Report Number: WUCS-TM-94-1

1994-01-01

Learning and Teaching of Boolean and Geometric Classes

H. David Mathias

We consider the concept classes of DNF formulas and unions of discretized, axis-parallel d-dimensional boxes in discretized d-dimensional space with respect to several different learning models. In the model of learning with queries we present an algorithm to learn unions of boxes. We introduce a model of teaching that prevents illicit communication between the teacher and the leaner but that captures the intuitive aspect of teaching: a learner should perform at least as well with a cooperative teacher as with an adversarial teacher. We propose the study of teaching of DNF formulas and unions of boxes in this model.... **Read complete abstract on page 2.**

Learning and Teaching of Boolean and Geometric Classes

H. David Mathias

Complete Abstract:

We consider the concept classes of DNF formulas and unions of discretized, axis-parallel d-dimensional boxes in discretized d-dimensional space with respect to several different learning models. In the model of learning with queries we present an algorithm to learn unions of boxes. We introduce a model of teaching that prevents illicit communication between the teacher and the leaner but that captures the intuitive aspect of teaching: a learner should perform at least as well with a cooperative teacher as with an adversarial teacher. We propose the study of teaching of DNF formulas and unions of boxes in this model. We also propose a study of a teaching model in which the learner may be randomized. It is currently unclear if randomization of the learner allows teaching of classes that are unteachable in the current teaching model.

Learning and Teaching of Boolean and Geometric Classes

H. David Mathias

WUCS-TM-94-01

April 9, 1994

Department of Computer Science
Campus Box 1045
Washington University
One Brookings Drive
St. Louis, MO 63130-4899

Abstract

We consider the concept classes of DNF formulas and unions of discretized, axis-parallel d-dimensional boxes in discretized d-dimensional space with respect to several different learning models. In the model of learning with queries we present an algorithm to learn unions of boxes. We introduce a model of teaching that prevents illicit communication between the teacher and the learner but that captures the intuitive aspect of teaching: a learner should perform at least as well with a cooperative teacher as with an adversarial teacher. We propose the study of teaching of DNF formulas and unions of boxes in this model. We also propose a study of a teaching model in which the learner may be randomized. It is currently unclear if randomization of the learner allows teaching of classes that are unteachable in the current teaching model.

1. Introduction

Computational learning theory is, roughly speaking, the formal study of methods by which machines can learn, with emphasis on efficient use of time and data. While much valuable research in machine learning has been done with only *ad hoc* attention given to computational efficiency, we feel that it is essential to the goal of truly understanding machine learning that a sound theoretical framework is present.

One of the great difficulties confronted in learning theory research at this early stage is finding appropriate models for learning problems that enable careful analysis of data and time requirements but also adequately express the underlying problem. How should the learner acquire its information? What constitutes successful learning? How much interaction with the environment is allowed? What is the nature of the environment? These questions have been answered in various ways and the answers have provided several models. The last question, in particular, may have a very large impact on what can and cannot be learned within a model. The research proposed in this paper examines this question in some detail.

Much of the research thus far in computational learning theory has centered on learning (subclasses of) Boolean formulas (though not exclusively, as we shall see). In large part, this is due to the expressive power of these formulas for representing information. The fact that computers (and computer scientists) work well with such things is also important. In this proposal we will consider disjunctive normal form (DNF) formulas. In addition we will examine the geometric class of unions of boxes (these classes will be defined later).

The models of learning that we will use consist of one existing model (learning with queries) and a new model that will be defined (a model of teaching). In the query model the learner asks questions which are answered by an adversary who tries to reveal as little information as possible with each answer. Thus, the learner is interacting with a worst-case environment. In the teaching model we define, the adversary is replaced by a helpful teacher who tries to speed the learner's learning process. Intuitively, the helpful teacher should allow more efficient learning. However, due to technicalities that will be discussed later, this was not always the case in previous models of teaching.

This paper is structured as follows. Section 2 defines the models we use and provides the notation that will be used throughout the proposal. Section 3 presents the teaching model of Goldman and Mathias and discusses some of the important related results. Section 4 presents an algorithm for learning the unions of discrete d-dimensional boxes in discrete d-dimensional space and proposes two related teaching problems. Section 6 discusses the open problem of the learnability of DNF formulas and proposes developing algorithms for this class, and subclasses, in the model of teaching discussed in Section 3. Finally, Section 7 proposes development of a model for randomized teaching and suggests a relationship to previous work that may help in teaching DNF formulas.

2. General Definitions

We now describe the model of concept learning with queries as developed by Angluin [4]. Under this model the learner's goal is to learn *exactly* how an unknown, Boolean-valued target function f, drawn from *concept class* $\mathcal{C}$, classifies, as positive or negative, all instances from *instance space* $\mathcal{X}$. It is often convenient to consider a concept as the set of instances that it classifies as positive. In set notation, for each concept $f \in \mathcal{C}$, $f \subseteq \mathcal{X}$. Often we define $\mathcal{C} = \bigcup_{n \geq 1} \mathcal{C}_n$. That is, $\mathcal{C}$ is parameterized by a dimension measure appropriate to the domain. (In the case of DNF formulas n is the number of Boolean variables). Similarly, $\mathcal{X} = \bigcup_{n \geq 1} \mathcal{X}_n$ is the set of instances to be classified for a problem of size n. We say that instance x is a *positive instance* for target concept f if $x \in f$ and say that x is a *negative instance* otherwise. For instance $x \in \mathcal{X}_n$, let $f(x) = 1$ denote that x is a positive instance for f, and and let $f(x) = 0$ denote that x is a negative instance for f. A *hypothesis* h is a polynomial-time algorithm that given any $x \in \mathcal{X}_n$ outputs a prediction for $f(x)$.

As mentioned above, the learning criterion expected here is that of *exact identification*. This is achieved by the learner if it can infer a hypothesis h that is logically equivalent to the target concept. That is, in order to achieve exact identification we require the hypothesis h to be such that $h(x) = f(x)$ for all instances $x \in \mathcal{X}$. In this model, the learner is provided queries with which to learn about f. The two types of queries in most common use are membership queries and equivalence queries. A *membership query* $\mathrm{MQ}(x)$ returns "yes" if $f(x) = 1$ and returns "no" if $f(x) = 0$. An *equivalence query*, $\mathrm{EQ}(h)$, takes a polynomially evaluatable hypothesis h and returns "yes" if h is logically equivalent to f or returns a counterexample otherwise. A *positive counterexample* x is one for which $f(x) = 1$ but $h(x) = 0$. Likewise, a *negative counterexample* is one for which $f(x) = 0$ but $h(x) = 1$. Two other query types that will be used here are superset and subset queries. A *superset query*, $\mathrm{SupQ}(h)$, takes a hypothesis, h, and returns "yes" if $h \supseteq f$ or a counterexample otherwise (in this case the counterexample must be an instance x such that $h(x) = 0$ and $f(x) = 1$). Similarly, a *subset query*, $\mathrm{SubQ}(h)$, takes hypothesis h and returns "yes" if $h \subseteq f$ or a counterexample otherwise (in this case the counterexample must be an instance x such that $h(x) = 1$ and $f(x) = 0$). Each of the queries that returns a counterexample has a *restricted* counterpart that simply answers "no" rather than returning a counterexample.

Another important learning model is the PAC model introduced by Valiant [42]. In this model the learner is presented with labeled examples chosen at random according to an unknown, arbitrary distribution $\mathcal{D}$ over the instance space. The learner's goal is to output a hypothesis that with high probability, at least $(1 - \delta)$, correctly classifies most of the instances in the instance space (the weight, under $\mathcal{D}$, of misclassified instances is at most ϵ). The learner is permitted time polynomial in $1/\epsilon$, $1/\delta$ and n (recall that n is some size measure relevant to the doamin) to formulate a hypothesis.

Boolean formulas in disjunctive normal form (DNF) are an important concept class discussed throughout this proposal. In the Boolean domain we have $\mathcal{X}_n = \{0,1\}^n$ where $y_1, y_2, \ldots, y_n$ are the Boolean variables. A *literal*, ℓ_i, is a variable or its negation. A DNF formula $f = t_1 + t_2 + \ldots + t_m$ is a disjunction of terms where each term $t_i = \ell_1 \ell_2 \cdots \ell_r$ is a conjunction of some number of literals. A k-term DNF formula is simply a DNF with k terms where k is usually some constant. Finally, a *monotone* DNF formula is a DNF

formula in which all of the variables are unnegated. An *implicant* of a DNF formula f is a term t such that if $t(x) = 1$ then $f(x) = 1$. A *prime implicant* of f is a term T that is an implcant with the property that if any literal is deleted from T then it is no longer an implicant of f.

