JOURNAL OF COMPUTER AND SYSTEM SCIENCES 52, 255–267 (1996)
ARTICLE NO. 0020

Teaching a Smarter Learner*

SALLY A. GOLDMAN† AND H. DAVID MATHIAS‡

Department of Computer Science, Washington University, St. Louis, Missouri 63130

Received September 27, 1993; revised April 22, 1994

We introduce a formal model of teaching in which the teacher is tailored to a particular learner, yet the teaching protocol is designed so that no collusion is possible. Not surprisingly, such a model remedies the nonintuitive aspects of other models in which the teacher must successfully teach any consistent learner. We prove that any class that can be exactly identified by a deterministic polynomial-time algorithm with access to a very rich set of example-based queries is teachable by a computationally unbounded teacher and a polynomial-time learner. In addition, we present other general results relating this model of teaching to various previous results. We also consider the problem of designing teacher/learner pairs in which both the teacher and learner are polynomial-time algorithms and describe teacher/learner pairs for the classes of 1-decision lists and Horn sentences.

1. INTRODUCTION

Recently, there has been interest in developing formal models of *teaching* [4, 10, 11, 16, 27] through which we can develop a better understanding of how a teacher can most effectively aid a learner in accomplishing a learning task. A weakness of the formal models of teaching that have been introduced in the learning theory community is that they place stringent restrictions on the learner to ensure that the teacher is not just providing the learner with an encoding of the target. In particular, previous models require the teacher to present a set of examples for which only the target function (or a function logically equivalent to the target) is consistent. Thus, teaching under these models is made unnecessarily difficult since the problem reduces to teaching an obstinate, adversarial learner that tries as hard as possible *not* to learn while always outputting a hypothesis consistent with all previous examples. In fact, if a teacher is required to teach any consistent learner,[1] there are many examples for which an exponential length teaching set is required to teach even those classes for which efficient learning algorithms are known. For many of the classes that can be taught efficiently, it is necessary in previous models to allow the teacher and learner to share a small amount of "trusted information," such as the size of the target function, since there is no other way to eliminate concepts from the given class that are more "complicated" than the target. Jackson and Tomkins [16] are able to show that anything learnable with membership and equivalence queries is teachable with trusted information. However, their method for preventing collusion still requires that the teacher successfully teaches any consistent learner.

As a concrete example of a consequence of requiring that the teaching set is sufficient for any consistent learner, consider the class $\mathscr{C}$ consisting of all singletons plus the empty set. While, this class is quite simple and thus should be easy to teach, Goldman and Kearns show that $|\mathscr{C}|-1$ examples are required to teach it in their model. Furthermore, this hardness result has been embedded into several other hardness results such as that for teaching full decision lists [16] and teaching linearly separable Boolean functions [4]. Thus these hardness results appear to be due to a defect in the model rather than an intrinsic difficulty in teaching these classes. Problems similar to the ones discussed above emerge when comparing these teaching models to the self-directed learning model.[2] In particular, for many classes the self-directed learning complexity is asymptotically less than the teaching complexity. Again, because the teacher must successfully teach all consistent learners, a "smart" self-directed learner can perform better on its own than with the teacher's guidance. Yet, if the teacher and learner could cooperate (which should be the case if the teacher is working with the single learner), intuitively this phenomenon should not occur.

There are many applications in which it is desirable to have teacher/learner pairs. For example, such results provide lower bounds on the number of examples required by a learning algorithm and such bounds could then be used to direct efforts in collecting data. Another application area is the problem of training a neural network. Currently, most

* This research was supported in part by NSF Grant CCR-9110108 and by a GE Foundation Junior Faculty Grant. An earlier version appears in the *Proceedings, of the Sixth Annual Workshop on Computational Learning Theory, July* 1993, pp. 67–86.

† E-mail: sg@cs.wustl.edu.

‡ E-mail: dmath@cs.wustl.edu.

[1] A learner is consistent if its hypotheses are consistent with all previously seen examples.

[2] A self-directed learner [12, 13] is a learner that selects the presentation order for the instances. In this model, the learning complexity is measured according to the number of *incorrect* predictions made.

0022-0000/96 $18.00

training algorithms work by adjusting the weights in the network based on randomly selected labeled examples. By having a teacher carefully select the set of labeled examples, the training time might be drastically reduced. In addition, there are potential applications of the research on formal models of teaching to improving automated manufacturing environments. For example, consider the problem of training sensor-referenced intelligent robot controllers. In this problem the goal is to map a general purpose robot to specific application domains. The task of directly programming the robot to perform the given task is extremely difficult for two reasons: (1) the operator thinks about the robot's motions in terms of Cartesian space, whereas the robot's effectors are described in terms of rotations or movements of the joints, and (2) there are inherent calibration errors in the movement of the robot's effectors. In such an application it would be very desirable to design a teacher/learner pair in which a learning algorithm is selected for the robot, and then a teacher (*i.e.*, the operator) guides the robot through a "representative" set of actions to enhance the robot's speed of learning.

In previous work on developing formal models of teaching, in order to deal with the collusion issue the goal of pairing the teacher and learner has been completely sacrificed. Namely, in Goldman and Kearns [11] the learner has effectively been replaced by an adversarial learner who attempts not to learn, whereas the goal of such work is to allow the teacher and learner to work together. Jackson and Tomkins [16] attempted to return to such pairing of the teacher and learner but to prevent the most blatant of coding schemes they required that the learner must still succeed if the teacher was replaced by an adversarial substitute. While this model looks very different from the Goldman and Kearns model, surprisingly they showed that their model also required that the teacher could teach *any* consistent learner. Note that effectively both of these models bypassed the collusion issue by ensuring that the teacher provides examples to rule out all concepts that are not logically equivalent to the target concept.

One key contribution of this work is the introduction of a formal teaching model that allows the teacher and learner to truly cooperate, yet the teacher cannot simply encode the target function. In other words, unlike the previous formal models of teaching, the teacher's task is *not* reduced to that of teaching an obstinate learner that tries has hard as possible not to learn while always outputting a hypothesis consistent with all previous examples. The result is that we *can* design teacher/learner pairs as desired.

As we have discussed above, for a formal teaching model to be at all useful, it is necessary that no unnatural "collusion" between the teacher and the learner is allowed. However, formalizing what is meant by collusion is a very difficult task that all other work in this area has avoided addressing. Another important contribution of this work is that we have successfully formalized one reasonable notion of collusion. We now provide the intuition behind our formalization which appears in Section 3. To get us started, imagine a situation in which the domain is $\{0, 1\}^n$. Consider a strategy in which the teacher and learner have agreed upon some fixed enumeration of the concepts in the concept class $\mathscr{C}$. Then to teach target concept $c \in \mathscr{C}$ the teacher could simply take the binary representation r_c of c's number in the fixed enumeration. Next the teacher could break r_c into groups of n bits each and then pass each group with the appropriate label to the learner.

Why is it that almost everyone agrees that the above strategy should be forbidden? If the teacher and the learner used the same representation class and could thus agree upon a fixed enumeration of the concepts then for these problems there is no learning to be done, but rather the teacher can just tell the learner what hypothesis to output. Let us think about such a scheme in the context of our two examples from above. Suppose you (the teacher) want to train a neural network (the learner) to recognize the letter "A." While humans are good at classifying letters as "A"s or non-"A"s, we do not know the correct representation from the learner's representation class (i.e., how to set the weights in the network) and thus we could not determine the number of this unknown representation in the enumeration. Likewise, in the example of training a sensor-referenced intelligent robot controller, the primary difficulty in training the controller is that the operator (the teacher) and the controller (the learner) do not think in terms of the same representation. Otherwise, the operator could just program the controller to do the desired task.

Thus, for problems in which there is value in constructing teacher/learner pairs, the teacher and learner do not understand (or at least have a complete understanding) of the other's representation class. Furthermore, for these problems the teacher is *not* going to have the ability to give the learner any information about its representation of the target concept, but rather must rely on a careful selection of labeled examples. For example, in trying to train a neural network to recognize an "A," although we do not know how we would like the weights set, we do have ideas of what a useful set of examples (e.g., near hits and near misses) may be. Thus, intuitively one view of collusion is for the teacher to use the teaching set to pass information about the representation of the target rather than the logical function it represents. In this paper we formalize this notion of collusion and prove that any teacher/learner pair that is valid under our model does not perform any such collusion.

