Washington University in St. Louis

Washington University Open Scholarship

All Computer Science and Engineering
Research

Computer Science and Engineering

Report Number: WUCS-94-19

1994-01-01

Exact Learning of Discretized Geometric Concepts

Nader H. Bshouty, Paul W. Goldberg, Sally A. Goldman, and H. David Mathias

We first present an algorithm that uses membership and equivalence queries to exactly identify a discretized geometric concept defined by the union of m axis-parallel boxes in d -dimensional discretized Euclidean space where each coordinate can have n discrete values. This algorithm receives at most md counterexamples and uses time and membership queries polynomial in m and $\log(n)$ for any constant d . Furthermore, all equivalence queries can be formulated as the union of $O(md \log(m))$ axis-parallel boxes. Next, we show how to extend our algorithm to efficiently learn, from only equivalence queries, any discretized geometric concept generated from any number of... [Read complete abstract on page 2.](#)

Follow this and additional works at: https://openscholarship.wustl.edu/cse_research

Part of the [Computer Engineering Commons](#), and the [Computer Sciences Commons](#)

Recommended Citation

Bshouty, Nader H.; Goldberg, Paul W.; Goldman, Sally A.; and Mathias, H. David, "Exact Learning of Discretized Geometric Concepts" Report Number: WUCS-94-19 (1994). *All Computer Science and Engineering Research*.

https://openscholarship.wustl.edu/cse_research/340

Department of Computer Science & Engineering - Washington University in St. Louis
Campus Box 1045 - St. Louis, MO - 63130 - ph: (314) 935-6160.

Exact Learning of Discretized Geometric Concepts

Nader H. Bshouty, Paul W. Goldberg, Sally A. Goldman, and H. David Mathias

Complete Abstract:

We first present an algorithm that uses membership and equivalence queries to exactly identify a discretized geometric concept defined by the union of m axis-parallel boxes in d -dimensional discretized Euclidean space where each coordinate can have n discrete values. This algorithm receives at most md counterexamples and uses time and membership queries polynomial in m and $\log(n)$ for any constant d . Furthermore, all equivalence queries can be formulated as the union of $O(md\log(m))$ axis-parallel boxes. Next, we show how to extend our algorithm to efficiently learn, from only equivalence queries, any discretized geometric concept generated from any number of halfspaces with any number of known (to the learner) slopes in a constant dimensional space. In particular, our algorithm exactly learns (from equivalence queries only) unions of discretized axis-parallel boxes in constant dimensional space in polynomial time. Further, this algorithm can be modified to handle a polynomial number of lies in the counterexamples provided by the environment. Finally, we introduce a new complexity measure that better captures the complexity of the union of m boxes than simply the number of boxes and the dimension. Our new measure, σ , is the number of segments in the target, where a segment is a maximum portion of one of the sides of the target that lies entirely inside or entirely outside each of the other halfspaces defining the target. We present a modification of our first algorithm that uses time and queries polynomial in σ and $\log(n)$. In fact, the time and queries (both membership and equivalence) used by this single algorithm are polynomial for either m or d constant.

**Exact Learning of Discretized Geometric
Concepts**

**Nader H. Bshouty, Paul W. Goldberg, Sally A. Gold-
man and H. David Mathias**

WUCS-94-19

July 1994

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

Exact Learning of Discretized Geometric Concepts*

Nader H. Bshouty[†]
Department of Computer Science
The University of Calgary
Calgary, Alberta, Canada T2N 1N4
bshouty@cpsc.ucalgary.ca

Paul W. Goldberg[‡]
Department 1423
Sandia National Labs, MS 1110
P.O. Box 5800
Albuquerque, NM 87185-1110
pwgoldb@cs.sandia.gov

Sally A. Goldman[§]
Dept. of Computer Science
Washington University
St. Louis, MO 63130
sg@cs.wustl.edu

H. David Mathias
Dept. of Computer Science
Washington University
St. Louis, MO 63130
dmath@cs.wustl.edu

WUCS-94-19

July 5, 1994

Abstract

We first present an algorithm that uses membership and equivalence queries to exactly identify a discretized geometric concept defined by the union of m axis-parallel boxes in d -dimensional discretized Euclidean space where each coordinate can have n discrete values. This algorithm receives at most md counterexamples and uses time and membership queries polynomial in m and $\log n$ for d any constant. Furthermore, all equivalence queries can be formulated as the union of $O(md \log m)$ axis-parallel boxes.