3. A Teaching Model

All of the results in this section are presented in more detail by Goldman and Mathias [20].

Recently, there has been interest in developing formal models of *teaching* [7, 18, 22, 28, 41] 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 one logically equivalent to it) is consistent. Thus, teaching under these models is made unnecessarily difficult since the problem reduces to teaching an obstinate 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 Tompkins [28] 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 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 $|C| - 1$ 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 [28] and teaching linearly separable Boolean functions [7]. Thus these hardness results appear to be due to a defect in the model rather than an intrinsic difficulty in teaching these classes. Similar problems 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

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

[2] A self-directed learner [21, 24] 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.

and learner could cooperate (which should be the case if the teacher is working with the single learner), intuitively this phenomenon should not occur.

The key contribution of this work is the introduction of a formal teaching model that allows the teacher and learner to cooperate, yet the teacher cannot simply encode the target function. We start with a teacher/learner pair as in the model introduced by Jackson and Tompkins [28]. 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 his 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.

3.1. Previous Work

We now briefly review the theoretical work studying the complexity of teaching. Goldman, Rivest and Schapire [23] introduced the model of teacher-directed learning, 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 [22] 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 [41] 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. Romanik and Smith [39, 40] 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 learning theory community, our new teaching model is most closely related to the model introduced by Jackson and Tomkins [28]. 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 [25] 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 (or some logically equivalent 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 size complexity of the target function or a stopping condition.

Within the inductive inference paradigm, Freivalds, Kinber and Wiehagen [18] and Lange and Wiehagen [29] 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. Lange and Wiehagen [29] examine learning pattern languages and show that this can be achieved with good examples.

3.2. 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* $\mathcal{C}$, which is a set of representations of functions mapping some domain $\mathcal{X}$ into $\{0,1\}$. Each representation $f \in \mathcal{C}_n$ has a *size* denoted by $|f|$. Typically, this is the number of symbols needed to write the representation of f as a member of the representation class $\mathcal{C}_n$ from which it is chosen.

A *teaching set* for $f \in \mathcal{C}$ is an unordered set of labeled instances where each instance is selected from $\mathcal{X}$ and labeled according to f. We define the teacher T to be an algorithm that when given a representation $f \in \mathcal{C}$ 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 $\mathcal{C}$. 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 $\mathcal{C}' \supseteq \mathcal{C}$.) If the learner is deterministic then $L(S)$ is well-defined, however, in the case that the learner uses a randomized algorithm, instead $L(S)$ induces a probability distribution over $\mathcal{C}$. We shall denote this distribution by $P_L(S)$. For the remainder of this section we consider only deterministic learners.

We now describe our teaching protocol. The learner L and teacher T both have prior knowledge of the representation class $\mathcal{C}$ 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 Figure 1, proceeds as follows:

- The adversary selects a target function $f \in \mathcal{C}$ and gives f to T.
- The teacher computes $T(f)$ and gives it to A.
- Next the adversary (with knowledge of $\mathcal{C}, f, T$ 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 $L(S_A)$.

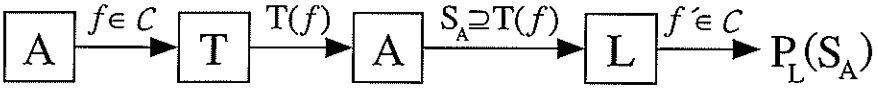

Figure 1: An overview of a teaching session. The adversary chooses the target function, f, giving it to the teacher. The teacher then generates $T(f)$. Given $T(f)$ the adversary generates $S_A \supseteq T(f)$ and gives this to the learner. The learner (possibly randomized) outputs a representation from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

The goal of the teacher is to teach the learner to predict perfectly 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 the representation of the target via the instances selected without regard for the labels) which could trivialize our model. Informally, we define collusion as the passing of information, by the teacher to the learner, about the representation of the target rather than about the function represented.

We define a *valid T/L pair* for $\mathcal{C}$ to consist of a teacher T and learner L such that: For any $f \in \mathcal{C}$ 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 $f' \in \mathcal{C}$, if f' has non-zero 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 $f \in \mathcal{C}_n$, 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 $\mathcal{C}$ is *T/L-teachable* if, for all $f \in \mathcal{C}_n$, there exists a valid T/L pair for which $|T(f)|$ is polynomial in $|f|$ and n. We say that $\mathcal{C}$ 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 $\mathcal{C}$ 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.

3.3. Justification for Our Model

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

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

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	$(h(x_1) = f(x_1)) \vee (h(x_1) \neq *)$

Figure 2: We show how to represent various queries as example-based queries. Equivalence, membership, subset, superset, disjointness and exhaustiveness queries are defined by Angluin [4]. 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 [35]) the learner can present a hypothesis $h : \mathcal{X} \to \{0, 1, *\}$, and is either told that all specified instances are correct or is given an $x \in \mathcal{X}$ such that $h(x) \in \{0, 1\}$ and x is misclassified by h.

computationally unbounded teacher and a polynomial-time learner. We define an *example-based query* to be any query of the form:

$$\forall\ (x_1, x_2, \ldots, x_k) \in \mathcal{X}^k\ \text{, does } \varphi_f(x_1, x_2, \ldots, x_k) = 1?$$

where k is constant, and $\varphi_f(x_1, x_2, \ldots, x_k)$ 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 $(x_1, x_2, \ldots, x_k) \in \mathcal{X}^k$ (with their labels) for which $\varphi_f(x_1, x_2, \ldots, x_k) = 0$ and the examples (and their labels) for which membership queries were made to evaluate the predicate. In Figure 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.

THEOREM 3.1. *Any representation class 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 $\mathcal{C}$ learnable in deterministic polynomial-time by an algorithm $\mathcal{A}$ that uses only example-based queries. We assume there is some total ordering π on $\mathcal{X}$ (such as a lexicographical order) upon which the learner and teacher have agreed. Given any two sets of k instances from $\mathcal{X}$ we define the following ordering among them. Let $x = (x_1, \ldots, x_k)$ for $x_1 < x_2 < \cdots < x_k$ be one set and let $y = (y_1, \ldots, y_k)$ 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 $1 \leq j \leq k$ 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 $T(f) = \emptyset$. Now T simulates $\mathcal{A}$'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 $(x_1, x_2, \ldots, x_k) \in \mathcal{X}^k$ from smallest to largest, evaluating $\varphi(x_1, \ldots, x_k)$. If there are no counterexamples to q then the teacher replies "yes" to $\mathcal{A}$'s query. Otherwise, let $(x_1, x_2, \ldots, x_k)$ be the smallest counterexample and let $q_1, \ldots, q_\ell$ be the instances on which membership queries were made to evaluate $\varphi(x_1, \ldots, x_k)$. 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 $\mathcal{A}$ with S, and updates $T(f)$ to be $T(f) \cup S$. The teacher continues in this manner until $\mathcal{A}$ halts.

We now create the learner L from $\mathcal{A}$ as follows. Let $S_A \supseteq T(f)$ be the teaching set that the learner receives. Whenever $\mathcal{A}$ makes a query q, the leaner will proceed as follows. The learner will consider all k-tuples in S_A from smallest to largest. For each such tuple $(x_1, \ldots, x_k)$ the learner attempts to evaluate $\varphi(x_1, \ldots, x_k)$. 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 $\varphi(x_1, \ldots, x_k)$. If $\varphi(x_1, \ldots, x_k) = 0$, then L gives $\mathcal{A}$ the k-tuple $(x_1, \ldots, x_k)$ 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 $(x_1, \ldots, x_k)$ must not be a counterexample. Thus, if the learner computes that $\varphi(x_1, \ldots, x_k) = 1$ 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 $\mathcal{A}$ 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 $\mathcal{A}$ always remain the same. Since $\mathcal{A}$ 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 $\mathcal{A}$ exactly the same counterexamples. ■

The following corollary follows directly from Theorem 3.1.

COROLLARY 3.1. *If representation class 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 Tompkins [28] that the class of 1-decision lists is not teachable without trusted information, and the negative result of Anthony et. al [7] that linearly separable Boolean functions are not efficiently teachable. In fact, Bshouty's [12] 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.

3.4. Research Plan

There is further work to be done in this new model of teaching. We propose the study of teaching of specific classes in this model. While it may be possible to teach some class in this model that connot be learned in any of the previously existing models, it is also be interesting to examine how T/L pairs compare to standard learning algorithms for classes known to be larrnable. In particular, how much more efficient in the use of time and data can T/L pairs be than learning algorithms in the standard learning models?