Our new model starts with a teacher/learner pair as in the model introduced by Jackson and Tomkins [16]. However, unlike in their work, we only require that if the teacher is replaced by an adversarial "substitute" that embeds the teaching set of the true teacher within its teaching set, then the learner will still output a hypothesis

that is logically equivalent to the target function. While the adversarial substitute has the strength to prevent collusion, it does not require that the teacher successfully teaches any consistent learner. We show that any class for which there is an efficient deterministic learning algorithm (even when provided with sophisticated queries) can be taught (without trusted information) under our model. Also the number of examples required by the teacher is at most the maximum number of mistakes made by any self-directed learning algorithm. Furthermore, using our model there is an interesting relationship between teaching and data compression. Applying the results of Floyd [8] we obtain that for any *maximum class* $\mathscr{C}$ there is a teacher/learner pair for which at most $\text{VCD}(\mathscr{C})$ examples are presented. Likewise, from the results of Helmbold, Sloan, and Warmuth [15], it follows that for any intersection-closed class $\mathscr{C}$, the nested difference of p functions from $\mathscr{C}$ can be taught in our model with at most $p \cdot \text{VCD}(\mathscr{C})$ examples. It is clear that in this paper we only scratch the surface of such results that follow from previous work of others.

In addition to the more general results, we apply our model to the representation classes of 1-decision lists and Horn sentences to demonstrate the design of teacher/learner pairs in which both the teacher and learner require only polynomial computation time. For both classes, the sample complexity is asymptotically less than that for the best known learning algorithm.

2. PREVIOUS WORK

We now briefly review the theoretical work studying the complexity of teaching. Goldman, Rivest, and Schapire [12] introduced the model of teacher-directed learning, a variant of the on-line learning model in which a helpful teacher selects the instances, and applied it to the problem of learning binary relations and total orders. Building upon this framework, Goldman and Kearns [11] defined a formal model of teaching in which they measured the complexity of teaching by the minimum number of examples that must be presented to *any* consistent learner so that the learner outputs a hypothesis logically equivalent to the target function. Independently, Shinohara and Miyano [27] introduced an equivalent notion of teachability in which a class is *teachable by examples* if there exists a polynomial size sample under which all consistent learners will exactly identify the target.

In other related work, Anthony, Brightwell, Cohen, and Shawe–Taylor [4] compute bounds on the size of the smallest sample with which only the target function is consistent for subclasses of linearly separable Boolean functions. Natarajan [21] defines a dimension measure for classes of Boolean functions that measures the complexity of a class $\mathscr{C}$ by the length of the shortest example sequence for which the target function is the unique, most specific function from $\mathscr{C}$ consistent with the sample. Salzberg, Delcher, Heath, and Kasif [25] have also considered a model of learning with a helpful teacher. Their model requires the teacher to present the shortest example sequence so that any learner using a particular algorithm (namely, the nearest-neighbor algorithm) learns the target. However, their work does not address the issue of preventing the teacher and learner from colluding. Romanik and Smith [23, 24] propose a testing problem that involves specifying, for a given target function, a set of test points that can be used to determine if a tested object is equivalent to the target. However, their primary concern is to determine for which classes there exists a finite set of instances such that any representation in the class that is consistent on the test set is "close" to the target function in a probabilistic sense.

Within the inductive inference paradigm, Freivalds, Kinber, and Wiehagen [10] and Lange and Wiehagen [17] have examined inference from "good examples." Good examples are chosen by a helpful teacher to reduce the number of examples required. In both, encoding is avoided by requiring that the inference task is accomplished even when the learner is presented with any superset of the set of teacher-chosen examples. Neither of these results, however, offer careful proof that this method actually prevents collusion between the teacher and learner. (In fact, neither of these papers really address the issue of collusion.) Lange and Wiehagen [17] examine learning pattern languages and show that this can be achieved with good examples.

Our new teaching model is most closely related to the model introduced by Jackson and Tomkins [16]. In their model there are teacher/learner pairs in which the teacher chooses examples tailored to a particular learner. To avoid collusion between the teacher and learner, they consider the interaction between the teacher and learner as a modified prover-verifier session [14] in which the learner and teacher can collude but no adversarial substitute teacher can cause the learner to output a hypothesis inconsistent with the sample. While it appears that the teacher's knowledge of the learner in this model is powerful, they showed that under their model the teacher must still produce a teaching set that eliminates all but the target function from the representation class. They also introduced the notion of a small amount of *trusted information* that the teacher can provide the learner. This trusted information is used by the teacher to provide the learner with the complexity of the target function or a stopping condition.

3. OUR MODEL

We now formally define our model. The teacher's goal is to teach the learner the target function[3] f chosen from some known *representation class* $\mathscr{C}$, which is a set of representations of functions mapping some domain $\mathscr{X}$ into $\{0, 1\}$. In

[3] Technically, the teacher is given a representation $f \in \mathscr{C}$. However, we shall equate f with the logical function it represents.

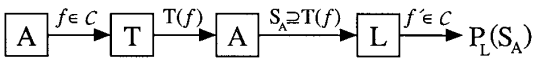

FIG. 1. An overview of a teaching session. The adversary chooses the target function, f, giving it to the teacher. The teacher then generates $T(f)$. Given $T(f)$ the adversary generates $S_A \supseteq T(f)$ and gives this to the learner. The learner (possibly randomized) outputs a representation from $\mathscr{C}$ thus defining a probability distribution $P_L(S_A)$ over $\mathscr{C}$.

addition, $\mathscr{C} = \bigcup_{n \geq 1} \mathscr{C}_n$ is often parameterized by some natural dimension measure n. Let $\mathscr{X}_n$ denote the set of instances to be classified for each problem of size n, and let $\mathscr{X} = \bigcup_{n \geq 1} \mathscr{X}_n$ denote the *instance space*. For $f \in \mathscr{C}_n$ and $x \in \mathscr{X}_n$, $f(x)$ denotes the classification of the function represented by f when evaluated on instance x. Each representation $f \in \mathscr{C}_n$ has a *size* denoted by $|f|$. Typically, this is the number of symbols (or bits) needed to write the representation of f as a member of the representation class $\mathscr{C}_n$ from which it is chosen. Finally, a *hypothesis* h is any polynomially evaluatable function that, given any $x \in \mathscr{X}_n$, outputs a prediction for $f(x)$.

A *teaching set* for $f \in \mathscr{C}$ is an unordered set of labeled instances where each instance is selected from $\mathscr{X}$ and labeled according to f. We define the teacher T to be an algorithm that when given a representation $f \in \mathscr{C}$ outputs a teaching set $T(f)$ for f. Similarly, we define the learner L to be an algorithm that takes as an argument any teaching set S and outputs a representation f' from $\mathscr{C}$. We use $L(S)$ to denote the representation output by L. (Observe that this definition can easily be extended to allow the learner to output a representation from some class $\mathscr{C}' \supseteq \mathscr{C}$). If the learner is deterministic then $L(S)$ is well defined, however, in the case that the learner uses a randomized algorithm, $L(S)$ instead induces a probability distribution over $\mathscr{C}$. We shall denote this distribution by $P_L(S)$.

We now describe our teaching protocol. The learner L and teacher T both have prior knowledge of the representation class $\mathscr{C}$ from which the target function will be selected. Furthermore, they can cooperate to develop coordinated teaching and learning strategies that best enable the teacher to teach the learner some unknown function from the class. In addition to the teacher and learner, there is an adversary A who has unlimited computing power and complete knowledge of T and L. The teaching session, illustrated in Fig. 1, proceeds as follows:

- The adversary selects a target function $f \in \mathscr{C}$ and gives f to T.
- The teacher computes teaching set $T(f)$ and gives it to A.
- Next the adversary (with knowledge of $\mathscr{C}$, f, T, and L) adds properly labeled examples to $T(f)$ with the goal of causing the learner to fail. The teaching set obtained $(S_A \supseteq T(f))$ is then given to the learner.
- Finally, the learner outputs the representation given by $L(S_A)$.

The goal of the teacher is to teach the learner to perfectly predict whether any given instance is a positive or negative instance of the target function. Thus, the learner must achieve exact (logical) identification of the target. Of course, the teacher would like to help the learner achieve this goal with the fewest number of examples possible. However, as discussed above, we must preclude unnatural "collusion" between the teacher and the learner (such as agreed-upon coding schemes to communicate information about the representation of the target function, rather than the logical function itself, via the instances selected without regard for the labels) which could trivialize our model.

We define a *valid T/L pair* for $\mathscr{C}$ to consist of a teacher T and learner L such that: For any $f \in \mathscr{C}$ the teaching set $T(f)$ output has the property that if L is provided with any teaching set $S_A \supseteq T(f)$ where all added examples are properly labeled according to f, then $P_L(S_A)$ has the property that for all $f' \in \mathscr{C}$, if f' has nonzero weight in the distribution $P_L(S_A)$ then f' is logically equivalent to f. In other words, any representation output by L will be logically equivalent to f.