Next, we show how to extend our algorithm to efficiently learn, from *only* equivalence queries, any discretized geometric concept generated from any number of halfspaces with any number of known (to the learner) slopes in a constant dimensional space. In particular, our algorithm exactly learns (from equivalence queries *only*) unions of discretized axis-parallel boxes in constant dimensional space in polynomial time. Further, this algorithm can be modified to handle a polynomial number of lies in the counterexamples provided by the environment.

Finally, we introduce a new complexity measure that better captures the complexity of the union of m boxes than simply the number of boxes and the dimension. Our new measure, σ , is the number of segments in the target where a segment is a maximum portion of one of the sides of the target that lies entirely inside or entirely outside each of the other halfspaces defining the target. We present a modification of our first algorithm that uses time and queries polynomial in σ and $\log n$. In fact, the time and queries (both membership and equivalence) used by this single algorithm are polynomial for *either* m or d constant.

*Portions of this paper appear in preliminary form in [16] and [7].

[†]This research was supported in part by the NSERC of Canada.

[‡]This research was performed while visiting Washington University. Currently supported by the U.S. Department of Energy under contract DE-AC04-76AL85000.

[§]Supported in part by NSF Grant CCR-9110108 and an NSF NYI Grant CCR-9357707.

1 Introduction

Recently, learning geometric concepts in d -dimensional Euclidean space has been the subject of much research. We study the problem of learning geometric concepts under the model of learning with queries [2] in which the learner is required to output a final hypothesis that correctly classifies *every* point in the domain. To apply such a learning model to a geometric domain, it is necessary to look at a discretized (or digitalized) version of the domain. We use d to denote the number of dimensions and n to denote the number of discrete values that exist in each dimension. Thus a discretized geometric concept G is a set of integer points $G \subseteq \mathcal{N}_n^d$ where $\mathcal{N}_n = \{1, \dots, n\}$. In this paper we consider discretized geometric concepts whose boundaries are defined by hyperplanes of known slope.

We begin by studying the special case of the well studied class of unions of axis-parallel boxes. (By a “box”, we mean an axis-aligned hypercuboid.) This algorithm is easily extended to learn discretized geometric concepts defined by axis-parallel hyperplanes. We use BOX_n^d to denote the class of axis-parallel boxes¹ over $\mathcal{N}_n^d$, and $\bigcup_m \text{BOX}_n^d$ denote the class of the union of at most m 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 and a special case of the class of unions of intersections of half-spaces over $\mathcal{N}_n^d$.

We first present a query algorithm that exactly learns $\bigcup_m \text{BOX}_n^d$ with at most $md + 1$ equivalence queries², and $O((4m)^d + md \log n)$ membership queries and computation time. Thus our algorithm exactly learns the union of m discretized axis-parallel boxes over $\mathcal{N}_n^d$ in polynomial time for any constant d .

The hypothesis class used by this algorithm can be evaluated in time $O(d \log m)$. Furthermore, in $O((2m)^{2d})$ time we can transform our hypothesis to the union of at most $O(md \log m)$ boxes. Thus we obtain the even stronger result that our algorithm can exactly learn the union of m axis-parallel boxes over $\mathcal{N}_n^d$ while making at most $md + 1$ equivalence queries where each equivalence query is simply the union of $O(md \log m)$ concepts from BOX_n^d , making $O((4m)^d + md \log n)$ membership queries, and using $O(md \cdot (2m)^{2d} + md \log n)$ computation time. Thus for any constant d , this algorithm still uses time and queries polynomial in m and $\log n$. We also describe a variation of this basic algorithm that uses *only* equivalence queries and still has complexity polynomial in m and $\log n$ for d constant.

Next we study the problem of learning with only equivalence queries the class of discretized geometric concepts in which the hyperplanes defining the boundaries of the concept need *not* be axis parallel but rather can be defined by m hyperplanes of any known slopes. That is, the geometric discretized concepts we study here are those whose boundaries lie on hyperplanes

¹Note that we include in BOX_n^d boxes with zero size in any dimension.