Two classes are of particular interest in this regard. Decision lists are of interest because they have a rich representational power. As defined by Rivest [38], a 1-decision list (1-DL) over the set $Y_n = \{y_1, y_2, \ldots, y_n\}$ of n Boolean variables is an ordered list $f = \langle(\ell_1, b_1), \ldots, (\ell_r, b_r)\rangle$ where each ℓ_i is y_i or $\overline{y_i}$ for $y_i \in Y_n$ and each $b_i \in \{0,1\}$. For an instance $x \in \{0,1\}^n$, we define $f(x) = b_j$ where $1 \leq j \leq r$ is the least value such that ℓ_j is 1 in x; $f(x) = \overline{b}_r$ if there is no such j. We refer to each pair (ℓ_i, b_i) as a node in f. One may think of a decision list as an extended "**if—then—elseif—... else**" rule. 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 Y_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.

Another class of interest is Horn sentences. The question of the learnability of Horn sentences was open for some time before being answered positively by Angluin, Frazier and Pitt [5]. A Horn clause is a disjunction of literals at most one of which is unnegated. A Horn sentence is a conjunction of Horn clauses. The algorithm by Angluin, Frazier and Pitt [5] exactly identifies an m-clause Horn sentence using $O(mn)$ equivalence queries, $O(m^2n)$ membership queries and, $\tilde{O}(m^2n^2)$ [5], where n is the number of variables in the instance

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

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

space. Note that each Horn clause can be viewed as a logical implication in which the consequent contains the, at most one, unnegated variable.

We propose the study of teaching the classes of 1-DL and Horn sentences in this new model of teaching. The study of the teachability of these classes may lead to the study of other classes that are known to be learnable. The question of the teachability of a class not known to be learnable in existing models is addressed in Section 6.

4. Learning Unions of Boxes

Recently, learning geometric concepts in d-dimensional Euclidean space has been the subject of much research. One such class of geometric concepts is unions of boxes. (By a "box", we mean an axis-aligned hypercuboid. So a box is the set of all points whose cartesian coordinates satisfy a given set of univariate linear inequalities.) We study this problem under the model of learning with queries [4] in which the learner is required to output a final hypothesis that correctly classifies *every* point in the domain as to whether or not it is inside of one of the target boxes. Note that this class is easily learnable in the PAC model. The key to that algorithm is the use of an approxiamte set cover approach to cover all of the positive examples seen. To apply the query model to a domain such as learning boxes (or unions of boxes) in d-dimensional Euclidean space, it is necessary to look at a discretized version of the domain. More formally, let BOX_n^d denote the class of axis-parallel boxes over $\{1,\ldots,n\}^d$. (So d represents the number of dimensions and n represents the number of discrete values that exist in each dimension.) Let $[i,j]$ denote the set $\{m \in \mathrm{N} |\ i \leq m \leq j\}$. Then, $\text{BOX}_n^d = \{\times_{k=1}^d [i_k, j_k] \mid 1 \leq i_k \leq j_k \leq n\}$. So i_k and j_k are the minimum and maximum positive values of the k-th coordinate of a box. Note that by allowing equality of i_k and j_k we include in BOX_n^d boxes with zero size in dimension k. Finally, let $\bigcup_s \text{BOX}_n^d$ denote the class of the union of at most s concepts from BOX_n^d. We note that it is easy to show that this class is a generalization of disjunctive normal form (DNF) formulas[6] and a special case of the class of unions of intersections of half-spaces over $\{1,\ldots,n\}^d$.

We present an algorithm that uses membership and equivalence queries to exactly learn the concepts given by the union of s axis-parallel boxes over $\{1,\ldots,n\}^d$. This algorithm receives at most sd counterexamples, makes $O((8s)^d + sd\log n)$ membership queries, and uses $O((8s)^d + sd\log n)$ time. Thus our algorithm is the first algorithm to exactly learn the union of s discretized boxes in d-dimensional discretized Euclidean space in polynomial time in s and $\log n$ for any consant d.

The hypothesis class used by this algorithm, selected to keep the algorithm simple, can be evaluated in time $O(d\log s)$[7]. However, in $O((2s)^{2d})$ time we can transform our

[6] For the class of DNF formulas let $n = 2$, d be the number of boolean variables in the instance space and s be the number of terms in the formula. (Thus, dimensions correspond to variables and boxes correspond to terms). Note that unless the target DNF is a tautology the maximum dimension of any term is $d-1$.

[7] The hypothesis essentially partitions $\{1,\ldots,n\}^d$ into at most $(4s+1)^d$ regions where all points in any region are classified as either positive or negative. The classifiations for the regions are stored in a bit matrix, and we use a set of d balanced binary search trees to efficiently find the region in which a point is contained. (See Section 4 for more details.)

hypothesis to the union of at most $O(ds \log s)$ boxes from BOX_n^d. Thus we obtain the even stronger result that our algorithm learns the union of s axis-parallel boxes over $\{1, \ldots, n\}^d$, makes at most $sd+1$ equivalence queries[8] where each equivalence query is simply the union of $O(ds \log s)$ concepts from BOX_n^d, makes $O((8s)^d + sd \log n)$ membership queries, and has time complexity $O((2s)^{2d} + sd \log n)$. Thus for any constant d, this algorithm still uses time and queries polynomial in s and $\log n$.

4.1. Previous Work

The problem of learning geometric concepts over a discrete domain was extensively studied by Maass and Turan [33, 34, 35]. One of the geometric concepts that they studied was the class BOX_n^d. They showed that if the learner was restricted to only make equivalence queries in which each hypothesis was drawn from BOX_n^d then $\Omega(d \log n)$ queries are needed to achieve exact identification [30, 35]. Auer [8] improves this lower bound to $\Omega(\frac{d^2}{\log d} \log n)$.

If one always makes an equivalence query using the simple hypothesis that produces the smallest box consistent with the previously seen examples, this yields an algorithm that makes $O(dn)$ equivalence queries. An algorithm making $O(2^d \log n)$ equivalence queries was given by Maass and Turan [32, 34]. The best known result for learning the class BOX_n^d was provided by the work of Chen and Maass [15] in which they gave an algorithm making $O(d^2 \log n)$ equivalence queries. They also provide an algorithm to learn the union of two axis-parallel rectangles in the discretized space $\{1, \ldots, n\} \times \{1, \ldots, m\}$ in time polynomial in $\log n$ and $\log m$, where one rectangle has a corner in the top left corner of the instance space and the other has a corner in the bottom right corner of the instance space. Finally, Auer [8] investigates exact learning of rectangles where some of the counterexamples, given in response to equivalence queries, are noisy. Auer shows that BOX_n^d is learnable if and only if the fraction of noisy examples is less than $1/(d+1)$ and presents an efficient algorithm that handles a noise rate of $1/(2d+1)$. More recently, Chen [14] gave an algorithm that used equivalence queries to learn general unions of two boxes in the (discretized) plane. The algorithm uses $O(\log^2 n)$ equivalence queries, and involves a detailed case analysis of the shapes formed by the two rectangles. It does not appear to generalize easily to higher numbers of boxes or dimensions.

Since designing algorithms to exactly learn the union of boxes from equivalence queries has been difficult, a natural problem to study is the problem of exactly learning unions of boxes from *both* membership and equivalence queries. In this direction, in work independent of ours, Chen and Homer [27] have given an algorithm to learn the union of s rectangles in the plane using $O(s^3 \log n)$ queries (both membership and equivalence) and $O(s^5 \log n)$ time. The hypothesis class of their algorithm is the union of $8k^2 - 2$ rectangles.

Closely related to the problem of learning the union of discretized boxes, is the problem of learning the union of non-discretized boxes in the PAC model [42]. There are known PAC-algorithms to efficiently learn an s-fold *union* of boxes in E^d for s constant and arbitrary d (the dominating term contains the factor d^s) [31] and for d constant and arbitrary s (the dominating term contains the factor s^d) [11]. Note that a polynomial-time algorithm for

[8]The final equivalence query is the correct hypothesis, and thus at most sd counterexamples are received.

arbitrary d and s would solve the problem of learning DNF, in view of the earlier observation that unions of boxes are a generalization of DNF. Under a variation of the PAC model in which membership queries can be made, Frazier et al. [17] have given an algorithm to PAC-learn the s-fold union of boxes in E^d for which each box is entirely contained within the positive quadrant *and* contains the origin. Furthermore, their algorithm learns this subclass of general unions of boxes in time polynomial in both s and d.

Because the teaching model described in Section 3.2 is very new, we know of no previous attempt to design efficient algorithms in this model for the class $\bigcup_s \text{BOX}_n^d$. However, Goldman and Kearns [22] show that $\text{TD}(\text{BOX}_n^d) = 2 + 2d$ (the teaching dimension model is described in Section 3.1).