Given a valid T/L pair, we say that the teacher T is a polynomial-time teacher if, given any $f \in \mathscr{C}_n$, it outputs $T(f)$ in time polynomial in n and $|f|$. Likewise, the learner L is a polynomial-time learner if it runs in time polynomial in n, $|f|$ and $|S_A|$ for any $S_A \supseteq T(f)$. We say that a representation class $\mathscr{C}$ is *T/L-teachable* if, for all $f \in \mathscr{C}_n$, there exists a valid T/L pair for which $|T(f)|$ is polynomial in $|f|$ and n. We say that $\mathscr{C}$ is *polynomially T/L-teachable* if it is T/L-teachable by a pair for which T is a polynomial-time teacher and L is a polynomial-time learner. Finally, we say that $\mathscr{C}$ is *semi-poly T/L-teachable* if it is T/L-teachable with a polynomial-time learner but a teacher that may be computationally unbounded.

In the next section we argue that the adversarial substitute is sufficient to prevent collusion. As we discussed in the introduction, intuitively one view of collusion is for the teacher to use the teaching set to pass information about the representation of the target versus the function it represents. We formalize this notion in the following manner. We define a *colluding T/L pair* for $\mathscr{C}$ to consist of a teacher T and learner L such that: There exist logically equivalent f_0, f_1 in $\mathscr{C}$ (possibly with $f_0 = f_1$) such that for all $S_0 \supseteq T(f_0)$ and all $S_1 \supseteq T(f_1)$, $P_L(S_0) \neq P_L(S_1)$. In other words, if the teacher is able to influence the distribution on $\mathscr{C}$ returned by L when presented with teaching sets for logically equivalent representations then there is collusion between T and L. Clearly there are other ways in which one could formalize the notion of collusion. However, the definition that we have chosen (and will use throughout the remainder of this paper) is one reasonable definition of collusion that also captures the type of interaction between the teacher and learner that can occur in the domains for which such teacher/learner pairs are meaningful. In the next section we argue that the adversarial substitute is sufficient to prevent this form of collusion.

4. GENERAL RESULTS

In this section we explore the properties of our new teaching model. We begin by proving that the adversarial substitute is sufficient to prevent collusion.

THEOREM 1. *There is no colluding T/L pair.*

Proof. We use a proof by contradiction. Suppose there exists a colluding T/L pair. Since the T/L pair is colluding, by definition there exist logically equivalent f_0, f_1 in $\mathscr{C}$ (possibly with $f_0 = f_1$) such that for all $S_0 \supseteq T(f_0)$ and all $S_1 \supseteq T(f_1)$, $P_L(S_0) \neq P_L(S_1)$. However, let $S_0 = S_1 = T(f_0) \cup T(f_1)$. Then, since $S_0 = S_1$ it follows that $P_L S_0 = P_L(S_1)$ giving the desired contradiction. ▌

It immediately follows from this theorem, that any valid T/L pair must not collude. Notice that, unlike the Jackson and Tomkins model in which the teacher uses "trusted bits" (in addition to labeled examples) to successfully teach many classes, our teacher must teach the target function entirely through a careful selection of labeled examples.

While technically not collusion as we have defined it, there is a type of cooperation between the teacher and learner, allowing the transmission of information about the target, that merits discussion. Let $\mathscr{C}$ be a representation class over the domain $\mathscr{X} = \{0, 1\}^n$ with no more than $2^{O(n)}$ representations and let the teacher and learner agree upon a fixed enumeration of $\mathscr{C}$. Thus, each representation f can be identified by a constant number of instances encoding the place of f in the enumeration. To teach such a class, the teacher simply places the instances $\{x_1, ..., x_k\}$ identifying f, along with their labels according to f, in its teaching set. Additionally, the teacher must add to the teaching set a set of examples that uniquely identify the function represented by f (*i.e.*, a teaching sequence as defined by Goldman and Kearns [11]). The learner chooses an ordered k-tuple of instances from the teaching set and identifies its associated $f \in \mathscr{C}$. If every instance in the teaching set is labeled according to f then f is the target function and is output by the learner. Otherwise, the learner continues choosing ordered k-tuples until the target is identified in this manner.

We now demonstrate that the above method is not collusion according to the definition we have given. Let $f_1, f_2 \in \mathscr{C}$ such that for all $x \in \mathscr{X}$, $f_1(x) = f_2(x)$. Without loss of generality, if the teacher attempts to teach f_1 then the adversarial substitute can simply add the k-tuple for f_2, with their labels, to the teaching set. Then both f_1 and f_2 are consistent with the entire teaching set and either may be output by the learner. Thus, no collusion is occurring. Intuitively, we do not feel that collusion is occurring since the teaching set contains examples to *uniquely identify* the function represented by the target representation. However, the learner may have to solve a hard consistency problem to identify the target without using the above scheme. So, this T/L pair relies on the power of the teacher's knowledge of the learner to help the learner solve a hard computational problem.

We now show that any representation class learnable in deterministic polynomial time from "example-based" queries (including equivalence, membership, subset, superset, disjointness, exhaustiveness, justifying assignments, partial equivalence) is teachable in our model by a computationally unbounded teacher and a polynomial-time learner. We define an *example-based query* to be any query of the form

$$\forall (x_1, x_2, ..., x_k) \in \mathscr{X}^k \qquad \text{does} \quad \varphi_f(x_1, x_2, ..., x_k) = 1?,$$

where k is constant and $\varphi_f(x_1, x_2, ..., x_k)$ is any poly-time computable predicate with membership-query access to the target f. Observe that the predicate φ may use the x_i's to compute other instances on which to perform membership queries. The answer provided to the example-based query is either "yes" or a counterexample consisting of $(x_1, x_2, ..., x_k) \in \mathscr{X}^k$ (with their labels) for which $\varphi_f(x_1, x_2, ..., x_k) = 0$ and the examples (and their labels) for which membership queries were made to evaluate the predicate. In Fig. 2 we give the definition for the predicate φ_f corresponding to queries in standard use. Observe that all reasonable, and some fairly bizarre, queries can be formulated in this manner.

Type of Query	k	$\varphi_f(x_1, \ldots, x_k)$
equivalence(h)	1	$h(x_1) = f(x_1)$
membership(x)	0	$f(x)$
subset(h)	1	$h(x_1) = 1 \Rightarrow f(x_1) = 1$
superset(h)	1	$f(x_1) = 1 \Rightarrow h(x_1) = 1$
disjointness(h)	1	$(f(x_1) = 1 \Rightarrow h(x_1) = 0) \wedge (h(x_1) = 1 \Rightarrow f(x_1) = 0))$
exhaustiveness(h)	1	$(f(x_1) = 0 \Rightarrow h(x_1) = 1) \wedge (h(x_1) = 0 \Rightarrow f(x_1) = 1))$
justifying-assign(v_i)	1	$f_{v_i \to 0}(x_1) = f_{v_i \to 1}(x_1)$
partial-equivalence(h)	1	$(h(x_1) = f(x_1)) \vee (h(x_1) \neq *)$

FIG. 2. We show how to represent various queries as example-based queries. Equivalence, membership, subset, superset, disjointness, and exhaustiveness queries are defined by Angluin [3]. A *justifying assignment* for an input variable is an instance whose classification changes if the value of the variable is changed. Thus, for Boolean domains, a justifying assignment query on v_i returns "yes" if there is no justifying assignment, or as a counterexample it returns two instances that provide a justifying assignment for v_i. (The notation $f_{v_i \to 0}$ denotes the function obtained from f by fixing $v_i = 0$.) Finally, in a partial equivalence query (as defined by Maass and Turán [19]) the learner can present a hypothesis h: $\mathscr{X} \to \{0, 1, *\}$ and is either told that all specified instances are correct or is given an $x \in \mathscr{X}$ such that $h(x) \in \{0, 1\}$ and x is misclassified by h.

THEOREM 2. *Any representation class $\mathscr{C}$ learnable in deterministic polynomial-time using example-based queries is semi-poly T/L-teachable.*

Proof. We prove this result by demonstrating a valid T/L pair for any representation class $\mathscr{C}$ learnable in deterministic polynomial-time by an algorithm $\mathscr{A}$ that uses only example-based queries. We assume there is some total ordering π on $\mathscr{X}$ (such as a lexicographical order) upon which the learner and teacher have agreed. Given any two sets of k instances from $\mathscr{X}$, we define the following ordering among them. Let $x=(x_1, ..., x_k)$ for $x_1<x_2<\cdots<x_k$ be one set and let $y=(y_1, ..., y_k)$ for $y_1<y_2<\cdots<y_k$ be the other set. Then we say that $x<y$ (according to π) if there exists a $1\leqslant j\leqslant k$ such that $x_j<y_j$ and for all $i<j$, $x_i=y_i$.

