong the oracle answers "Wrong" and in case $f \not\models \alpha$ it also supplies a counterexample (i.e., $x \in f \setminus \alpha$).*

Denote by $I(f)$ the *interface* available to the learner when learning f. This can be any subset of the oracles defined above, and might depend on some fixed but arbitrary and unknown distribution D over the instance space $\{0,1\}^n$.

Definition 6 *An algorithm A is an* Exact Learn to Classify (E-L2C) *algorithm for a class of functions $\mathcal{F}$, if there exists a polynomial $p()$ such that for all $f \in \mathcal{F}$, when given access to $I(f)$, A runs in time $p(n)$ and then, given any $x \in \{0,1\}^n$, takes time $p(n)$ to predict σ such that $\sigma = f(x)$.*

Definition 7 *An algorithm A is a* Probably Approximately Correct Learn to Classify (PAC-L2C) *algorithm for a class of functions $\mathcal{F}$, if there exists a polynomial $p(,,)$ such that for all $f \in \mathcal{F}$, on input ϵ, δ, given access to $I(f)$, A runs in time $p(n, 1/\epsilon, 1/\delta)$ and then given any $x \in \{0,1\}^n$, predicts $h(x)$ in time $p(n, 1/\epsilon, 1/\delta)$. A's predictions have the property that with probability at least $1-\delta$, $Prob_{x \in D}[f(x) \neq h(x)] < \epsilon$.*

In the on-line (or, mistake-bound) scenario, algorithm A is presented with a sequence of examples in $\{0,1\}^n$. At each stage, the algorithm is asked to predict $f(x)$ and is then told whether its prediction was correct. Each time the learning algorithm makes an incorrect prediction, we charge it one *mistake*.

Definition 8 *An algorithm A is a* Mistake Bound Learn to Classify (MB-L2C) *algorithm for a class of functions $\mathcal{F}$, if there exists a polynomial $p()$ such that for all $f \in \mathcal{F}$, for every (arbitrary infinite) sequence of instances, A runs in time $p(n)$ (on each example) and makes no more than $p(n)$ mistakes.*

Learning to Reason

Let $W \in \mathcal{F}$ be a boolean function that describes the world exactly. Let α be some boolean function (a query) and let D be some fixed but arbitrary and unknown probability distribution over the instance space $\{0,1\}^n$. As in the learning framework, we assume that D governs the occurrences of instances in the world.

The query α is called *legal* if $\alpha \in \mathcal{Q}$. It is called (W, ϵ)-*fair* if either $Prob_D[W \setminus \alpha] = 0$ or $Prob_D[W \setminus \alpha] > \epsilon$. The intuition behind this definition is that the algorithm is allowed to err in case $W \not\models \alpha$, but the weight of W outside α is very small. Along with ϵ, the *accuracy* parameter, we use a *confidence* parameter, δ, and sometimes might allow the reasoning algorithm to err, with small probability, less than δ.

A Sampling Approach

Consider the following simple approach to reasoning: Whenever presented with a query α, first use the Example Oracle $EX_D(W)$ and take a sample of size $m = (1/\epsilon)\ln(1/\delta)$, where δ and ϵ are the required confidence and accuracy parameters. Then, perform the following model-based test: for all the samples $(x, 1)$ sampled from $EX_D(W)$ (note that we ignore the samples labeled 0), check whether $\alpha(x) = 1$. If for some x, $\alpha(x) = 0$ say $W \not\models \alpha$; otherwise say $W \models \alpha$.

A standard learning theory argument shows that if α is (W, ϵ)-fair then with probability at least $1-\delta$ the algorithm is correct. (The algorithm makes a mistake only if $W \not\models \alpha$ and no instance in $W \cap \overline{\alpha}$ is sampled.) This analysis depends on the fact that the samples are independent of the query α, and therefore a different sample has to be taken for every query α. We call this a *repeated sampling* approach[6]. However, repeated

[6] A similar, more sophisticated approach was developed in (Kearns 1992) for the case in which both the knowledge base and the queries are learned concepts in the PAC sense. It is implicit there that for each possible query one needs a new sample.

sampling is not a plausible approach to reasoning in intelligent systems. When presented with a query, an agent cannot allow itself further interactions with the world before answering the query. Especially if the query is "A lion is approaching $\Rightarrow$ I have to run away".