4.2. Definitions

We use $y_1, \ldots, y_d$ to denote the d dimension variables. An alternate view of a concept $f \in \bigcup_s \text{BOX}_n^d$ that we use is to treat it as an axis-parallel d-dimensional discretized polygon. For the remainder of this paper, when using the word "polygon" we refer to such an axis-parallel discretized polygon. Observe that each box is defined by the intersection of $2d$ halfspaces, two for each dimension. We introduce the following definitions:

DEFINITION 1. *We define a* hyperplane *to be the set of all points for which one of the y_i has some fixed value.*

(Thus a hyperplane is metrically equivalent to $\{1, \ldots, n\}^{d-1}$.)

DEFINITION 2. *For a concept $f \in \bigcup_s \text{BOX}_n^d$, we define a dimension i, $+/-$ pair to be a positive point $p_+ = (x_1, \ldots, x_i, \ldots, x_d)$, where $y_j = x_j$ for $1 \leq x_j \leq n$, paired with a negative point p_- where $p_- = (x_1, \ldots, x_i + 1, \ldots x_d)$ or $p_- = (x_1, \ldots, x_i - 1, \ldots, x_d)$ where we implicitly assume that any point that is outside of $\{1, \ldots, n\}^d$ is a negative point. We use $+/^i-$ pair to denote a dimension i, $+/-$ pair.*

It can be seen that any $+/-$ pair defines a unique (axis-aligned) halfspace separating the positive from the negative point. We also define the hyperplane associated with a given $+/-$ pair to be the (unique) hyperplane H that contains the positive but not the negative point. Namely, for a $+/^i-$ pair where the positive point's ith coordinate is c, if the negative point's ith coordinate is $c + 1$ then H is given by $y_i \leq c$. Similarly, if the negative point's ith coordinate is $c - 1$ then H is given by $y_i \geq c$. Observe that there are a total of at most $2sd$ hyperplanes that define the target polygon.

OBSERVATION 1. *For $f \in \bigcup_s \text{BOX}_n^d$, if there is a $+/^i-$ pair in which the positive point $p_+ = (x_1, \ldots, x_i, \ldots, x_d)$ where $x_j \in \{1, \ldots, n\}$ is the coordinate of p_+ in dimension j, then there must be some side of the target polygon that is on the hyperplane defined by $y_i = x_i$.*

Generalizing Observation 1 we obtain the following.

OBSERVATION 2. *For any two instances p_i and p_j such that $f(p_i) = 0$ and $f(p_j) = 1$, a straight line drawn between p_i and p_j must cross an odd number of sides of f. Thus, such a line must cross at least one side of f.*

4.3. A Learning Algorithm for Unions of Boxes

All of the results in this section are presented in more detail by Goldberg, Goldman and Mathias [19].

In this section we present an algorithm that exactly identifies any concept from $\bigcup_s \text{BOX}_n^d$ while receiving at most sd counterexamples, and using $O((8s)^d + sd \log n)$ membership queries and processing time.

We now describe the hypothesis class used by this algorithm. For each of the d dimensions, we maintain a set of hyperplanes defined by the sides of the target polyhedron that have been identified by the existence of $+/-$ pairs for the given dimension. For $1 \leq i \leq d$, let n_i be the number of hyperplanes that have been defined by $+/^i-$ pairs. Note that $n_i \leq 2s$. For $1 \leq j \leq n_i$, suppose that the n_i hyperplanes found are $y_i = x_j$ where $1 \leq x_j \leq n$. Thus in dimension i we have decomposed $\{1, \ldots, n\}^d$ into up to $2n_i + 1$ regions: $n_i + 1$ corresponding to the regions defined by the hyperplanes and n_i corresponding to the hyperplanes themselves.

For our hypothesis, we would like to divide $\{1, \ldots, n\}^d$ into

$$\Pi_{i=1}^d (2n_i + 1) \leq \Pi_{i=1}^d (4s + 1) = (4s + 1)^d$$

regions, and then just classify all points in a given region as positive (or as negative). To achieve this goal, for $1 \leq i \leq d$ we maintain a balanced binary search tree T_i for dimension i where each internal node of the tree corresponds to one of the hyperplanes (with x_j used for the key), and each external node corresponds to one of the regions defined by the hyperplanes. Also in each leaf node, the key field is replaced by a *range* field that holds a pair (min, max), where min (respectively, max) holds the minimum (respectively, maximum) x_j such that $y_i = x_j$ is a point in the region corresponding to that leaf. (For the internal nodes, the key itself serves the role of both min and max.)

Let $S_i = \{[min_v, max_v] \mid min_v \text{ and } max_v \text{ are the range for node } v \text{ in tree } T_i\}$ and let $\mathcal{T} = \{T_1, \ldots, T_d\}$. Observe that the hyperplanes stored in the internal nodes of $\mathcal{T}$ partition $\{1, \ldots, n\}^d$ into a set of regions $R_\mathcal{T}$ given by the cross product $R_\mathcal{T} = S_1 \times S_2 \times \cdots \times S_d$.

In addition to the trees $T_1, \ldots, T_d$ our hypothesis also maintains a prediction array A with $|R_\mathcal{T}|$ entries where for $r \in R_\mathcal{T}$, $A[r]$ is either 0 (indicating that for any point in r the hypothesis will predict negative), or 1 (indicating that for any point in r the hypothesis will predict positive) or contains a pointer to an element of a queue (to be explained in a moment) that indicates that the classification of the points in region r is not well defined. We use $h_{R_\mathcal{T},A}$ to denote the hypothesis defined by the regions in $R_\mathcal{T}$ with the classifications given in A. Figure 3 shows the set of regions defined by a target concept once all hyperplanes are discovered. The classifications of all of the regions (as stored in A) are also shown in Figure 3. The corresponding trees T_1 and T_2 are shown in Figure 4.3.

Given hypothesis $h_{R_\mathcal{T},A}$, we define a region $r \in R_\mathcal{T}$ to be *valid* if each of the at most 2^d corner points of r have the classification given by $A[r]$. We define $h_{R_\mathcal{T},A}$ to be *valid* if each region in $R_\mathcal{T}$ is valid. For a valid hypothesis $h_{R_\mathcal{T},A}$ and a point $x = (x_1, \ldots, x_d)$ we can compute $h(x)$, the prediction made by hypothesis h on point x, as follows. For $1 \leq i \leq d$

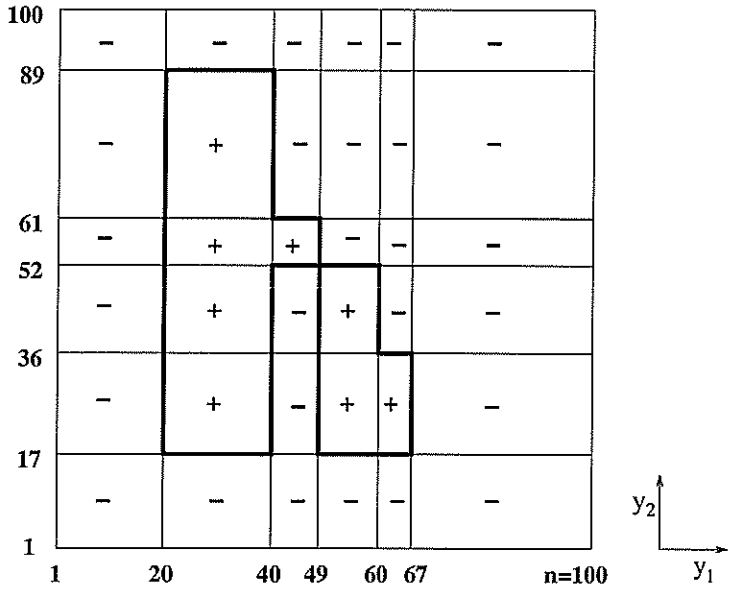

Figure 3: This shows the final set of regions corresponding to the target polyhedron that is outlined in bold. Observe that the target polyhedron consists of 20 segments since each segment is a maximum portion of one of the bold lines that is not intersected by one of the thin lines. The classification of all the two-dimensional regions are shown inside the region. (The classifications of the one-dimensional and zero-dimensional regions are not shown, but are stored in the prediction matrix A.)

we perform a search for x_i in tree T_i to find the node having x_i in its range. Combining the ranges of the d nodes found defines the region $r \in R_T$ that contains x. Finally $h(x) = A[r]$. Observe that, for point x, if k of the binary searches end at non-leaf nodes, and $d-k$ of the binary searches end at leaf nodes, then the region r in which x is contained has dimension at most $d-k$.