²The final equivalence query is the correct hypothesis, and thus at most md counterexamples are received.

$\sum_{j=1}^d a_{i,j}x_j = b$ for $i = 1, \dots, |S|$. The possible slopes of those hyperplanes, *i.e.* $a_i = (a_{i,1}, \dots, a_{i,d})$, are known to the learner, but the same slope with different shifts b can be used for many hyperplanes in the target concept g . Note that if we choose the slopes $S = \{e_i\}$, the standard basis, then we get the special case in which all hyperplanes are axis-parallel.

Let $S \subset \mathcal{Z}^d$ where $\mathcal{Z}$ is the set of integer numbers, and let $\|S\|$ denote the size of S (the sum of the logarithms of the absolute values of the integers in S). Let g be a geometric concept whose boundaries lie on m hyperplanes in $\mathcal{N}_n^d$ with slopes from S . A key result of this paper is that for d any constant, any such geometric concept is exactly learnable in $\text{poly}(l, m, \|S\|, \log n)$ time and equivalence queries even if the equivalence oracle lies on l counterexamples. So for example, if the space is the plane $\mathcal{N}_n^2$ and $S = \{(0, 1), (1, 0), (1, 1), (1, -1), (1, 2), (2, 1), (1, -2), (2, -1)\}$ then our algorithm can efficiently learn the geometric concepts, generated from lines that make angles 90; 0; 135; 45; 120; 150; 30 and 60, respectively, with the x -axis, in polynomial time. (As this example illustrates, the representation that we use for a slope is that derived from the formula, given above, defining a hyperplane.) In higher constant dimensional space our algorithm can efficiently learn any geometric concept whose boundary slopes are known. Another generalization of this result is an algorithm to exactly learn polynomially sized decision trees over the basis “Is $y_i \succ c$ ” where $\succ \in \{>, <, \geq, \leq\}$ (in the nodes) in constant dimensional space.

Finally, we re-examine our first algorithm for learning the union of m discretized boxes. We introduce a new complexity measure that better captures the complexity of a union of boxes than simply the number of boxes and dimensions. More specifically, our new measure, σ , is the number of segments in the target concept where a segment is a maximum portion of one of the defining hyperplanes of the target that lies entirely inside or entirely outside each of the other defining hyperplanes. It is easily seen that $\sigma \leq (2m)^d$. We present an improvement of our first algorithm that uses time and queries polynomial in σ and $\log n$. The hypothesis class used by this modified algorithm is decision trees of height at most $2md$. Thus, observe that the hypothesis output (and the intermediate hypotheses) can be evaluated in time polynomial without any restrictions on m or d . We then use an alternate analysis of this algorithm to show that the time and queries used are polynomial in d and $\log n$ for m any constant thus generalizing the exact learnability of DNF formulas with a constant number of terms. Combining these two methods of analysis, we get the interesting result that this single algorithm is efficient for *either* m or d constant.

The paper is organized as follows. In Section 2 we describe the learning model that we use. Next, in Section 3 we summarize the previous work on learning geometric concepts. Then in Section 4 we give some preliminary definitions. Section 5 describes our results for learning unions of boxes with membership and equivalence queries. We present this algorithm, in part, because it introduces the approach used to obtain our other results and also because it uses

very few equivalence queries which is of interest if one's goal is to minimize the number of prediction errors made by the learner [8]. Next in Section 6 we describe a modification of this algorithm that efficiently learns the union of boxes in constant dimensional space with only equivalence queries. In Section 7 we present our extensions to learn the class of geometric concepts defined by any hyperplanes of known slopes using only equivalence queries. In Section 8 we describe how to modify this algorithm to handle the situation in which there are lies in the answers to the equivalence queries. In Section 9 we present our new complexity measure and describe a modification of our first algorithm that runs in polynomial time with respect to this complexity measure. Finally, in Section 10 we conclude with some open problems.