The teacher T constructs its teaching set $T(f)$ for f as follows. Initially, let $T(f)=\varnothing$. Now T simulates $\mathscr{A}$'s execution until the point at which the first example-based query q is performed. Let k be the number of instances over which q is quantified. The teacher now goes through all $(x_1, x_2, ..., x_k)\in\mathscr{X}^k$ from smallest to largest, evaluating $\varphi(x_1, ..., x_k)$. If there are no counterexamples to q then the teacher replies "yes" to $\mathscr{A}$'s query. Otherwise, let $(x_1, x_2, ..., x_k)$ be the smallest counterexample and let $q_1, ..., q_\ell$ be the instances on which membership queries were made to evaluate $\varphi(x_1, ..., x_k)$. Note that the number of membership queries made by φ_f (and thus ℓ) is polynomial since φ_f is poly-time computable. The teacher lets $S=\{x_1, f(x_1)\}\cup\cdots\cup\{x_k, f(x_k)\}\cup\{q_1, f(q_1)\}\cup\cdots\cup\{q_\ell, f(q_\ell)\}$, replies to $\mathscr{A}$ with S, and updates $T(f)$ to be $T(f)\cup S$. The teacher continues in this manner until $\mathscr{A}$ halts.

We now create the learner L from $\mathscr{A}$ as follows. Let $S_A\supseteq T(f)$ be the teaching set that the learner receives. Whenever $\mathscr{A}$ makes a query q, the learner will proceed as follows. The learner will consider all k-tuples in S_A from smallest to largest. For each such tuple $(x_1, ..., x_k)$ the learner attempts to evaluate $\varphi(x_1, ..., x_k)$. In order to evaluate φ, recall that the learner may need to perform some additional membership queries. If these instances appear in S_A then the learner computes $\varphi(x_1, ..., x_k)$. If $\varphi(x_1, ..., x_k)=0$, then L gives $\mathscr{A}$ the k-tuple $(x_1, ..., x_k)$ along with the labeled examples corresponding to the membership queries made in evaluating φ on this k-tuple. What if the learner is unable to evaluate φ? Since $T(f)$ contained the minimum counterexample for φ (including all instances needed to evaluate φ) and $S_A\supseteq T(f)$, it follows that $(x_1, ..., x_k)$ must not be a counterexample. Thus, if the learner computes that $\varphi(x_1, ..., x_k)=1$ or is unable to evaluate it, then the learner continues with the next k-tuple in the ordering. If for all k-tuples in S_A the predicate φ is true or unevaluatable (i.e., there is no counterexample to q in S_A) then L responds to $\mathscr{A}$ with "yes." Observe that since q is quantified over a constant number of instances, in time polynomial in $|S_A|$ the learner can consider all k-tuples from S_A. Furthermore, because the teacher evaluated k-tuples of instances from minimum to maximum when constructing $T(f)$, the membership queries needed to evaluate φ will be present for the minimum counterexample.

We now argue that L will halt in polynomial time and output f. The key observation here is that the teacher's and learner's simulations of $\mathscr{A}$ always remain the same. Since $\mathscr{A}$ is deterministic its execution is altered only by the responses given to its queries. While the adversarial substitute may add other counterexamples, the learner will always find the minimum one, which was included in $T(f)$ by T, and thus T and L both give $\mathscr{A}$ exactly the same counterexamples. ∎

We say that class $\mathscr{C}$ is *learnable in polynomial time using example-based queries* if there exists an algorithm using only a polynomial number of example-based queries, running in time polynomial in the relevant size measures of $\mathscr{C}$ and returning a hypothesis that is logically equivalent to the target concept $f\in\mathscr{C}$ (that is, $h(x)=f(x)\ \ \forall x\in\mathscr{X}$). The following corollary follows directly from Theorem 2.

COROLLARY 3. *If representation class $\mathscr{C}$ is not semi-poly T/L-teachable then it is not deterministically learnable in polynomial time using example-based queries.*

Thus, negative results obtained for a class in our model give very strong negative results with regards to the learnability of the class. It is possible that this correspondence will provide new techniques to prove hardness results for learning. As an immediate consequence of this result we know that many classes (namely, all of those for which exact-identification is efficiently achieved with queries) are T/L-teachable with an efficient learner. In particular, this contrasts the negative result of Jackson and Tomkins [16] that the class of 1-decision lists is not teachable without trusted information, and the negative result of Anthony *et al.* [4] that linearly separable Boolean functions are not efficiently teachable. In fact, Bshouty's [7] result that arbitrary decision *trees* are learnable with membership and equivalence queries implies that a much broader class than 1-decision lists is T/L-teachable with a polynomial-time learner.

Letting $\mathscr{A}$ in the proof of Theorem 2 be the halving algorithm [5, 18], we immediately get the following corollary.

COROLLARY 4. *Any representation class $\mathscr{C}$ is T/L-teachable (by a computationally unbounded teacher and learner) with a teaching set of length at most* $\log|\mathscr{C}|$.

Because our model incorporates a very powerful set of queries, classes that may not be learnable using membership and equivalence queries are T/L-teachable. In particular, from Angluin's result [3] that pattern languages can be exactly identified in polynomial time using only restricted

superset queries, we get that pattern languages are semi-poly T/L-teachable. Lange and Wiehagen independently presented an algorithm to learn pattern languages from good examples [17].

It is also easily shown that the self-directed learning model [13] can be simulated in our model. We now give the definition of the self-directed learning model. A *query sequence* is a permutation $\pi = \langle x_1, x_2, ..., x_{|\mathscr{X}|} \rangle$ of the instance space $\mathscr{X}$, where x_t is the instance the learner will predict at the tth trial. The learner may build its query sequence in an on-line manner. Namely, for the tth trial, x_t may be selected by any polynomial time algorithm that takes as input the set of labeled examples obtained in the first $t-1$ trials. The *mistake bound* for learning concept c with a given self-directed learning algorithm is the number of incorrect predictions made during the learning session. We define the mistake bound of a self-directed learning algorithm for $\mathscr{C}$ to be the maximum of the mistake bounds for each concept $c \in \mathscr{C}$.

We now show that the number of examples in an optimal teaching set is at most the number of mistakes made under the self-directed learning model. Namely, we can build a valid T/L pair by having T simulate the self-directed learning algorithm $\mathscr{A}$, and then include in $T(f)$ only those examples that $\mathscr{A}$ misclassifies. Then L can simply simulate $\mathscr{A}$ assuming that any example not in $S_A \supseteq T(f)$ is properly labeled by its hypothesis.

THEOREM 5. *If there is a deterministic self-directed learning algorithm for representation class $\mathscr{C}$ that makes polynomially bounded mistakes, then $\mathscr{C}$ is T/L-teachable. Furthermore, the number of examples in an optimal teaching set is at most the number of mistakes made by the self-directed learning algorithm.*

Thus, for example, it follows that the classes of monomials and axis-parallel rectangles in $\{0, 1, ..., n-1\}^d$ are both T/L-teachable using only two examples. Furthermore, for these classes it is easily shown that both the teacher and learner can be efficiently implemented.

5. EFFICIENT TEACHING STRATEGIES

While we know from Theorem 2 that any representation class learnable by an algorithm using example-based queries is T/L-teachable, we now demonstrate that often both the learner and the teacher use polynomial time and that for many classes the number of examples in $T(f)$ is asymptotically less than the number of queries made by existing learning algorithms. While we have currently only designed good teachers for existing learning algorithms, in the long run we expect to design new learning algorithms that may not be good against an adversarial environment, but work very well when paired with an appropriate teacher.

In this section we give polynomial T/L pairs for the classes of decisions lists and Horn sentences. Jackson and Tomkins show that the class of 1-decisions lists with no irrelevant variables is teachable in their model. They prove, however, that 1-decision lists are not teachable in their model without trusted information.

5.1. *Decision Lists*

As defined by Rivest [22], a 1-decision list (1-DL) over the set $V_n = \{v_1, v_2, ..., v_n\}$ of n Boolean variables is an ordered list of nodes $f = \langle n_1, ..., n_r \rangle$, where node n_i (for $1 \leqslant i \leqslant r-1$) is the pair (ℓ_i, b_i), where ℓ_i is v_i or $\overline{v_i}$ for $v_i \in V_n$ and node n_r is the pair (TRUE, b_r), where $b_1, ..., b_r \in \{0, 1\}$. We refer to a node of the form (TRUE, b_r) as a *constant node*. For an instance $x \in \{0, 1\}^n$, we define $f(x) = b_j$, where $1 \leqslant j \leqslant r-1$ is the least value such that ℓ_j is 1 in x; $f(x) = b_r$ if there is no such j, where b_r is the bit associated with the constant node. Note that a constant node always terminates a decision list. One may think of a decision list as an extended "**if–then–elseif**–$\cdots$ **else**" rule. Note that the final **else** clause is equivalent to the constant node defined above. Rivest also defines the class of k-decision lists (k-DL) as the generalization of 1-decision lists in which ℓ_i is any conjunction of at most k literals from V_n. Rivest presents an algorithm to learn the class of k-decision lists in the PAC model, and later Nick Littlestone[4] constructed an algorithm to exactly identify k-DLs using only equivalence queries. When applied to the class of 1-DLs, Littlestone's algorithm uses $O(rn)$ equivalence queries and $O(rn^2)$ time.