A slightly more elaborate argument shows that a *one time sampling* approach can also guarantee reasoning with respect to (W, ϵ)-fair queries, with confidence $1 - \delta$. This depends on taking $m = \frac{1}{\epsilon}(\ln |\mathcal{Q}| + \ln \frac{1}{\delta})$ samples from $EX_D(f)$. Since all the queries in $\mathcal{Q}$ are propositional formulas of polynomial size, the number m of samples required to guarantee this performance is polynomial. This approach is therefore feasible.

However, the one-time sampling approach is not the ultimate solution for reasoning. For example it is not adequate in cases where exact reasoning performance is required. In the full version of the paper we elaborate on why this sampling approach is not sufficient as the sole solution for the reasoning problem. (E.g., space considerations and the availability of various oracles needed to model reasoning.)

Learning to Reason: Definitions

Definition 9 *An algorithm A is an* Exact Learn to Reason (E-L2R) *algorithm for the reasoning problem $(\mathcal{F}, \mathcal{Q})$, if there exists a polynomial $p()$ such that for all $f \in \mathcal{F}$, given access to $I(f)$, A runs in time $p(n)$ and then, when presented with any query $\alpha \in \mathcal{Q}$, A runs in time $p(n)$, does not access $I(f)$, and answers "yes" if and only if $f \models \alpha$.*

Definition 10 *An algorithm A is a* Probably Approximately Correct Learn to Reason (PAC-L2R) *algorithm for the reasoning problem $(\mathcal{F}, \mathcal{Q})$, if there exists a polynomial $p(,,)$ such that for all $f \in \mathcal{F}$, on input ϵ, δ, given access to $I(f)$, A runs in time $p(n, 1/\epsilon, 1/\delta)$ and then with probability at least $1 - \delta$, when presented with any (f, ϵ)-fair query $\alpha \in \mathcal{Q}$, A runs in time $p(n, 1/\epsilon, 1/\delta)$, does not access $I(f)$, and answers "yes" if and only if $f \models \alpha$.*

In the above definitions, we did not allow access to $I(f)$ while in the query answering phase. It is possible, however, (although we do not do it in this paper) to consider a query α given to the algorithm as if given by the reasoning oracle $RQ(f, \mathcal{Q})$ defined above. Thus, a reasoning error may supply the algorithm a counterexample which in turn can be used to improve its future reasoning behavior.

The relations between L2R and L2C

Intuitively, the classification task seems to be easier than the reasoning task. In the former we need to evaluate correctly a function on a single point, while in the latter we need to know if *all the models* of the function are also models of another function, the query. It is not surprising therefore, that if *any* subset of $\{0,1\}^n$ is a legal query, the ability to L2R implies the ability to L2C. This is formalized in the following theorem. We note, however, that the proof of the theorem does not go through if the class of queries $\mathcal{Q}$ does not include all of DISJ, the class of all disjunctions over n variables. (See Theorem 7.)

Theorem 1 *If there is an Exact-L2R algorithm for the reasoning problem $(\mathcal{F}, DISJ)$ then there is an Exact-L2C algorithm for the class $\mathcal{F}$.*

L2R via PAC Learning

Assume that the world description W is in $\mathcal{F}$ and there is a PAC-L2C algorithm A for $\mathcal{F}$.

Definition 11 *An algorithm that PAC learns to classify $\mathcal{F}$ is said to* learn f from below *if, when learning f, the algorithm never makes mistakes on instances outside of f. (I.e., if h is the hypothesis the algorithm keeps then it satisfies $h \subseteq f$.)*

Theorem 2 *Let A be a PAC-Learn to Classify algorithm for the function class $\mathcal{F}$ and assume that A uses the class of representations $\mathcal{H}$ as its hypotheses. Then, if A learns $\mathcal{F}$ from below, and there is an exact reasoning algorithm B for the reasoning problem $(\mathcal{H}, \mathcal{Q})$, then there is a PAC-Learn to Reason algorithm C for the reasoning problem $(\mathcal{F}, \mathcal{Q})$.*

The significance of this result is that it exhibits the *limitations* of L2R by combining reasoning and learning algorithms: relaxing the requirement that the algorithm learns from below is not possible. On the positive side it explains the behavior of mistake bound algorithms discussed next and allows for other PAC learning algorithms to be used in this framework.

L2R via Mistake Bound Learning

Consider a Mistake Bound algorithm that keeps a hypothesis that allows for efficient reasoning. Then, it can be used to construct a Learn to Reason algorithm.