2 Learning Model

The learning model we use in this paper is that of learning with queries developed by Angluin [2]. When applied to our class of discretized geometric concepts, the learner's goal is to learn *exactly* how an unknown target concept, g , drawn from the *concept class* $\mathcal{G} \subseteq 2^{\mathcal{N}_n^d}$, classifies as positive or negative all instances from the *instance space* $\mathcal{N}_n^d$. Thus each concept is the set of instances from $\mathcal{N}_n^d$ that it classifies as positive. We say that $x \in \mathcal{N}_n^d$ is a positive instance for target concept g if $x \in g$ (also denoted $g(x) = 1$) and say that x is a negative instance otherwise (also denoted $g(x) = 0$). It is often convenient to view the target concept g as the Boolean function $g : \mathcal{N}_n^d \rightarrow \{0, 1\}$. A *hypothesis* h is a polynomial-time algorithm that, given any $x \in \mathcal{N}_n^d$, outputs a prediction for $g(x)$.

As mentioned above, the learning criterion in this paper is that of *exact identification*. In order to achieve exact identification, the learner's final hypothesis, h , must be such that $h(x) = g(x)$ for all instances $x \in \mathcal{N}_n^d$. To achieve this goal the learner is provided with two types of queries with which to learn about g . A *membership query*, $\text{MQ}(x)$, returns "yes" if $g(x) = 1$ and returns "no" if $g(x) = 0$. An *equivalence query*, $\text{EQ}(h)$, returns "yes" if h is logically equivalent to g or returns a counterexample otherwise. A *positive counterexample* x is an instance such that $g(x) = 1$ and $h(x) = 0$. Similarly, a *negative counterexample* is such that $g(x) = 0$ and $h(x) = 1$.

An *l -liar* equivalence oracle is an oracle that is allowed to lie at most l times during the learning session when providing a counterexample to an equivalence query.

Another important learning model is the PAC model introduced by Valiant [29]. 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 instance space. That is, the weight, under $\mathcal{D}$, of misclassified instances must be at most ϵ . The

learner is permitted time polynomial in $1/\epsilon$, $1/\delta$ and relevant size measures to formulate a hypothesis.

3 Previous Work

The problem of learning geometric concepts over a discrete domain was extensively studied by Maass and Turán [22, 23, 24, 25]. 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 [19, 24]. Auer [3] 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, then the resulting algorithm makes $O(dn)$ equivalence queries. An algorithm making $O(2^d \log n)$ equivalence queries was given by Maass and Turán [21, 23]. The best known result for learning the class BOX_n^d was provided by the work of Chen and Maass [12] 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 $\{1, \dots, n\} \times \{1, \dots, m\}$ in time polynomial in $\log n$ and $\log m$, where one rectangle has a corner at $(0, m)$ and the other has a corner at $(n, 0)$. More recently, Chen [10] 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.

In work independent of ours, Chen and Homer [15] have given an algorithm to learn the union of m rectangles in the plane using $O(m^3 \log n)$ queries (both membership and equivalence) and $O(m^5 \log n)$ time. The hypothesis class of their algorithm is the union of $8m^2 - 2$ rectangles. Also in work independent of our work, Chen and Homer [11] have improved upon their earlier result by giving an algorithm that learns any concept from $\bigcup_m \text{BOX}_n^d$ using $O(m^{2(d+1)} d^2 \log^{2d+1} n)$ equivalence queries by applying techniques from recursive function theory. Finally, in other independent work, Maass and Warmuth [27] have developed, as part of a more general result, an algorithm to learn any concept from $\bigcup_m \text{BOX}_n^d$ using $O(md \log n)$ equivalence queries and $O((md \log n)^{O(d)})$ computation time. In addition their technique enables them to efficiently learn a single box in a constant dimensional space that is not axis parallel but uses slopes from a known set of slopes.

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 [29]. Blumer et al. [6] present an algorithm to PAC-learn an m -fold union of boxes in E^d by drawing a sufficiently large sample of size $m' = \text{poly}\left(\frac{1}{\epsilon}, \lg \frac{1}{\delta}, m, d\right)$, and then performing a greedy covering over the at most $\left(\frac{em'}{2d}\right)^{2d}$ boxes defined by the sample. Thus for d constant this algorithm runs in

polynomial time. Long and Warmuth [20] present an algorithm to PAC-learn this same class by again drawing a sufficiently large sample and constructing a hypothesis that consists of at most $m(2d)^m$ boxes consistent with the sample. Thus both the time and sample complexity of their algorithm depend polynomially on $m, d^m, \frac{1}{\epsilon}$, and $\lg \frac{1}{\epsilon}$. So for m constant this yields an efficient PAC algorithm.