For a 1-DL, f, let (ℓ_i, b_i) and (ℓ_j, b_j) be two nodes in f for $i < j$. If $\ell_i = \overline{\ell_j}$ then a logically equivalent decision list is obtained by replacing node j by the constant node (TRUE, b_j) and thus effectively removing all the following nodes. Also, if $\ell_i = \ell_j$ then node j can be removed. We say a *reduced* 1-DL is one to which these reductions have been applied. Let r' be the number of nodes in the target decision list (before reducing). The above reductions can be applied in $O(n + r')$ time by simply maintaining a table of the potential nodes (there are at most $4n + 2$ of these). Then during a scan of the target decision list place a 1 in the table entry for a node the first time it appears. The reduced target decision list is constructed as the original is being scanned (with appropriate action when a variable is repeated). Note that the number of nodes r in a reduced decision list is at most $n + 1$ (this includes the constant node at the end). We now show that the class of 1-DLs is polynomially T/L-teachable.

THEOREM 6. *The class of 1-DLs is polynomially T/L-teachable with a teaching set of length at most $2r$, where r is the number of nodes in the reduced target decision list. The teacher requires $O(rn + r')$ time to generate $T(f)$, where r'*

[4] This result is unpublished and may have been discovered independently by others.

is the number of nodes in the target decision list before reductions are applied, and the learner requires $O(r^2n)$ time when given $T(f)$ as input.

Proof. We informally describe the teacher and learner for this class. The first step of the teacher is to reduce the target decision list as described above. Let $X_b^{(\ell_i)}$ be the example for which literals $\ell_1, ..., \ell_{i-1}$ are 0, literal ℓ_i is 1 and all remaining variables are assigned b, for b either 0 or 1. For $1 \leqslant i \leqslant r-1$, the teacher simply includes $X_0^{(\ell_i)}$ and $X_1^{(\ell_i)}$. Note that for a constant node only a single example is included. In this example literals $\ell_1, ..., \ell_r$ are 0 and all other variables are set to 0. Thus, the teacher generates only $\Theta(r)$ examples, each requiring $O(n)$ time to generate.

The ideas in the design of our learner are based on the algorithm given by Littlestone to exactly identify 1-DLs using only equivalence queries. It is convenient to view target decision list f as "leveled" where each level contains consecutive nodes that have the same associated bit and level 1 is the top level. Thus the number of levels in f is exactly one plus the number of times that $b_i \neq b_{i+1}$. The learner initially assumes that all of the $4n+2$ possible nodes are the first (or top) node in f. Then on example x, any node that would have caused x to be mislabeled is moved to the set of candidates for the second level of f. So, in general, L maintains a set of candidates for all $n+1$ possible levels of f. For a given example x, the learner finds the highest level (least index) containing some node (ℓ_i, b_i) for which $\ell_i = 1$ in x. Each node (ℓ_j, b_j) in this level such that $b_j \neq f(x)$ is moved down to the next level by L. Finally, when no such counterexample exists, L outputs a 1-DL by arbitrarily ordering the nodes within each level, stopping at the earliest level in which a constant node appears. See Fig. 3 for a complete description of the teaching and learning algorithms.

T/L pair for 1-DL:

Teacher:

Reduce the target DL

$T(f) \leftarrow \emptyset$

For each node i in the reduced target DL

$T(f) \leftarrow T(f) \cup \{X_0^{(\ell_i)}\} \cup \{X_1^{(\ell_i)}\}$

Learner:

All candidate nodes begin at level 1

Repeat

For each example $(x, f(x))$ in S_A

Let k be the highest level for which $\ell_i = 1$ in x, for (ℓ_i, b_i) at level k

For each candidate node i at level k

If $\ell_i = 1$ in x and $b_i \neq f(x)$ then increment level of node i

Until no candidate changes level

Output the 1-DL obtained by arbitrarily ordering the nodes within each level, stopping when reaching a constant node

FIG. 3. Algorithms for the teacher and learner for the class of 1-DL.

We now argue that this is a valid T/L pair and that L runs in $O(r^2n)$ time. We use induction to show that after the k th iteration of the repeat loop all candidates that belong on level $j \geqslant k+1$ have been bumped to at least level $k+1$. Note that any literal reaching level $r+1$ is irrelevant. The inductive hypothesis holds for $k=0$ (i.e., at the beginning all nodes are at level 1). Given that all nodes belonging at level $j \geqslant k$ are at level k or greater after the $(k-1)$ th iteration of the loop, we must show that after the k th iteration the nodes that belong at level $j \geqslant k+1$ will be at level $k+1$ or greater. Let $(\ell_i, b_i), ..., (\ell_j, b_j)$ be the nodes of f that are at level k and assume, for ease of exposition, that for $i \leqslant t \leqslant j$, $b_t = 1$. Observe that $X_0^{(\ell_t)}$ and $X_1^{(\ell_t)}$ for $i \leqslant t \leqslant j$ bump all candidates that do not belong in level k, except for those with an associated bit of 1 (including the constant 1) and nodes of the form $(\bar{\ell}_t, 0)$ for $i \leqslant t \leqslant j$. For now, assume that $k+3 \leqslant r$, and for $s = k+1$ or $k+3$, let $(\ell_s, 0)$ be any node in f at level s. Of the nodes that still must be bumped from level k, observe that $X_0^{(\ell_{k+1})}$ and $X_1^{(\ell_{k+1})}$ bump all but $(\bar{\ell}_{k+1}, 1)$ and all nodes of the form $(\bar{\ell}_t, 0)$ for $i \leqslant t \leqslant j$. Finally, these remaining nodes are bumped by $X_b^{(\ell_{k+2})}$ and $X_b^{(\ell_{k+3})}$ for $b \in \{0, 1\}$, where $(\ell_{k+2}, 1)$ is a node at level $k+2$ in f. In the case that $k+3 > r$ it is easily shown that the candidates that do not belong at level k are bumped by the examples associated with node r. Finally, notice that instances placed in S_A by the adversary will not affect the correctness of the hypothesis returned by the learner. ∎

This result can be generalized to k-DL but to do so the teacher must present examples for node i that turn off all nodes above node i, turn on node i and individually turn off nodes below node i (rather than en masse as is done for 1-DL). This increases the number of examples required to $O(n^k)$ and increases the running time of the learner correspondingly. The running time of such a teacher is no longer polynomial, assuming $P \neq NP$, since implicit in turning off all nodes above node i is a satisfiability problem that the teacher must solve.

5.2. *Conjunctions of Horn Clauses*

A Horn clause is a disjunction of literals at most one of which is unnegated. A Horn sentence is a conjunction of Horn clauses. Angluin, Frazier, and Pitt [2] gave a polynomial-time algorithm to exactly identify an m-clause Horn sentence using $O(mn)$ equivalence queries, $O(m^2n)$ membership queries, and $\tilde{O}(m^2n^2)$ time,[5] where n is the number of variables in the instance space. Note that each Horn clause can be viewed as a logical implication in which the consequent contains the, at most one, unnegated variable.

A clause C of the Horn sentence f is violated by x when all variables in the antecedent (denoted ant(C)) are 1 in x

[5] The "soft-oh" notation is like the standard "big-oh" except that log factor are also ignored.

and the variable in the consequent (denoted cons(C)) is 0 in x. For clause C, where C is not a tautology, let N_C denote the minimum (i.e., with the fewest variables set to 1) negative example that violates only clause C. Namely, N_C is the example that violates C (i.e., the variables in ant(C) are set to 1 and cons(C) is set to 0), where all the variables not in C are set to 0 unless this causes some other clause to be violated. (We will give a procedure for generating such examples.) When C is a tautology, there is no such minimum negative example and thus there is no corresponding N_C. Likewise when cons(C) $\neq$ FALSE, let P_C be the minimum positive example for which all variables in ant(C) and cons(C) are 1 (each remaining variable is set to 0 unless this causes some clause in f to be violated). When cons(C) = FALSE there is no corresponding P_C.