A key step of our algorithm is the ability of the learner to construct a valid hypothesis that incorporates all known hyperplanes. We first prove that given a set of hyperplanes (represented in the d binary trees) the learner can efficiently construct a valid hypothesis. To help in the process of making a hypothesis valid, we maintain a queue Q of invalid regions. In addition, for each region $r \in Q$ we store two lists: a list *pos–list* containing all positive corner points of r, and a list *neg–list* containing all negative corner points of r. Observe that by the definition of an invalid region, for all regions in Q both *pos–list* and *neg–list* must be nonempty.

4.3.1. Building a Valid Hypothesis. In this section we provide a procedure that takes an invalid hypothesis $h_{R_T,A}$ and the queue Q of invalid regions from R_T (and thus Q is non-empty), and refines $h_{R_T,A}$ so that it will be valid. In refining $h_{R_T,A}$ our procedure uses membership queries to find *new* hyperplanes and uses them to modify the hypothesis.

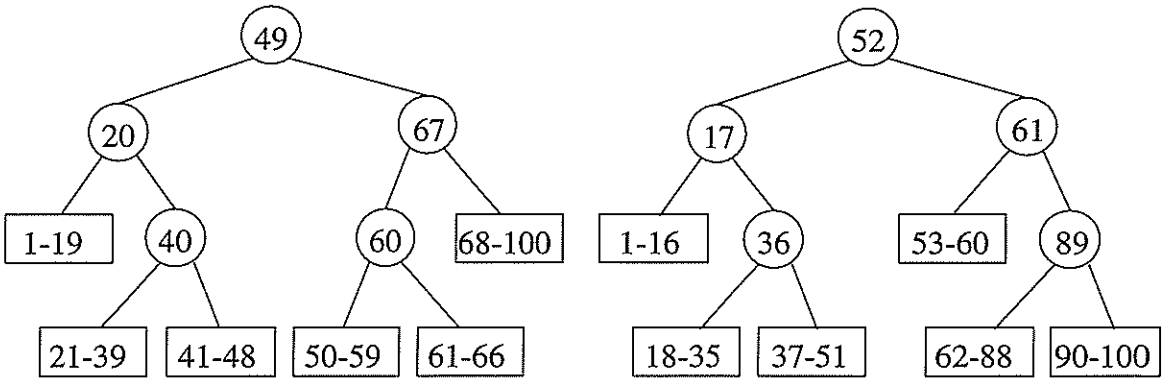

Figure 4: This shows the T_1 and T_2 trees for $S_1 \times S_2$ corresponding to the regions shown in Figure 3. The internal nodes are shown as circles and the external nodes are shown as rectangles. The range for each node is shown inside it.

Our procedure to build a valid hypothesis never removes any hyperplane from any tree in $\mathcal{T}$, and only searches for a new hyperplane in such a way that we are certain that an existing hyperplane will not be rediscovered in the process. We also maintain the invariant that Q always contains exactly one entry for each invalid region of $R_{\mathcal{T}}$.

Our procedure to build a valid hypothesis repeatedly does the following until Q is empty (and thus the hypothesis is valid). Let r be the region at the front of the queue. Since r is an invalid region it must have at least one corner that is positive and one corner that is negative. Using such a positive corner x_+ and such a negative corner x_-, we search for a $+/-$ pair contained within region r. The search for such a $+/-$ pair takes the form of a binary search between x_+ and x_- where comparisons in the binary search are replaced by membership queries. Observe that since the instance space is $\{1, \ldots, n\}^d$, we are guaranteed to find a $+/-$ pair for which both points in the pair are contained within r while using only $O(\log n)$ membership queries and time. Furthermore, the hyperplane defined by this $+/-$ pair is guaranteed to be a hyperplane that has not yet been discovered (by the way we have defined the region). For the remainder of this paper, we shall just speak of performing a binary search between a positive and negative point to find a hyperplane.

We now describe the procedure ADD-HYPERPLANE that modifies our hypothesis to incorporate the new hyperplane found. Without loss of generality, we assume that the hyperplane found by the binary search is $y_i = c$. Let v be the leaf in T_i with the range $[min_v, max_v]$ such that $min_v \leq c \leq max_v$. We begin by using the standard tree insertion procedures for a balanced search tree to update tree T_i so that v becomes an internal node with a key of c, it has left child v_{left} with the range $[min_v, c-1]$, and it has right child v_{right} with the range $[c+1, max_v]$. Thus each region in

$$R_{delete} = S_1 \times \cdots \times S_{i-1} \times \{[min_v, max_v]\} \times S_{i+1} \times \cdots \times S_d$$

is replaced by three regions, giving us the new regions

$$R_{add} = S_1 \times \cdots \times S_{i-1} \times \{[min_v, c-1], [c,c], [c+1, max_v]\} \times S_{i+1} \times \cdots \times S_d.$$

Observe that for each $r \in R_{add}$ that there are at most 2^{d-1} corner points for which a membership query has not yet been performed. Since all regions in R_{delete} no longer exist, we remove any that are in Q by using the pointer provided in A (storing the points in *pos–list* and *neg–list* in the corresponding new region in R_{add}). We then take each region $r \in R_{add}$ making membership queries on the, at most, 2^{d-1} previously unqueried corners and then determine if the new region is valid (in which case the classification can be entered in A) or invalid (in which case it can be enqueued). Once the queue is empty, we know that we have a valid hypothesis and thus have completed the process. The detailed algorithm is shown in Figure 5.

4.3.2. Putting it Together. Our algorithm LEARN-RECTANGLES1 works in the following way. For ease of exposition we artificially extend the instance space from $\{1, \ldots, n\}^d$ to $\{0, 1, \ldots, n, n+1\}^d$ where it is known a priori that any example with a coordinate of 0 or $n+1$ in any dimension is a negative example. (The pseudocode does not explicitly make this check, but one could imagine replacing the calls to MQ by a procedure that first checks for such cases.) Initially, R_T just contains the single region corresponding to the entire instance space. Furthermore, since all of the corners are negative, the intial hypothesis predicts 0 for all instances.

We then repeat the following process until a successful equivalence query is made. Let x be the counterexample received from an equivalence query made with a valid hypothesis. We now discuss how to use membership queries (in the form of a binary search) to find two new hyperplanes defined by the target concept. Without loss of generality, we assume that x is a positive counterexample. (Negative counterexamples are handled in the same manner.) Every counterexample is within one of the regions, say region r, in $S_1 \times \cdots \times S_d$. Since the hypothesis was valid and x is a positive counterexample, we know that all corners of r are classified as negative. Thus we can use two opposing corners of r (in conjunction with x) as the endpoints for binary searches to discover two new hyperplanes. The hypothesis is updated using ADD-HYPERPLANE to incorporate these two hyperplanes. Finally, we call MAKE-VALID-HYPOTHESIS to further refine any invalid regions. Figure 6 gives the complete algorithm.

4.3.3. Using a Hypothesis Class of Unions of Boxes. We now describe how a valid hypothesis can be converted to the union of $O(sd \log s)$ boxes from BOX_n^d. Recall that all equivalence queires are made with valid hypotheses, and thus such a conversion enables our algorithm to learn the union of s boxes from BOX_n^d using as a hypothesis class the union of $O(sd \log s)$ boxes from BOX_n^d.

Recall that a valid hypothesis h essentially encodes the set of positive regions. Thus our goal is to find the union of as few boxes as possible that "cover" all the positive regions. We now describe how to formulate this problem as a set covering problem for which we can then

ADD-HYPERPLANE($h_{R_T,A}, Q, i, c$)

Let v be the leaf of T_i for which $min_v \leq c \leq max_v$
Using a standard balanced tree insertion procedure, update T_i so that
  v is an internal node with key c
  v has a left child with range $[min_v, c-1]$
  v has a right child with range $[c+1, max_v]$
Let $R_{delete} = S_1 \times \cdots \times S_{i-1} \times \{[min_v, max_v]\} \times S_{i+1} \times \cdots \times S_d$
Let $R_{add} = S_1 \times \cdots \times S_{i-1} \times \{[min_v, c-1], [c,c], [c+1, max_v]\} \times S_{i+1} \times \cdots \times S_d$

For each $r \in R_{delete}$
  Let $r_=$, $r_<$ and $r_>$ be the regions in R_{add} for which $(r_= \cup r_< \cup r_>) = r$
  If $A[r] = b$ (for $b = 0$ or $b = 1$)
    Let $A[r_=] = A[r_<] = A[r_>] = b$
  Else (so $A[r]$ is a pointer to element q of Q)
    Generate a new queue node for $r_=$, $r_<$ and $r_>$
    Set the corresponding entries of A to point to these new nodes
    Divide points on *q.pos–list* and *q.neg–list* among the queue entries for $r_<$ and $r_>$
    Remove q from Q

For each $r' \in R_{add}$
  Make membership queries on the 2^{d-1} new corners of r'
  If $A[r']$ points to q (versus being 0 or 1)
    If *q.pos–list* $= \emptyset$ and all new corners are negative, let $A[r'] = 0$
    If *q.neg–list* $= \emptyset$ and all new corners are positive, let $A[r'] = 1$
    Else (r is invalid)
      Add the new positive corners to the *q.pos–list*
      Add the new negative corners to the *q.neg–list*
      Q.ENQUEUE(q)
  Else If ($A[r'] = 0$ and a corner of r' is pos.) or ($A[r'] = 1$ and a corner of r' is neg.)
    Build a queue entry q for r'
    Place each corner of r' on *q.pos–list* if positive or *q.neg–list* if negative
    Let $A[r']$ point to q
    Q.ENQUEUE(q)

Figure 5: Our subroutine to update $h_{R_T,A}$ to incorporate the newly discovered hyperplane $y_i = c$. The new hyperplane is added to tree T_i. Then all regions in R_T that are split are removed from Q. Finally this procedure initializes the new entries of R_T in the prediction matrix A.