We note that either of these PAC algorithms can be applied to the class $\bigcup_m \text{BOX}_n^d$ giving efficient PAC algorithms for this class for either d constant or m constant. As discussed by Maass and Turán [24] the task of a concept learning algorithm is to provide a “smart” hypothesis based on the data available. The results from Blumer et. al [6] show that under the PAC model any concise hypothesis that is consistent with the data is “smart enough”. In other words, the PAC model provides no suitable basis for distinction among different consistent hypotheses. On the other hand, a method for defining a “smart” hypothesis is implicitly contained within the exact learning model. One must select hypotheses for the equivalence queries so that sufficient progress is made with each counterexample. This requirement of selecting a “smart” hypothesis makes the problem of obtaining an efficient algorithm to *exactly* learn the class $\bigcup_m \text{BOX}_n^d$ significantly harder than obtaining the corresponding PAC result. Also Blum [4] has shown that if one-way functions exist then there exist functions that are PAC-learnable but not exactly learnable.

Finally, under a variation of the PAC model in which membership queries can be made, Frazier et al. [14] have given an algorithm to PAC-learn the m -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 m and d . Recall that since $\bigcup_m \text{BOX}_n^d$ generalizes DNF, a polynomial-time algorithm for arbitrary d and m would solve the problem of learning DNF. Observe that the class considered by Frazier et al. is a generalization of the class of DNF formulas in which all variables only appear negated.

While there has been some work addressing the general issue of mislabelled training examples in the PAC model [1, 18, 28, 17], there has been little research on learning geometric concepts with noise. Auer [3] investigates exact learning of boxes where some of the counterexamples, given in response to equivalence queries, are noisy. Auer shows that BOX_n^d is learnable using hypotheses from BOX_n^d 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 $1/(2d+1)$.

There has also been some work on learning discretized geometric concepts defined by non axis-parallel hyperplanes. Maass and Turán [26] study the problem of learning a single discretized halfspace using only equivalence queries. They give an efficient algorithm using $O(d^2(\log d + \log n))$ queries and give an information theoretic lower bound of $\binom{d}{2}$ on the number of queries when all hypotheses are discretized halfspaces. There has also been work on learning

non axis-parallel discretized rectangles with only equivalence queries. Maass and Turán [25] show an $\Omega(n)$ information theoretic lower bound on the number of equivalence queries when the hypotheses must be drawn from the concept class. Contrasting this lower bound, Bultman and Maass [9] give an efficient algorithm that uses membership and equivalence queries to efficiently learn this class using $O(\log n)$ equivalence queries.

4 Preliminaries

Let $\mathcal{N}$, $\mathcal{Z}$ and $\mathcal{R}$ be the set of nonnegative integers, integers and reals, respectively. We use $y_1, \dots, y_d$ to denote the d dimension variables. Let $\mathcal{N}_n = \{1, \dots, n\}$ and $S = \{a_1, \dots, a_s\} \subset \mathcal{Z}^d$ be a set of slopes. A d -dimensional *hyperplane* is $\sum_{j=1}^d a_{i,j} y_j = b$ for some $a_{i,j}, b \in \mathcal{Z}$ (when we write $\sum_{j=1}^d a_{i,j} y_j = b$ we mean $\{x \mid \sum_{j=1}^d a_{i,j} y_j = b\}$). A *halfspace* is $\sum_{j=1}^d a_{i,j} y_j \succ b$ where $\succ \in \{>, \geq, <, \leq\}$. A *discretized hyperplane* (respectively halfspace) is $H \cap \mathcal{N}_n^d$ for some hyperplane (respectively, halfspace) H . A *geometric concept generated from hyperplanes* with slopes from $S \subset \mathcal{Z}^d$ is a set $\hat{g} \subset \mathcal{R}^d$ whose boundaries are defined by hyperplanes with slopes from S . A *discretized geometric concept generated from hyperplanes* with slopes from $S \subset \mathcal{Z}^d$ is $g = \hat{g} \cap \mathcal{N}_n^d$.