Let A be a Mistake Bound algorithm and assume it has been used long enough to guarantee PAC performance (Littlestone 1989). In the case it has used up all of its mistakes on negative examples (i.e., on assignments outside of W), the hypothesis it uses is a "learn from below" hypothesis, and we can reason with it and succeed on all (W, ϵ)-fair queries.

However, we cannot force the algorithm (or rather the interface) to make all these mistakes within the grace period. If we use an initial grace period to ensure its PAC properties then after the algorithm is ready to answer queries it may still make (a limited number of) mistakes. We call this type of algorithm a Mistake Bound Learning to Reason algorithm.

It is interesting to note that reasoning with this type of an algorithm yields a non monotonic reasoning behavior. Every time the algorithm makes a reasoning mistake, it changes its mind, learns something about the world, and would not make the same mistake again. This is an inherent feature of the learning to reason approach, and it captures a phenomenon that is hard to

formalize, when dealing with reasoning systems defined independent of learning.

L2R via Model Based Reasoning

In this section we develop the main technical results of this paper and exhibit the advantages of the Learning to Reason approach. We deviate from the traditional setting of "first learn to classify, then reason with the hypothesis": A learning algorithm is used first, but rather then learning a "classifying hypothesis", it constructs a knowledge representation that allows for efficient reasoning.

The results in this section use two recent results, one on learning via monotone theory (Bshouty 1993) and the other on reasoning with models (Khardon & Roth 1994b). Combining these two results yields Theorem 3. (Notice, though, that the reasoning algorithm does not use the "classifying hypothesis" of the learning algorithm but rather a set a models, a byproduct of it.)

Queries are called *relevant* if they are in $\mathcal{F}$ (we also assume $W \in \mathcal{F}$). Queries are called *common* if they belong to some set $\mathcal{L}_E$ of *efficient* propositional languages. Important examples of efficient languages are: $\log n$CNF theories (CNF in which the clauses contain at most $O(\log n)$ literals), k-quasi-Horn queries (a generalization of Horn theories in which there are at most k positive literals in each clause) and others.

Theorem 3 *There is an Exact-Learn to Reason algorithm, that uses an Equivalence Query and a Membership Query Oracles, for $(CNF \cap DNF, \mathcal{Q})$, where $\mathcal{Q}$ is the class of all relevant and common queries.*

The above theorem is an example for a reasoning problem that is provably hard in the "traditional" sense and has an efficient solution in the new model. Given a CNF knowledge base, even with the added information that it has a short DNF, the reasoning problem is still hard. This is so since it is NP-hard to find a satisfying assignment for a CNF expression even if one knows that it has exactly one satisfying assignment (Valiant & Vazirani 1986). The algorithm does not solve an NP-hard problem; the additional reasoning power of the agent is gained through the interaction with the world by using $EQ(f)$ or $EX_D(f)$.

To present the next result we first introduce some definitions and results from the monotone theory of boolean functions and the theory of reasoning with models (Bshouty 1993; Khardon & Roth 1994b).

Monotone Theory and Reasoning with Models

Definition 12 (Order) *We denote by $\leq$ the* usual partial order *on the lattice $\{0,1\}^n$, the one induced by the order $0 < 1$. That is, for $x, y \in \{0,1\}^n$, $x \leq y$ if and only if $\forall i, x_i \leq y_i$. For an assignment $b \in \{0,1\}^n$ we define $x \leq_b y$ if and only if $x \oplus b \leq y \oplus b$ (Here $\oplus$ is the bitwise addition modulo 2).*

Intuitively, if $b_i = 0$ then the order relation on the ith bit is the normal order; if $b_i = 1$, the order relation is reversed and we have that $1 <_{b_i} 0$. We now define: The *monotone extension of* $z \in \{0,1\}^n$ with respect to b:

$$\mathcal{M}_b(z) = \{x \mid x \geq_b z\}.$$

The *monotone extension of* f with respect to b:

$$\mathcal{M}_b(f) = \{x \mid x \geq_b z, \ \textit{for some } z \in f\}.$$

The set of *minimal assignments of* f with respect to b:

$$\min_b(f) = \{z \mid z \in f, \ \textit{such that } \forall y \in f, z \not\geq_b y\}.$$

Every boolean function f can be represented in the following form:

$$f = \bigwedge_{b \in B} \mathcal{M}_b(f) = \bigwedge_{b \in B} \bigvee_{z \in \min_b(f)} \mathcal{M}_b(z) \qquad (1)$$

In the above representation $B \subseteq \{0,1\}^n$ is called a basis. It is known that the size of the basis is at most the CNF size of f, and the size of $\min_b(f)$ is at most its DNF size. (See (Khardon & Roth 1994b) for an exact characterization and a discussion of this issue.) A basis can be used to characterize a class of boolean functions: those which can be expressed with it as in Eq. (1). It is known, for example, that the class of Horn CNF functions has a basis of size $n+1$, and that the class of $\log n$CNF functions has a basis of size less than n^3.