LEARN-RECTANGLES1:

Let $Q \leftarrow \emptyset$
For $1 \leq i \leq d$
  Initialize T_i to be a single leaf covering the range 0 to $n+1$
For r be the single region of $R_{\mathcal{T}}$, let $A[r] = 0$

While Equiv$(h_{R_{\mathcal{T}},A}) \neq$ "yes"
  Let x be the counterexample where x is in region r of $h_{R_{\mathcal{T}},A}$
  Let z_1 and z_2 be two opposing corners of r
  For $1 \leq \ell \leq 2$
    Perform binary search between x and z_ℓ to find hyperplane $y_i = c$
    ADD-HYPERPLANE$(h_{R_{\mathcal{T}},A}, Q, i, c)$
  $h_{R_{\mathcal{T}},A} \leftarrow$MAKE-VALID-HYPOTHESIS$(h_{R_{\mathcal{T}},A}, Q)$
Return $h_{R_{\mathcal{T}},A}$

MAKE-VALID-HYPOTHESIS$(h_{R_{\mathcal{T}},A}, Q)$

While $Q \neq \emptyset$
  $q \leftarrow$ DEQUEUE(Q)
  Let p_+ be the first point on the *q.pos–list*
  Let p_- be the first point on the *q.neg–list*
  Perform binary search between p_+ and p_-
    to find the hyperplane $y_i = c$
  ADD-HYPERPLANE$(h_{R_{\mathcal{T}},A}, Q, i, c)$

Figure 6: Algorithm for learning unions of d-dimensional axis-parallel rectangles.

use the standard greedy set covering heuristic [16] to perform the conversion. The set X of the objects to cover will simply contain all positive regions in h. Thus $|X| \leq (4s+1)^d$. Then the set $\mathcal{F}$ of subsets of X will be made as follows. Consider the set B of boxes where each box in B is formed by picking a minimum and maximum coordinate, in each dimension, from the hyperplanes represented in h for that dimension. For any such $b \in B$, if b contains any negative region, then throw it out. Otherwise, place in $\mathcal{F}$ the set of regions contained within b. Thus $|\mathcal{F}| \leq (2s)^{2d}$, and furthermore, it contains a subset of size s that covers all items in X. Finally, we can apply the greedy set covering heuristic [16] to find a set of at most $s(\ln|X|+1) = s(d\ln(4s+1)+1) = O(ds\log s)$ boxes. The time to perform the conversion is $O((2s)^{2d})$. Thus, since at most $sd+1$ equivalence queries are made, the total time spent in converting the internal hypotheses into hypotheses that are unions of rectangles is at most $O(sd(2s)^{2d})$.

4.4. Research Plan

There are several areas in the algorithm presented that may be improved. In Section 4.3.3 we discussed constructing a hypothesis composed of a small number of boxes using an approximation algorithm for the set covering problem. It is clear from the work of Masek [36] that the covering problem we consider is NP-complete. However, it may be possible to exploit some properties of the domain to get an approximation that is better than the logarithmic ratio given by the greedy algorithm.

The running time of our algorithm is due, largely, to the number of regions it creates. If we could create fewer regions then the running time might be decresed. We have considered methods for doing this but it is presently unclear how they affect the performance of our algorithm. Consider that when a counterexample is received, and the corresponding hyperplane found, the learner extends the hyperplane only the length of the region containing that counterexample. Thus, for every counterexample only one new region is created. While this seems very promising at first, it is easy to construct problem instances that would require rediscovery of a hyperplane $O(s)$ times. With this method it is also easy to show that for some concepts the order in which the counterexamples are recieved can make a significant difference in the number of regions created. Discovery of a method for decreasing the number of regions created is a goal of this research.

5. Teaching Unions of Boxes

While it may be possible, due to the power of a computationally unbounded teacher, to devise a T/L pair for $\bigcup_s \text{BOX}_n^d$ where the learner is polynomial in s, $\log n$ and d, it is probably more reasonable to try to design learners that are exponential in either s or d. In fact, simply combining Theorem 3.1 with our algorithm for learning $\bigcup_s \text{BOX}_n^d$ gives an algorithm for teaching this class that is polynomial in s^d and $\log n$. If, however, we could reduce this to be polynomial in 2^d or 2^s then we could handle more than a constant number of dimensions (2^d is polynomial in s for $d = O(\log s)$) or a constant number of boxes (2^s is polynomial in d for $s = O(\log d)$). In this section we propose further study of a T/L pair

for the class $\bigcup_s \text{BOX}_n^d$ in which the learner runs in time polynomial in s and 2^d. This T/L pair is outlined in Section 5.1. We also discuss a plan for designing a T/L pair to learn this class with a learner that is polynomial in d and 2^s.

5.1. Research Plan

Recall that the class $\bigcup_s \text{BOX}_n^d$ is a generalization of DNF formulas. Here we present a T/L pair to learn $\bigcup_s \text{BOX}_n^d$ with a learner that is polynomial in s and 2^d. Thus, this learner is efficient for $O(\log s)$ dimensions. In terms of DNF formulas, this allows learning over a number of variables that is logarithmic in the number of terms. The T/L pair should allow us to achieve the learning task with asymptotically fewer quiries and less time than the learning algorithm of Goldberg, Goldman and Mathias [19] thereby allowing efficient teaching of $\bigcup_s \text{BOX}_n^d$ for non-constant values of d.

5.1.1. Teaching Unions of Boxes: A First Cut. At a high level, the algorithms for the teacher and the learner function as follows. The teacher simply provides examples for every *exterior* corner of the target concept where an exterior corner is one that borders negative instances (i.e. is not contained within another box). The teacher provides enough examples to identify every corner as either convex or concave and to give its orientation (*e.g.* upper-right corner). See Figure 7 for examples of concave and convex corners. Note that this can be done with at most $(d+2)\cdot 2^d$ examples for each box. To see this consider the concave corner in Figure 7. Notice that positive examples are used at the corner, distance one form the corner in each dimension and a negative is used that is distance one from the corner in every dimension. Thus, the total number of examples provided by the teacher is at most $(d+2)\cdot s\cdot 2^d = O(sd\cdot 2^d)$. Notice that the teacher gives enough information for the learner to be able to ignore any adversarial examples. This is because the teacher has shown the learner every exterior corner. The adversary cannot add examples that look like a corner but that are not a corner.

Assume, for the moment, that the entire positive space is contiguous. The learner reads in all of the examples and constructs corners by placing adjacent examples together in a record (provided that the examples form a valid corner configuration as seen in Figure 7). Once all of the examples have been processed the learner deletes any corners that are incomplete. Since, as mentioned, the adversary cannot create a false corner all adversarial examples are discarded. The learner now finds a minimum corner (closest to the origin) and for every other corner calculates the number of dimensions in which it differs from the minimum corner (this is the distance to the minimum corner when all sides have length one). Sorting the corners by this distance yields a breadth-first ordering. Using this breadth-first ordering on the corners the learner can "create" the sides of the positive region. To make these ideas viable it is necessary to develop a hypothesis that can be maintained and evaluated without asymptotically increasing the running time of the algorithm.