The *complexity* $C_S(\hat{g})$ of a geometric concept $\hat{g}$ is the minimal number of hyperplanes with slopes from S such that its union contains the boundary of $\hat{g}$. In other words $C_S(\hat{g})$ is the minimum number of halfspaces from S that are needed to define the boundaries of $\hat{g}$. The complexity $C_S(g)$ of a discretized geometric concept g is the minimal $C_S(\hat{g})$ over all geometric concepts $\hat{g}$ that satisfies $g = \hat{g} \cap \mathcal{N}_n^d$. By a simple information theoretic argument it follows that any exact learning algorithm for a discretized geometric concept g cannot run in time less than the complexity $C_S(g)$. Also the complexity of learning one point in $\mathcal{N}_n^d$ is at least $\log n$. In our algorithms the input will be S . We use $\|S\|$ to denote the sum the logarithm of the absolute values of the integers in S and call this the size of S . Therefore we say that a learning algorithm runs in polynomial time if the time is $\text{poly}(\|S\|, C_S(g), \log n, d)$.

Observe that if we choose the slopes $S = \{e_i\}$ the standard basis then for any discretized geometric concept g defined by the union of m boxes in d dimensional space, $C_S(g) \leq 2md$ since at most $2d$ halfspaces are needed to define the boundaries of each box.

5 Learning Unions of Boxes with Membership and Equivalence Queries

In this section we present an algorithm that exactly identifies any concept from $\bigcup_m \text{BOX}_n^d$ (so $S = \{e_i\}$, the standard basis) while receiving at most md counterexamples and using time and membership queries that are polynomial in m and $\log n$ for d any constant. This section

serves two purposes: (1) the other algorithms presented build upon this basic algorithm and thus for ease of exposition we present it here, and (2) it uses very few equivalence queries which is of interest if one's goal is to minimize the number of prediction errors made by the learner [8]

The following definition will be used throughout this section

Definition 1 For a discretized geometric concept g , we define a $+/-$ pair to be a positive point $x_+ = (x_1, \dots, x_i, \dots, x_d)$ paired with a negative point x_- where $x_- = (x_1, \dots, x_i + 1, \dots, x_d)$ or $x_- = (x_1, \dots, x_i - 1, \dots, x_d)$ where we implicitly assume that any point outside of $\mathcal{N}_n^d$ is a negative point.

We define the *halfspace* associated with a given $+/-$ pair to be the unique, axis-aligned halfspace H that contains the positive but not the negative point. So for a $+/-$ pair where the positive point's i th coordinate is c , if the negative point's i th coordinate is $c + 1$ then H is given by $y_i \leq c$. Similarly, if the negative point's i th coordinate is $c - 1$ then H is given by $y_i \geq c$. We define the associated *hyperplane* to be the set of all points satisfying $y_i = c$.

For each of the d dimensions, we maintain a set $\mathcal{H}_i$ of hyperplanes discovered for that dimension that define the boundaries of g . We let $\mathcal{H} = \bigcup_{i=1}^d \mathcal{H}_i$. Observe that in dimension i we have decomposed $\mathcal{N}_n^d$ into up to $2|\mathcal{H}_i| + 1$ regions: $|\mathcal{H}_i|$ corresponding to the hyperplanes themselves, and $|\mathcal{H}_i| + 1$ corresponding to the “strips” obtained when $\mathcal{N}_n^d$ is cut by each of the $|\mathcal{H}_i|$ hyperplanes. Recall that $|\mathcal{H}_i| \leq 2m$ and thus our final hypothesis divides $\mathcal{N}_n^d$ into a *grid* $G_{\mathcal{H}}$ of

$$\prod_{i=1}^d (2|\mathcal{H}_i| + 1) \leq \prod_{i=1}^d (4m + 1) = (4m + 1)^d \quad (1)$$

regions (or connected components). We say that $G_{\mathcal{H}}$ is *consistent* if any two points with known classification in any region of $G_{\mathcal{H}}$ have the same classification. Given a consistent $G_{\mathcal{H}}$, a hypothesis is obtained by simply classifying all points according to the unique classification of all known points in that region (with negative used as a default).