It is possible that for some clause C such that cons(C) $\neq$ FALSE there is no positive instance in which all of the variables in ant(C) and cons(C) are set to 1. Consider the example

$$f = (ab \Rightarrow c) \wedge (ac \Rightarrow d) \wedge (bd \Rightarrow e) \wedge (de \Rightarrow \text{FALSE})$$

and let clause $C = (ab \Rightarrow c)$. For P_C we want the minimum positive instance in which $a, b, c = 1$. However, setting a, b, and c to 1 requires that d is set to 1 or the second clause will be violated. Then e must be set to 1 or the third clause will be violated. Now, however, both d and e are set to one and the fourth clause is violated. Thus, for clause C, there is no instance P_C. In Lemma 7 we argue that the lack of a positive counterexample in this case (and the case where cons(C) = FALSE) does not hinder the learner.

LEMMA 7. *If no P_C exists for clause C then the "extra" clauses added to the learner's hypothesis when N_C was processed have no effect on the logical meaning of the hypothesis.*

We delay the proof of this lemma until after a discussion of the main result of this section and a description of our algorithms for the teacher and learner.

THEOREM 8. *The class of Horn sentences is polynomially T/L-teachable using 2m examples, where m is the number of clauses in the representation of the target Horn sentence given to the teacher.*[6] *The teacher requires $O(m^2n + m^3)$ time to generate teaching set $T(f)$ and the learner requires $O(m^2n)$ time given $T(f)$ as input.*

Proof. For each clause C the teacher generates the negative example N_C and the positive example P_C. To generate all of the N_C the teacher uses the following algorithm: Until no clause is changed, repeatedly look for a pair of clauses C_i and C_j for which the antecedent of C_i is a *proper* subset of the antecedent of C_j. If this is the case, replace C_j by (ant(C_j) $\wedge$ cons(C_i) $\Rightarrow$ cons(C_j)). The resultant target f' is logically equivalent to the original target f and has the same number of clauses. Now, for each non-tautological clause C_i, N_{C_i} is simply the example in which the variables in ant(C_i) are set to one and all other variables are set to 0. To see that this N_{C_i} is a minimum negative example that violates only C_i, note that any other clause C_j, such that ant(C_j) is satisfied by N_{C_i}, is not violated by N_{C_i} since cons(C_j) is also included in ant(C_i). Note that if C_i is tautological it has no effect on the logical meaning of f and is, therefore, deleted by the teacher. Thus, this procedure allows the teacher to provide the minimum negative example for each clause of f such that this example will be a negative counterexample to the learner's hypothesis regardless of when the learner sees it.

We now describe an algorithm to compute P_C. To solve this problem we formulate it as the following graph problem. We construct a graph that has one vertex corresponding to the antecedent of each clause except C, one vertex corresponding to each variable not in the consequent or antecedent of clause C plus vertices v_T and v_F corresponding to the constants TRUE and FALSE. For each clause other than C we temporarily turn on variables that are in the consequent or antecedent of C. Now we place a directed edge from each variable vertex to the vertices corresponding to all of the antecedents in which that variable appears. Additionally, we place a directed edge from each antecedent vertex to the vertex that is the consequent of that clause. Finally, for any clause $C' \neq C$ for which ant(C') = TRUE, we place the edge (v_T, ant(C')) in the graph. Observe that we can build the adjacency matrix representation of this graph G in $O(mn)$ time. We will maintain an array of in-degrees that can be initialized (also in $O(mn)$ time) as G is built. We also make use of an *outlist* for v_T and for each of the variable vertices as well as a bit vector *contracted* that contains a bit, set to 1 when that vertex is contracted into v_T, for each variable vertex. Note that the graph constructed is bipartite: all edges are between an antecedent vertex and one of the other vertices. (It is possible to have an edge from a variable vertex to v_T but such edges are never considered by our algorithm and are, therefore, irrelevant.) The out-degree of every antecedent vertex is 1 and the out-degrees of the variable vertices are at most m since there are only m antecedent vertices.

Our goal now is to assign as many vertices as possible in G to be 0 with the restriction that if v is an antecedent vertex then if *all* incoming edges are set to 1 (i.e., the corresponding antecedent is true) then the vertex pointed to by v must be set to 1 (otherwise, some clause would be violated since an antecedent would be true and the consequent false). Note that no variables must be set to 1 unless there is some antecedent vertex, in the initial graph constructed, that is

[6] We note that in time polynomial in m and n the teacher can find a logically equivalent Horn sentence that has at most $m'n$ clauses, where m' is the minimum number of clauses in any logically equivalent Horn sentence. To do this the teacher simply runs the algorithm by Angluin, Frazier, and Pitt before generating its teaching set. This ratio bound appears in Frazier's thesis [9].

T/L pair for Horn sentences:

Teacher:

$T(f) \leftarrow \emptyset$
Repeat until no clause is changed
For all pairs of clauses in the target
If $ant(C_i) \subset ant(C_j)$
then replace C_j by $(ant(C_j) \wedge cons(C_i) \Rightarrow cons(C_j))$
For $i \leftarrow 1$ *to* m
If C_i is a tautology
then delete C_i
else
$N_{C_i} \leftarrow ant(C_i)$ set to 1 and all other vars 0
$P_{C_i} \leftarrow$ result of graph algorithm for C_i
$T(f) \leftarrow T(f) \cup \{P_{C_i}\} \cup \{N_{C_i}\}$

Learner:

$h \leftarrow$ **true**
Repeat
While S_A contains a positive counterexample
delete from h all clauses violated
If S_A contains a negative counterexample
then choose one, not previously used, with fewest 1's
let the 1's in the counterexample define $ant(C_i)$
$h \leftarrow h \wedge (\bigwedge_{(j \mid v_j \notin ant(C_i))}(ant(C_i) \Rightarrow v_j)) \wedge ant(C_i) \Rightarrow$ FALSE
Until S_A contains no counterexamples

FIG. 4. Algorithms for the teacher and learner for the class of Horn sentences.

pointed to only by v_T. Such vertices are *contracted* into v_T as are the consequents to which they point. Each contracted variable must be set to 1. The algorithm to determine which variables must be set to 1 is as follows. Scan the outlist of v_T. If any antecedent vertex v_a pointed to by v_T has in-degree 1 then check if the consequent v_c pointed to by v_a has already been contracted. If v_c has been contracted then simply remove v_a from the outlist of v_T. Otherwise, contract v_a and the consequent v_c to which it points into v_T and set contracted $[v_c]$ to 1. Then update the outlist of v_T with the *bitwise-or* of the outlists of v_T and v_c and subtract 1 from the in-degree of each vertex pointed to by both v_T and v_c. Repeat these steps until the in-degrees of all of the antecedent vertices pointed to by v_T are greater than 1. A special case that we must consider is that the v_c being contracted into v_T is v_F. Clearly we cannot contract v_F into v_T since this leads to the contradiction that FALSE = TRUE. Thus, if at any time v_F is to be contracted with v_T the algorithm simply returns $P_C = \emptyset$.

All vertices contracted with v_T will be assigned the value 1 and the remaining vertices in G will be assigned the value 0. Let V' be the set of remaining vertices (i.e., those not contracted with v_T). Observe that all antecedent vertices in V' have at least one incoming edge from a variable vertex in V' and, thus, by setting all variable vertices in V' to 0 no clauses will be violated. Finally, we argue that this procedure takes $O(m^2)$ time. To see this first observe that there are at most m contractions that can take place. Next, observe that each contraction can be performed in $O(m)$ time. It takes $O(m)$ time to scan the outlist of v_T. Let v_a be the node being considered for contraction with v_T and v_c be the consequent pointed to by v_a. The checks of the in-degree of v_a and the contraction status of v_c take constant time. Finally, the bitwise or of the outlists of v_T and v_c and updating the in-degrees of the vertices pointed to take $O(m)$ time. The time to generate P_C for a single clause C is, therefore, $O(m^2)$.

Thus, the time to construct all of the N_C is $O(m^2 n)$ and the time to construct all of the P_C is $O(m^3)$. The total time required by the teacher is $O(m^2 n + m^3)$. The length of the teaching set output is at most $2m$. Finally, the learner runs what is essentially the standard algorithm for learning Horn sentences [2]. Figure 4 summarizes the algorithms of T and L.

We now show that this learner and teacher are a valid T/L-pair. Each negative example, N_C, added to $T(f)$ by the teacher causes the learner to add one of the clauses of the target to its hypothesis (as well as some additional clauses that may be subsumed by the target and some additional clauses that would cause false negative errors). Observe, that any hypothesis of the learner that includes the additional clauses subsumed by the target will still be logically equivalent to the target (and as discussed below, the number of such "extra" clauses is polynomial in the size of the target). Furthermore, it is easily shown that P_C will violate any of the additional clauses that would cause false negative errors, thus causing the learner to remove them. In the case that for some clause C there is no P_C, Lemma 7 shows that all extra clauses due to C are subsumed by the target. Finally, there are $O(n)$ such extra clauses (including both types) for each true clause added. Since there are m clauses in the target, a total of $O(mn)$ clauses are added to the learner's hypothesis. Thus, given $T(f)$ as input, L runs in $O(m^2 n)$ time and returns a hypothesis that is logically equivalent to f. Finally, since L selects the minimum negative counterexample at each step, it follows that any negative example placed in S_A by the adversary will not alter the learner's course. Any positive counterexamples placed in S_A by the adversary are easily shown to cause no harm. ▌

We now restate and prove Lemma 7.