In the previous discussion of the learning algorithm we assumed that the entire positive space was contiguous. If this is not the case the learner can use the orientation of each corner to determine in which directions its "matching" corners can be found. Thus, it is

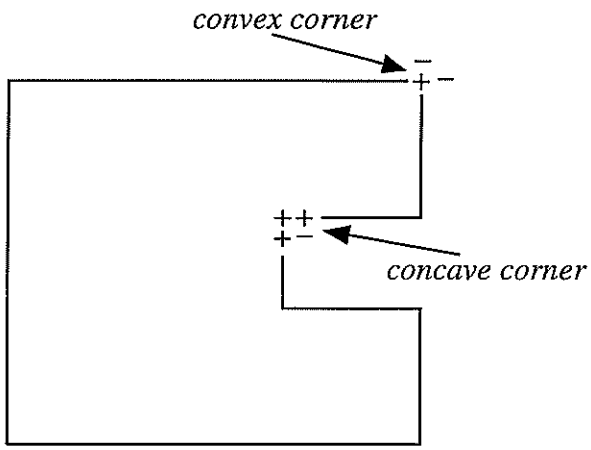

Figure 7: An example showing concave and convex corners and the examples given by the teacher for each. For a concave corner the teacher provides the positive example at the corner, the (at most) d positive examples each distance one from the corner and the negative example that is distance one from the corner in each dimension. For a convex corner the teacher again provides the positive example at the corner and also provides the d negative examples each distance one from the corner. Note that each corner uses at most $(d+2)$ examples.

impossible for a corner of a neighboring, non-contiguous positive region to be confused with a corner of the region under consideration by the learner.

5.1.2. Can We Do Better?. An alternate approach for teaching $\bigcup_s \text{BOX}_n^d$ is to attempt to find a T/L pair in which the learner runs in time polynomial in d and 2^s. This allows efficient teaching of concepts with the number of boxes logarithmic in the number of dimensions (this is analogous to DNF with a number of terms logarithmic in the number of variables).

Blum and Rudich [9] give a randomized learning algorithm for learning DNF formulas that runs in time $O(n \cdot 2^{O(k)})$ where n is the number of Boolean variables and k is the number of terms in the target DNF. This algorithm is efficient for values of k as large as $O(\log n)$. The randomization in this algorithm is used by the learner to find helpful examples. Because we have a powerful teacher choosing the examples to present to the learner it is possible that our teacher can do this work and thus allow our learner to be deterministic. We must then extend the algorithm to work for the broader class of $\bigcup_s \text{BOX}_n^d$. This may introduce a dependence on the size of the domain, n (note that this is not the same n as the number of variables in the DNF case), likely polynomial in $\log n$. It is also possible that modifications to the T/L pair discussed earlier in this section could produce a learner with the desired complexity.

6. Teaching Disjunctive Normal Form Formulas

One of the largest open problems in computational learning theory is the learnability of disjunctive normal form (DNF) formulas. There is some evidence that DNF formulas with an arbitrary number of terms are not learnable in existing learning models. Therefore, much of the research in this area has been directed at learning various subclasses of DNF formulas such as k-term DNF (formulas with a constant number of terms), read-μ DNF (formulas where each variable appears at most μ times) and monotone DNF (formulas in which all variables are unnegated). However, in the model of teaching of Goldman and Mathias [20] (discussed in Section 3) it is possible that the power of the cooperative teacher may be employed to aid the learner sufficiently to allow learning of DNF formulas.

6.1. Previous Work

There has not been any known previous work on the problem of teaching DNF formulas. There has been a great deal of previous work, however, on learning subclasses of DNF formulas in standard learning models.

In the PAC learning model there are algorithms for several subclasses of DNF formulas. Monotone DNF was shown learnable by Valiant [42] in the PAC model using membership queries. Hancock [26] has given a PAC algorithm for the class read-twice DNF. Angluin [4] proved that an algorithm for exact identification of a class using equivalence queries implies the PAC learnability of the class (if the query algorithm also uses membership queries then a similar result holds for the PAC model with membership queries). Thus, there are other positive results for PAC learning subclasses of DNF formulas implicit in results for the query model.

Angluin has shown that the class of monotone DNF is learnable in the query model using both equivalence and membership queries [4]. For the class of read-twice DNF, Aizenstein and Pitt [2] give an exact identification algorithm. Surprisingly, perhaps, Aizenstein, Hellerstein and Pitt [1] show that read-thrice DNF is not learnable by any equivalence and membership query algorithm, unless NP = co-NP, when all equivalence queries must use hypotheses from the class read-thrice DNF. (Note that there is no known exact-identification algorithm for read-twice DNF when hypotheses must also be read-twice DNF). The class of k-term DNF has been extensively studied. Blum and Singh [10] have given an algorithm for learning this class by the class of DNF formulas using only equivalence queries and $O(n^k)$ time. Angluin [3] gives an algorithm to exactly identify any k-term DNF formula in time $O(n^{k^2})$ using equivalence queries and membership queries and a hypothesis class of k-term DNF. Notice that both of these algorithms are efficient only for constant values of k. Finally, Blum and Rudich [9] give a randomized algorithm that exactly identifies the class k-term DNF, using equivalence and membership queries, in $O(n \cdot 2^{O(k)})$ expected time. This algorithm returns a hypothesis of $O(2^{O(k)})$ terms but is efficient for $k = O(\log n)$.

Pitt and Valiant [37] gave a representational-dependent hardness result for k-term DNF. They show that k-term DNF is not learnable by k-term DNF using only equivalence queries unless RP = NP. Since Angluin gave an algorithm to learn this class using equivalence

and membership queries and hypotheses from the concept class, we know that membership queries help in learning k-term DNF formulas when the hypothesis class is k-term DNF formulas. Angluin and Kharitonov [6], however, have shown that membership queries do not help in learning DNF formulas with an arbitrary number of terms. They show, under cryptographic assumptions, that the learner is unable to determine helpful instances on which to ask membership queries.

6.2. Research Plan

We have already expended some effort considering the problem of teaching DNF formulas and have discovered approaches that seem not to work. This is due in part to the existence of DNF formulas with an exponential number of prime implicants the disjunction of which are not sufficient to cover the function. In fact, formulas exist in which these terms are not only insufficient but also unnecessary. In our teaching model it is not hard to see that the adversary may cause these terms to be created and, thus, prevent the learner from creating a polynomial size hypothesis.

The result of Angluin and Kharitonov suggests that membership queries might help if the learner was able to formulate useful queries. This is encouraging since in our model the teacher can provide the learner with the answer to a membership query (ie. include in the teaching set a labeled example) in a way that does not require the learner to explicitly ask the query. Thus, by searching the teaching set the learner may be able to find a useful membership query. It may also be helpful to allow the learner to use other types of queries. That is, the learner may gain some power by treating the examples in the teaching set as answers to subset queries or superset queries. Note that this does not require any additional resources since we already have a computationally unbounded teacher that can simulate the learner and provide the needed answers to any example-based query.

There are also several interesting subclasses of DNF formulas that may be interesting to investigate. The negative result for read-thrice DNF raises the question of teachability of this class. It is possible that an approach that fails for general DNF may be promising for teaching read-thrice DNF (though even read-twice DNF formulas may have an exponential number of unnecessary prime implicants). Also, Blum and Rudich note that the learnability of DNF formulas with $polylog(n)$ terms is an open question. The teachability of either of these classes might provide some insight into the possibility of teaching general DNF.

Finally, it is clear from Theorem 3.1 that a negative teaching result for any of these classes shows that they are not learnable using example-based queries. (The implied negative result for learning would be representation-independent only if the negative teaching result was also representation-independent.)

7. A Model of Randomized Teaching

In the design of algorithms it is often useful to allow some amount of randomization. In learning algorithms randomization can be used in two ways: to help the learner find a useful

example (a randomized teacher) or to decide for the learner how to use the examples once they are found (a randomized learner). Because the teaching algorithm in our model can supply the learner with useful examples, the first type of randomization is probably not useful for a model of teaching [9]. The second type of randomization may, however, give additional power to our model.

It is not possible to simply allow randomization of the learning algorithm in our model as presently defined, however. The problem is that the teacher may need to simulate the learner in order to choose helpful examples. Obviously, if the learner is randomized the teacher cannot accurately simulate the learner. In this section we introduce one possible teaching model that allows randomization of the learner.

7.1. Research Plan

The following model is proposed as a starting point for investigation of this problem. In each phase of the process: the teacher builds a *partial* teaching set that gives the learner the required information until the first coin flip in the learning algorithm; the adversary may add to this partial teaching set in the same way as in Section 3.2; the learner uses the teaching set from the adversary deterministically and then makes the coin flip; the teacher observes the coin flip and the next phase begins. Such a teaching session is shown in Figure 8.