Let $\Gamma \subseteq KB \subseteq \{0,1\}^n$ be a set of models. To decide whether KB$\models \alpha$ use the model-based approach to deduction: for all the models $z \in \Gamma$ check whether $\alpha(z) = 1$. If for some z, $\alpha(z) = 0$ say "No"; otherwise say "Yes". This approach is feasible if Γ is small.

Definition 13 *Let $\mathcal{F}$ be a class of functions, and let B be a basis for $\mathcal{F}$. For a knowledge base $KB \in \mathcal{F}$ we define the set $\Gamma = \Gamma^B_{KB}$ of* characteristic models *to be the set of all minimal assignments of KB with respect to the basis B. Formally,*

$$\Gamma^B_{KB} = \cup_{b \in B}\{z \in min_b(KB)\}.$$

Theorem 4 *Let KB, $\alpha \in \mathcal{F}$ and let B be a basis for $\mathcal{F}$. Then $KB \models \alpha$ if and only if for every $u \in \Gamma^B_{KB}$, $\alpha(u) = 1$.*

Definition 14 (Least Upper-bound) *Let $\mathcal{F}, \mathcal{G}$ be families of propositional languages. Given $f \in \mathcal{F}$ we say that $f_{lub} \in \mathcal{G}$ is a $\mathcal{G}$-least upper bound of f iff $f \subseteq f_{lub}$ and there is no $f' \in \mathcal{G}$ such that $f \subset f' \subset f_{lub}$.*

Theorem 5 *Let f be any propositional theory and $\mathcal{G}$ a class of all propositional theories with basis B. Then*

$$f_{lub} = \bigwedge_{b \in B} \mathcal{M}_b(f).$$

Theorem 6 *Let $KB \in \mathcal{F}$, $\alpha \in \mathcal{G}$ and let B be a basis for $\mathcal{G}$. Then $KB \models \alpha$ if and only if for every $u \in \Gamma^B_{KB}$, $\alpha(u) = 1$.*

L2R without Learning to Classify

The following algorithm is based on a modified version of an algorithm from (Bshouty 1993). We make use of a Reasoning Query Oracle $RQ(f, \mathcal{Q})$ and a Membership Query Oracle $MQ(f)$ to exactly Learn to Reason any boolean function f with a polynomial size DNF.

We note that the requirements of the algorithm can be relaxed: a slightly more complicated version, using $EX_D(f)$ instead of $RQ(f, \mathcal{Q})$, can be used to PAC-Learn to Reason f. Details are given in the full version.

Let B be the basis for the class of queries $\mathcal{Q}$. The algorithm collects a set of models $\Gamma = \cup_{b \in B} \Gamma_b$, the set of minimal assignments of f with respect to B. By Theorem 6 this is the set of the minimal models of $f_{lub} = \wedge_{b \in B} \mathcal{M}_b(f)$.

> **Algorithm $\mathcal{A}$:** For all $b \in B$ initialize $\Gamma_b = \phi$. To get counterexamples, call $RQ(f, Q)$, for which the algorithm responses by performing the model-based test on the set $\cup_{b \in B} \Gamma_b$ (and therefore, answers "yes" initially). When it makes a mistake on a "yes" answer[7], it receives a positive counterexample x. In this case, the algorithm first finds $b \in B$ such that $x \notin \mathcal{M}_b(\Gamma_b)$ and then uses a greedy procedure that repeatedly calls $MQ(f)$ to find a new minimal model of f with respect to the order b. ((Angluin 1988; Bshouty 1993). Details in the full version.)
>
> In the query-answering phase, When given a query $\alpha \in \mathcal{Q}$, the algorithm answers by performing the model-based reasoning using the set $\Gamma = \cup_{b \in B} \Gamma_b$.

The algorithm is essentially a Mistake Bound algorithm that learns $f_{lub} = \wedge_{b \in B} \mathcal{M}_b(f)$ from below.