LEMMA 7. *If no P_C exists for clause C then the "extra" clauses added to the learner's hypothesis when N_C was processed have no effect on the logical meaning of the hypothesis.*

Proof. Since there is no P_C for clause C then either it is the case that $C = (\text{ant}(C) \Rightarrow \text{FALSE})$ or that at some point the algorithm for finding P_C attempted to contract v_F and v_T. When C was added to the learner's hypothesis, h, additional

clauses of the form $\text{ant}(C) \Rightarrow \text{FALSE}$ and $\text{ant}(C) \Rightarrow v_j$, $\forall v_j \notin \text{ant}(C)$ are also added. Let S_C denote the set of these additional clauses.

For clauses of the form $\text{ant}(C) \Rightarrow \text{FALSE}$, it is easy to see that none of the clauses in S_C change the logical meaning of h. Assume that $\text{ant}(C) \Rightarrow \text{FALSE}$ is not clause C. Then there must be some "logical chain" in f from C to FALSE as in the example discussed earlier in this section. That is, when $\text{ant}(C)$ and $\text{cons}(C)$ are both true then the antecedent of some clause C_j is also true where $\text{cons}(C_j) = \text{FALSE}$. Denote this clause C_F. We show that the clauses in S_C have no effect on the logical meaning of the learner's **final** hypothesis in this case. Let x be an arbitrary instance from $\mathcal{X}$:

- *Case* 1. C is violated by instance x. Then $h(x) = 0$ and the clauses of S_C are irrelevant with respect to x.
- *Case* 2. C is not violated by instance x.

— Case A. $\text{ant}(C)$ is satisfied by instance x. Since $\text{ant}(C)$ is true in x but C is not violated by x, $\text{cons}(C)$ is also true in x. Then, by the logical chain from C to FALSE, C_F is violated and $h(x) = 0$. Thus the clauses in S_C are irrelevant with respect to x.

— Case B. $\text{ant}(C)$ is false in instance x. Then clause C is true. All of the clauses in S_C are also true since they have the same antecedent as C.

Thus, we have shown that for all instances $x \in \mathcal{X}$, $h(x)$ is not affected by the presence of the clauses in S_C. ▌

The class of k-quasi Horn sentences is defined similarly to Horn sentences except that each clause can have at most k unnegated literals (consequents of size at most k). Angluin, Frazier, and Pitt note that learning 2-quasi Horn sentences with membership queries is as hard as learning CNF formulas with membership queries. With the added power of a cooperative teacher, however, it is unclear if there exists a valid T/L pair for this class. A straightforward adaptation of the above T/L pair cannot be used to teach 2-quasi Horn sentences. The problem lies in attempting to alter the target so that the N_C examples can be generated. Because each consequent may have size two there can be exponential blowup in the number of clauses in the resultant target function. Attempts at different approaches encountered similar difficulties (such as an exponential number of examples required to teach a polynomial number of clauses in an effort to anticipate membership queries). The semi-poly T/L-teachability of 2-quasi Horn sentences remains an interesting open problem.

6. RELATION TO DATA COMPRESSION

In this section we uncover an interesting relationship between teaching in our model and data compression. A *data compression* scheme of size k for representation class $\mathscr{C}$ consists of a *compression algorithm* $\mathscr{F}$ and a *reconstruction algorithm* $\mathscr{G}$. Let S_m be any subset of $m \geqslant k$ examples from $\mathcal{X}$ labeled according to some $f \in \mathscr{C}$. The compression algorithm $\mathscr{F}$ takes as input S_m and outputs some subset S of S_m such that $|S| \leqslant k$. The reconstruction algorithm $\mathscr{G}$ takes as input any possible subset of at most k labeled examples and outputs a hypothesis h on $\mathcal{X}$. A *valid data compression scheme* of size k for $\mathscr{C}$ consists of a pair $\mathscr{F}$ and $\mathscr{G}$ such that for any $f \in \mathscr{C}$ and any set S_m of at least k examples labeled according to f, the hypothesis output by $\mathscr{G}(\mathscr{F}(S_m))$ must be consistent with all examples in S_m. Let S_f be the labeled sample consisting of all instances in $\mathcal{X}$ labeled according to f. Observe that $\mathscr{G}(\mathscr{F}(S_f))$ must be logically equivalent to f. Thus, a computationally unbounded teacher could produce the examples in $\mathscr{F}(S_f)$ as a teaching set. We now give sufficient conditions under which the learner can simply use $\mathscr{G}$ to obtain a hypothesis logically equivalent to f.

THEOREM 9. *If there is a valid data compression scheme of size k for representation class $\mathscr{C}$ and $\mathscr{F}(S_f)$ produces an example set for which f is the minimum consistent hypothesis (for any predefined ordering of $\mathscr{C}$), then $\mathscr{C}$ is T/L-teachable with at most k examples.*

Proof. The teacher will use $\mathscr{F}(S_f)$ as the teaching set. The learner will output the minimum consistent hypothesis consistent with the teaching set. Since, by the conditions of the theorem, the target f is the minimum hypothesis consistent with $\mathscr{F}(S_f)$, the hypothesis output by the learner cannot be affected by the additional examples added to the teaching set by the adversary. ▌

Many of the space-bounded learning algorithms that Floyd [8] presents satisfy the conditions of Theorem 9 and thus we immediately obtain results for our teaching model. To state these results, we must define the Vapnik–Chervonenkis dimension [29]. Let $\mathcal{X}$ be any instance space, and let $\mathscr{C}$ be a concept class over $\mathcal{X}$. A finite set $Y \subseteq \mathcal{X}$ is *shattered* by $\mathscr{C}$ if $\{c \cap Y \mid c \in \mathscr{C}\} = 2^Y$. In other words, $Y \subseteq \mathcal{X}$ is shattered by $\mathscr{C}$ if for each subset $Y' \subseteq Y$, there is a concept $c \in \mathscr{C}$ which contains all of Y', but none of the instances in $Y - Y'$. The *Vapnik–Chervonenkis dimension* of $\mathscr{C}$, denoted $\text{VCD}(\mathscr{C})$, is defined to be the smallest d for which no set of $d+1$ points is shattered by $\mathscr{C}$. Blumer *et al.* [6] have shown that this combinatorial measure of a concept class characterizes the number of examples required for learning any concept in the class under the distribution-free or PAC model of Valiant [28].

Related to the VC-dimension are the notions of *maximal* and *maximum* concept classes [8, 30]. A concept class is *maximal* if adding any concept to the class increases the VC dimension of the class. Define

$$\Phi_d(m) = \begin{cases} \sum_{i=0}^{d} \binom{m}{i} & \text{for} \quad m \geqslant d \\ 2^m & \text{for} \quad m < d. \end{cases}$$

If $\mathscr{C}$ is a concept class of VC-dimension d on a finite set $\mathscr{X}$ with $|\mathscr{X}| = m$, then the cardinality of $\mathscr{C}$ is at most $\Phi_d(m)$ [26, 29]. A concept class $\mathscr{C}$ over $\mathscr{X}$ is *maximum* if for every finite subset $Y \subseteq \mathscr{X}$, the class $\mathscr{C}$, when restricted to be a class over Y, contains $\Phi_d(|Y|)$ concepts.

Floyd [8] shows that if $\mathscr{C}$ is a maximum class of VC-dimension d on the set $\mathscr{X}$, then there is a data compression scheme of size d for $\mathscr{C}$. Furthermore, it is easily shown that for this compression scheme, $\mathscr{F}(S_f)$ produces an example set for which f is the only consistent hypothesis. Thus from Floyd's results and Theorem 9, we get the following corollary.

COROLLARY 10. *For any maximum class $\mathscr{C}$, there is a valid T/L pair such that the optimal teaching set has length at most the VC-dimension of $\mathscr{C}$.*

We can show the corresponding result for intersection-closed classes by applying results from Helmbold, Sloan, and Warmuth [15]. They define a spanning set of a representation $c \in \mathscr{C}$ with respect to the class $\mathscr{C}$ to be a set $I \subseteq c$ having the property that c is the unique, most specific representation consistent with the instances in I and they show that for intersection-closed classes all minimum spanning sets have size at most $\text{VCD}(\mathscr{C})$. They then give an algorithm to learn the nested-difference of a set of size p of functions from $\mathscr{C}$ while saving at most $p \cdot \text{VCD}(\mathscr{C})$ examples, thus giving the following result.