We now demonstrate that we can represent such a hypothesis so that it is very efficient to evaluate. For each dimension i we maintain a balanced binary search tree T_i where each internal node corresponds to one of the hyperplanes (with x_j used for the key) in $\mathcal{H}_i$, and each leaf node corresponds to one of the strips created. For each leaf node v and dimension i we keep a pair $(\min_v^i, \max_v^i)$, where $\min_v^i$ (respectively, $\max_v^i$) holds the minimum (respectively, maximum) x_j such that $y_i = x_j$ is a point in the region corresponding to leaf v . (For the internal nodes, the key itself serves the role of both $\min_v^i$ and $\max_v^i$.)

Let $R_i = \{[\min_v^i, \max_v^i] \mid \min_v^i \text{ and } \max_v^i \text{ are the range for all nodes } v \text{ in the tree } T_i\}$ and let $\mathcal{T} = \{T_1, \dots, T_d\}$. Thus $G_{\mathcal{H}} = R_1 \times R_2 \times \dots \times R_d$. For $r \in G_{\mathcal{H}}$, corresponding to node v ,

Figure 1: This shows the final set of regions corresponding to the target concept outlined in bold. The classification of each two-dimensional region is 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 shall refer to the point $(\min_v^1, \dots, \min_v^d)$ as the *lower corner* of r and $(\max_v^1, \dots, \max_v^d)$ as the *upper corner* of r .

In addition to the trees $T_1, \dots, T_d$, our hypothesis also maintains a prediction array A with $|G_{\mathcal{H}}|$ entries where for $r \in G_{\mathcal{H}}$, $A[r]$ gives the classification for region r . For a consistent region (i.e. all known points have the same classification) A will contain either a 0 (for negative) or a 1 (for positive). However, for regions in which there is an inconsistency there will be a pointer into a queue Q of inconsistent regions. In addition for each region $r \in Q$ we store points x_+ and x_- giving a pair of inconsistent points in r . We use $h(G_{\mathcal{H}}, A)$ to denote the hypothesis defined by the regions in $G_{\mathcal{H}}$ with the classifications given in A . Figure 1 shows the set of regions defined by a target concept once all hyperplanes are discovered, and the classifications of all of the regions (as stored in A).

Given a hypothesis $h(G_{\mathcal{H}}, A)$ (we sometimes denote this simply as h) and a point x we can compute $h(x)$, the prediction made by hypothesis h on point x , as follows. For $1 \leq i \leq d$ 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 $r \in G_{\mathcal{H}}$ that contains x . Finally $h(x) = A[r]$.

5.1 The Membership and Equivalence Query Algorithm

Our algorithm works by repeatedly building a consistent hypothesis that incorporates all known halfspaces. Given an inconsistent hypothesis $h(G_{\mathcal{H}}, A)$ and the queue Q of inconsistent regions from $G_{\mathcal{H}}$ we refine $h(G_{\mathcal{H}}, A)$ so that it is consistent using the following procedure that

uses membership queries to find *new* hyperplanes with which to modify the hypothesis. We never remove any hyperplane from $\mathcal{H}$ and only search 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 inconsistent region of $G_{\mathcal{H}}$.

Our procedure to build a consistent hypothesis repeatedly does the following until Q is empty (and thus the hypothesis is consistent). Let r be the region at the front of the queue. Since r is not a consistent region we know that there must be some unknown hyperplane of g that goes between x_+ (a known positive point in r) and x_- (a known negative point in r). Thus we can perform a binary search between x_+ and x_- (where the comparisons are replaced by membership queries) to find a $+/-$ pair contained within region r using only $\lceil \log n \rceil$ membership queries and $O(\log n)$ time. Furthermore, the hyperplane defined by this $+/-$ pair is guaranteed to be a hyperplane that has not yet been discovered (by the definition of a region).