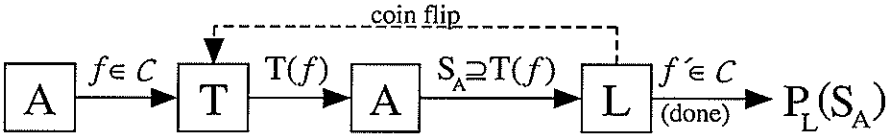

Figure 8: An overview of a teaching session with a randomized learner. The adversary chooses the target function, f, giving it to the teacher. The teacher then generates a partial teaching set $T(f)$. Given the partial $T(f)$ the adversary generates $S_A \supseteq T(f)$ and gives this to the learner. The learner uses S_A until the first random move, which is witnessed by the teacher, and then the process is repeated. When finished the learner outputs a representation from C thus defining a probability distribution $P_L(S_A)$ over C.

It is unclear if using the randomization in this way increases the power of the T/L pair (i.e. makes some class teachable that was unteachable in the model of Goldman and Mathias). It is also unknown if the adversary in this model is capable of preventing collusion. Much work remains to be done in the development of a randomized teaching model.

If it is possible to develop an appropriate randomized teaching model then we can teach the class of DNF formulas in that model. Recently Bshouty, Cleve and Tamon [13] have given a randomized algorithm for exactly identifying DNF formulas using only restricted superset and subset queries. Recall that a superset query on hypothesis h for target function

[9] This type of randomization might be useful to make an otherwise inefficient teacher run in expected polynomial time. However, we are generally less concerned with the complexity of the teaching algorithm.

f answers "yes" if $h \supseteq f$ and otherwise returns a counterexample x such that $x \in f - h$. Subset queries are analogous. Since both superset and subset queries are example-based queries we get (with an appropriate model) that DNF formulas are teachable. In our model it would not be necessary to rely on superset and subset oracles since the teacher would perform this task.

8. Summary

It is expected that the work proposed here will provide answers to several open questions in computational learning theory. The main contributions of this work will be:

- The first model of teaching that avoids collusion (as defined) but is able to benefit significantly from the presence of a helpful teacher.
- The first efficient (for constant d) algorithm to learn exactly the union of an arbitrary number of boxes in d dimensions.
- Algorithms, in the above teaching model, to teach the class of unions of boxes in d dimensions that are efficient for either a number of dimensions that is logarithmic in the number of boxes or a number of boxes that is logarithmic in the number of dimensions.
- An algorithm to teach efficiently the class of DNF formulas.
- A model of teaching in which the learner may be randomized but that still prevents collusion while taking advantage of the power of the teacher.

References

[1] Howard Aizenstein, Lisa Hellerstein, and Leonard Pitt. Read-thrice DNF is hard to learn with membership and equivalence queries. In *33nd Annual Symposium on Foundations of Computer Science*, pages 523–532, October 1992.

[2] Howard Aizenstein and Leonard Pitt. Exact learning of read-twice DNF formulas. In *32nd Annual Symposium on Foundations of Computer Science*, pages 170–179, October 1991.

[3] Dana Angluin. Learning k-term DNF formulas using queries and counterexamples. Technical Report YALEU/DCS/RR-559, Yale University, August 1987.

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

[5] Dana Angluin, Michael Frazier, and Leonard Pitt. Learning conjunctions of horn clauses. *Machine Learning*, 9:147–164, 1992. Special Issue for COLT 90.

[6] Dana Angluin and Michael Kharitonov. When won't membership queries help? In *Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing*, pages 444–454, May 1991.

[7] Martin Anthony, Graham Brightwell, Dave Cohen, and John Shawe-Taylor. On exact specification by examples. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 311–318. ACM Press, July 1992.

[8] Peter Auer. On-line learning of rectangles in noisy environments. In *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory*, pages 253–261, July 1993.

[9] Avrim Blum and Steven Rudich. Fast learning of k-term DNF formulas with queries. In *Proceedings of the Twenty Fourth Annual ACM Symposium on Theory of Computing*, pages 382–389, May 1992.

[10] Avrim Blum and Monah Singh. Learning functions of k terms. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 144–153, August 1990.

[11] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the Association for Computing Machinery*, 36(4):929–965, October 1989.

[12] Nader H. Bshouty. Exact learning via the monotone theory. In *34th Annual Symposium on Foundations of Computer Science*, November 1993.

[13] Nader H. Bshouty, Richard Cleve, and Christino Tamon. Oracles and queries that are sufficient for exact learning. Unpublished manuscript, November 1993.

[14] Zhixiang Chen. Learning unions of two rectangles in the plane with equivalence queries. In *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory*, pages 243–252. ACM Press, July 1993.

[15] Zhixiang Chen and Wolfgang Maass. On-line learning of rectangles. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 16–27. ACM Press, July 1992.

[16] V. Chvatal. A greedy heuristic for the set covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.

[17] Mike Frazier, Sally Goldman, Nina Mishra, and Leonard Pitt. Learning from a consistently ignorant teacher. Unpublished manuscript (submitted to COLT '94), July 1993.

[18] Rūsiņš Freivalds, Efim Kinber, and Rolf Wiehagen. Inference from good examples. *Theoretical Computer Science*, 110:131–144, 1993.

[19] Paul W. Goldberg, Sally A. Goldman, and H. David Mathias. Learning unions of rectangles with membership and equivalence queries. Technical Report WUCS-93-46, Washington University, November 1993.

[20] S. Goldman and D. Mathias. Teaching a smarter learner. In *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory*, pages 67–76. ACM Press, New York, NY, 1993.

[21] S. A. Goldman, R. L. Rivest, and R. E. Schapire. Learning binary relations and total orders. *SIAM Journal on Computing*, 22(5):1006–1034, October 1993.

[22] Sally A. Goldman and Michael J. Kearns. On the complexity of teaching. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 303–315. Morgan Kaufmann, August 1991. To appear in *Journal of Computer and System Sciences.*

[23] Sally A. Goldman, Ronald L. Rivest, and Robert E. Schapire. Learning binary relations and total orders. In *30th Annual Symposium on Foundations of Computer Science*, pages 46–51, October 1989. To appear in *SIAM Journal of Computing.*

[24] Sally A. Goldman and Robert H. Sloan. The power of self-directed learning. Technical Report WUCS–92–49, Washington University, Department of Computer Science, November 1992. To appear in *Machine Learning.*

[25] S. Goldwasser, S. Goldwasser, and C. Rackoff. The knowledge complexity of interactive proofs. In *26th Annual Symposium on Foundations of Computer Science*, pages 291–304, October 1985.

[26] Thomas R. Hancock. Learning 2μ DNF formulas and $k\mu$ decision trees. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 199–209, August 1991.

[27] Steven Homer and Zhixiang Chen. Fast learning unions of rectangles with queries. Unpublished manuscript, July 1993.

[28] Jeffrey Jackson and Andrew Tomkins. A computational model of teaching. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 319–326. ACM Press, July 1992.

[29] Stefan Lange and Rolf Wiehagen. Polynomial-time inference of arbitrary pattern languages. *New Generation Computing*, 8:361–370, 1991.

[30] Nick Littlestone. Learning when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[31] Philip M. Long and Manfred K. Warmuth. Composite geometric concepts and polynmomial predictability. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 273–287. Morgan Kaufmann, August 1990.

[32] Wolfgang Maass and György Turán. On the complexity of learning from counterexamples. In *30th Annual Symposium on Foundations of Computer Science*, pages 262–267, October 1989.

[33] Wolfgang Maass and György Turán. On the complexity of learning from counterexamples and membership queries. In *31st Annual Symposium on Foundations of Computer Science*, pages 203–210, October 1990.

[34] Wolfgang Maass and György Turán. Algorithms and lower bounds for on-line learning of geometrical concepts. Technical Report IIG-Report 316, Technische Universität Graz, TU Graz, Austria, October 1991.

[35] Wolfgang Maass and György Turán. Lower bound methods and separation results for on-line learning models. *Machine Learning*, 9:107–145, 1992.

[36] William J. Masek. Some np-complete set covering problems. Unpublished manuscript, 1978.

[37] Leonard Pitt and Leslie G. Valiant. Computational limitations on learning from examples. *Journal of the Association for Computing Machinery*, 35(4):965–984, 1988.

[38] Ronald L. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, 1987.

[39] Kathleen Romanik. Approximate testing and learnability. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 327–332. ACM Press, July 1992.

[40] Kathleen Romanik and Carl Smith. Testing geometric objects. Technical Report UMIACS-TR-90-69, University of Maryland College Park, Department of Computer Science, 1990.

[41] Ayumi Shinohara and Satoru Miyano. Teachability in computational learning. *New Generation Computing*, 8:337–347, 1991.

[42] Leslie Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.