COROLLARY 11. *For any intersection-closed representation class $\mathscr{C}$ for which there is an efficient query algorithm, the nested difference of p functions from $\mathscr{C}$ is T/L-teachable using at most $p \cdot \text{VCD}(\mathscr{C})$ examples with a polynomial-time learner.*

7. ALTERNATE MODELS

In this section we briefly describe some variations of our model that we feel are worthy of study. The model we have presented here places minimal restrictions on the T/L pair while ensuring that there is no collusion. For some applications, one may want to limit the assumptions the teacher may make about the learner without going to the extreme of only allowing the teacher to assume that the learner is consistent. For example, one class of learners that would be interesting to study is that of learners that always select a minimum (in terms of the number of instances classified as positive) consistent hypothesis. In fact, this type of learner was studied by Natarajan [21] in terms of one-sided learning. As another example, we could consider the class of learners that only select a minimal consistent hypothesis. This corresponds to requiring that the learner always selects an element from the set of most specific concepts in Mitchell's version space [20].

Another interesting variation is one in which the learner is not required to exactly identify the target, but rather needs only output an ε-good approximation to the target. (Romanik and Smith [23, 24] consider a PAC-style criterion in their work.)

Finally, another interesting model is one in which there are two distinct stages. The first stage operates as in our current model except that the teacher is not required to provide examples sufficient for exact identification. In the second stage the teacher lists all instances that are exceptions to the rule taught during the first stage. During the second stage the learner just appends to its hypothesis the list of "exceptions" given by the teacher. The motivation behind this model is that it is often easiest to teach by first oversimplifying the truth and then making the needed corrections. For example, in teaching a child about spelling, you could first teach the rule that the letter "i" always comes before "e." Then, you can add the needed exceptions. It is also possible that the list of exceptions could take the form of another rule. For example, "i" comes before "e" except after "c".

8. CONCLUDING REMARKS

We now briefly discuss the two directions of future research that we find most interesting. First of all, an interesting open question is to determine whether there exist teacher/learner pairs in which the learner requires only polynomial computation time for such classes as 2-quasi Horn sentences and read-thrice DNF that appear difficult to learn from queries [2, 1]. (It is easily shown for these problems that if the teacher is given an arbitrary representation of the target then the teacher must perform a satisfiability problem to even decide if there are any positive examples and thus could not run in polynomial time assuming $P \neq NP$.)

Another interesting research direction opened up by this new model is the following. As we did for 1-decision lists and Horn sentences, one can take known learning algorithms and design teachers that enable the sample and time complexity of the algorithm to be asymptotically reduced. However, the algorithms that have been designed to work against an adversarial environment are most likely not going to be the best algorithms when we allow the algorithm to be paired with a teacher. In other words, for some classes we expect that there should be teacher/learner pairs that work better than any known algorithm when paired with the best possible teacher, yet the algorithm used by the learner may be very poor against an adversarial environment. Such a study could lead to general techniques for designing teacher/learner pairs.

ACKNOWLEDGMENTS

We are very grateful to Dana Angluin for many valuable discussions on this topic. In particular, Theorem 1 was suggested by her and the extension of Theorem 2 to include all example-based queries (along with the definition of example-based queries) was due to her. In addition, the model of teaching

with patching was developed in collaboration with her. We thank Rob Schapire for suggesting that we explore the relationship between teaching and data compression. The observation of T/L cooperation for classes of size at most $2^{O(n)}$ is due to Avrim Blum and we thank him for that. We thank Lenny Pitt for his help in formalizing the notion of collusion and for describing to us Nick Littlestone's algorithm for exactly identifying decision lists. Finally, we thank an anonymous referee for many extremely useful comments and suggestions.

REFERENCES

1. H. Aizenstein, L. Hellerstein, and L. Pitt, Read-thrice DNF is hard to learn with membership and equivalence queries, *in* "33rd Annual Symposium on Foundations of Computer Science, October 1992," pp. 523–532.
2. D. Angluin, M. Frazier, and L. Pitt, Learning conjunctions of Horn clauses, *Mach. Learning* **9** (1992), 147–164.
3. D. Angluin, Queries and concept learning, *Mach. Learning* **2**, No. 4 (1988), 319–342.
4. M. Anthony, G. Brightwell, D. Cohen, and J. Shawe-Taylor, On exact specification by examples, *in* "Proceedings Fifth Annual Workshop on Computational Learning Theory, July 1992," pp. 311–318, Assoc. Comput. Mach., New York, 1992.
5. J. Barzdin and R. Freivalds, On the prediction of general recursive functions, *Soviet Math. Dokl.* **13** (1972), 1224–1228.
6. A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth, Learnability and the Vapnik–Chervonenkis dimension, *J. Assoc. Comput. Mach.* **36**, No. 4 (1989) 929–965.
7. N. Bshouty, Exact learning via the monotone theory, *in* "34th Annual Symposium on Foundations of Computer Science, November 1993."
8. S. Floyd, "On Space-bounded Learning and the Vapnik–Chervonenkis Dimension," Ph.D. thesis, International Computer Science Institute, December 1989.
9. M. Frazier, "Matters Horn and Other Features in the Computational Learning Theory Landscape: The Notion of Membership," Ph.D. thesis, University of Illinois, Urbana-Champaign, April 1994.
10. R. Freivalds, E. Kinber, and R. Wiehagen, Inference from good examples, *Theoret. Comput. Sci.* **110** (1993), 131–144.
11. S. A. Goldman and M. J. Kearns, On the complexity of teaching, *in* "Proceedings 4th Annual Workshop on Comput. Learning Theory," pp. 303–314, Morgan Kaufmann, San Mateo, CA, 1991; *J. Comput. System Sci.* **50** (1995), 20–31.
12. S. A. Goldman, R. L. Rivest, and R. E. Schapire, Learning binary relations and total orders, *SIAM J. Comput.* **22**, No. 5 (1993), 1006–1034.
13. S. A. Goldman and R. H. Sloan, "The Power of Self-directed Learning," Technical Report WUCS-92-49, Washington University, Department of Computer Science, November 1992; *Mach. Learning*, to appear.
14. S. Goldwasser, S. Goldwasser, and C. Rackoff, The knowledge complexity of interactive proofs, "26th Annual Symposium on Foundations of Computer Science, October 1985," pp. 291–304.
15. D. Helmbold, R. Sloan, and M. K. Warmuth, Learning nested differences of intersection-closed concept classes, *Mach. Learning* **5** (1990), 165–196, (special issue for *COLT 89*.)
16. J. Jackson and A. Tomkins, A computational model of teaching, *in* "Proceedings Fifth Annual Workshop on Computational Learning Theory," pp. 319–326, ACM Press, New York, July 1992.
17. S. Lange and R. Wiehagen, Polynomial-time inference of arbitrary pattern languages, *New Generation Comput.* **8** (1991), 361–370.
18. N. Littlestone, Learning when irrelevant attributes abound: A new linear-threshold algorithm, *Mach. Learning* **2** (1988), 285–318.
19. W. Maass and G. Turán, Lower bound methods and separation results for on-line learning models, *Mach. Learning* **9** (1992), 107–145.
20. T. M. Mitchell, Generalization as search, *Artif. Intell.* **18**, No. 2 (1982), 203–226.
21. B. K. Natarajan, On learning Boolean functions, *in* "Proceedings Nineteenth Annual ACM Symposium on Theory of Computing, May 1987," pp. 296–304.
22. R. L. Rivest, Learning decision lists, *Mach. Learning* **2**, No. 3 (1987), 229–246.
23. K. Romanik, Approximate testing and learnability, *in* "Proceedings Fifth Annual Workshop on Computational Learning Theory," pp. 327–332, ACM Press, New York, July 1992.
24. K. Romanik and C. Smith, "Testing Geometric Objects," Technical Report UMIACS-TR-90-69, University of Maryland College Park, Department of Computer Science, 1990.
25. S. Salzberg, A. Delcher, D. Heath, and S. Kasif, Learning with a helpful teacher, *in* "12th International Joint Conference on Artificial Intelligence, August 1991," pp. 705–711.
26. N. Sauer, On the density of families of sets, *J. Combin. Theory Ser. A* **13** (1972), 145–147.
27. A. Shinohara and S. Miyano, Teachability in computational learning, *New Generation Comput.* **8** (1991), 337–347.
28. L. Valiant, A theory of the learnable, *Commun. ACM* **27**, No. 11 (1984), 1134–1142.
29. V. N. Vapnik and A. Ya. Chervonenkis, On the uniform convergence of relative frequencies of events to their probabilities, *Theory Probab. Appl.* **16** (1971), 264–280.
30. E. Welzl, Complete range spaces, unpublished manuscript, 1987.