Theorem 7 *Algorithm $\mathcal{A}$ is a Exact-Learn to Reason algorithm for the problem $(DNF, \mathcal{Q})$, where $\mathcal{Q}$ is the class of all common queries.*

Proof: [sketch] Denote $h = \wedge_{b \in B} (\vee_{z \in \Gamma_b} \mathcal{M}_b(z))$. Clearly, $\Gamma \subseteq f$, and therefore the algorithm $\mathcal{A}$ never makes a mistake when it says "no" (and is therefore well defined). Whenever the algorithm errs on an $RQ(f, Q)$ query, it receives a positive counterexample, $x \in f \setminus h$. Since x is negative for at least one of the b's in the conjunction defining h, there exists a model $z \in \min_b(f) \setminus \Gamma_b$ for each of these b's. Therefore, in this case, the algorithm can use a sequence of calls to $MQ(f)$ to find a new model of f, an element of Γ_f^B. Thus, with every such mistake the algorithm makes progress toward collecting the elements in the set Γ_f^B. Therefore, after at most $|\Gamma_f^B|$ calls to $RQ(f, \mathcal{Q})$ algorithm $\mathcal{A}$ makes no more mistakes on $RQ(f, \mathcal{Q})$ queries and therefore $h = f_{lub}$. Theorem 6 implies that $\mathcal{A}$ is an Exact-Learn to Reason algorithm for f. ■

This should be contrasted with the inability to *learn to classify* DNF. One can learn f_{lub} and reason with it with respect to common queries, but f_{lub} is not sufficient as a substitute for f when classifying examples.

[7] The algorithm never makes mistakes when it responses with "no" on an $RQ(f, Q)$ query.

Acknowledgments

We are grateful to Les Valiant for many enjoyable discussions that helped us develop the ideas presented here.

References

Angluin, D. 1988. Queries and concept learning. *Machine Learning* 2(4):319–342.

Angluin, D. 1992. Computational learning theory: Survey and selected bibliography. In *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing*, 351–369.

Brooks, R. A. 1991. Intelligence without representation. *Artificial Intelligence* 47:139–159.

Bshouty, N. H. 1993. Exact learning via the monotone theory. In *Proceedings of the IEEE Symp. on Foundation of Computer Science*, 302–311.

Haussler, D. 1987. Bias, version spaces and Valiant's learning framework. In *Proceedings of the Fourth International Workshop on Machine Learning*, 324–336.

Kautz, H.; Kearns, M.; and Selman, B. 1993. Reasoning with characteristic models. In *Proceedings of the National Conference on Artificial Intelligence*, 34–39.

Kearns, M. 1992. Oblivious pac learning of concepts hierarchies. In *Proceedings of the National Conference on Artificial Intelligence*, 215–222.

Khardon, R., and Roth, D. 1994a. Learning to reason. Technical Report TR-2-94, Aiken Computation Lab., Harvard University.

Khardon, R., and Roth, D. 1994b. Reasoning with models. In these Proceedings.

Kirsh, D. 1991. Foundations of AI: the big issues. *Artificial Intelligence* 47:3–30.

Levesque, H. 1986. Making believers out of computers. *Artificial Intelligence* 30:81–108.

Littlestone, N. 1989. *Mistake bounds and logarithmic linear-threshold learning algorithms.* Ph.D. Dissertation, U. C. Santa Cruz.

McCarthy, J. 1958. Programs with common sense. In Brachman, R., and Levesque, H., eds., *Readings in Knowledge Representation, 1985.* Morgan-Kaufmann.

Roth, D. 1993. On the hardness of approximate reasoning. In *Proceedings of the International Joint Conference of Artificial Intelligence*, 613–618.

Selman, B. 1990. *Tractable Default Reasoning.* Ph.D. Dissertation, Department of Computer Science, University of Toronto.

Shastri, L. 1993. A computational model of tractable reasoning - taking inspiration from cognition. In *Proceedings of the International Joint Conference of Artificial Intelligence*, 202–207.

Valiant, L. G., and Vazirani, V. V. 1986. NP is as easy as detecting unique solutions. *Theoretical Computer Science* 47:85–93.

Valiant, L. G. 1984. A theory of the learnable. *Communications of the ACM* 27(11):1134–1142.

Valiant, L. G. 1994. *Circuits of the Mind.* Oxford University Press. Forthcoming.