The full details of procedure `ADD-HYPERPLANE`, which modifies $h(G_{\mathcal{H}}, A)$ to incorporate the new hyperplane found, is given in Figure 2. The learner begins by using a standard tree insertion procedure to insert the new hyperplane into the search tree for the appropriate dimension. Then the set of regions that have been split by the hyperplane are deleted from $\mathcal{H}$ and each is replaced by three new regions (one of them being a degenerate region corresponding to the hyperplane itself). For each new region of r we make a membership query on the lower and/or upper opposing corners if those queries have not already been made. (As we shall discuss this step can be replaced by just using 0 as the default value for $A[r]$). If the classification of these two corners are the same then the classification is entered in $A[r]$, otherwise the region is placed in Q with these corners used for x_+ and x_- .

Our algorithm `LEARN-WITH-MQS` works as follows. For ease of exposition we artificially extend the instance space from $\mathcal{N}_n^d$ to $\{0, 1, \dots, n, n+1\}^d$ 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 pseudo code 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, $G_{\mathcal{H}}$ just contains the single region corresponding to the entire instance space. Since the upper and lower corners of this region are negative, the initial 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 consistent hypothesis. Using membership queries (in the form of a binary search) we can find two new hyperplanes of the target concept. Without loss of generality, we assume that x is a positive counterexample in region r of $G_{\mathcal{H}}$. Since the hypothesis was consistent and x is a positive counterexample, we know that the upper and lower corners of r are classified as negative. Thus we can use these

ADD-HYPERPLANE($h(G_{\mathcal{H}}, A), Q, i, c$)

Let v be the leaf of T_i for which $\min_v^i \leq c \leq \max_v^i$

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 $[\min_v^i, c - 1]$

v has a right child with range $[c + 1, \max_v^i]$

Let $R_{delete} = R_1 \times \dots \times R_{i-1} \times \{[\min_v^i, \max_v^i]\} \times R_{i+1} \times \dots \times R_d$

Let $R_{add} = R_1 \times \dots \times R_{i-1} \times \{[\min_v^i, c - 1], [c, c], [c + 1, \max_v^i]\} \times R_{i+1} \times \dots \times R_d$

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

Copy x_- and x_+ in the appropriate queue entries for $r_<$ and $r_>$

Remove q from Q

For each $r \in R_{add}$

Let x_u (resp., x_ℓ) be the upper (resp., lower) corner of r

Make a membership query to determine x_u and x_ℓ if not already known

If $g(x_u) = g(x_\ell)$ then let $A[r] = g(x_u)$

Else (r is inconsistent)

Assign x_u and x_ℓ to x_- and x_+ as appropriate

$Q.\text{ENQUEUE}(r)$

Figure 2: Our subroutine to update $h(G_{\mathcal{H}}, A)$ to incorporate the newly discovered hyperplane $y_i = c$. The new hyperplane is added to tree T_i . Then all regions in $G_{\mathcal{H}}$ that are split are removed from Q . Finally this procedure initializes the new entries of $G_{\mathcal{H}}$ in the prediction matrix A .

corners of r (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-CONSISTENT-HYPOTHESIS` to refine any inconsistent regions. Figure 3 gives the complete algorithm.

5.2 Analysis

We now analyze the time and query complexity of `LEARN-WITH-MQS`. As part of this analysis we use the following lemma.

Lemma 1 *Every counterexample can be used to discover at least two distinct new hyperplanes of the target concept.*

Proof: Let x be the counterexample and $r \in G_{\mathcal{H}}$ be the region containing x . Since $h(G_{\mathcal{H}}, A)$ is a consistent hypothesis, we know that the upper and lower corners of r are classified opposite x and all points in r are classified opposite x by the hypothesis. Since a positive point and a negative point must be separated by some hyperplane of the target concept, searches between x and each of upper and lower corners will find some $+/-$ pair. These will be distinct since the two searches move away from each other in all dimensions. $\square$

We now prove that our first algorithm has the stated complexity.

Theorem 1 *Given any $g \in \bigcup_m \text{BOX}_n^d$, `LEARN-WITH-MQS` achieves exact identification of g making at most $md + 1$ equivalence queries, and using $O((4m)^d + md \log n)$ time and membership queries.*

Proof: The correctness of `LEARN-WITH-MQS` is trivial. Since the algorithm only returns a hypothesis $h(G_{\mathcal{H}}, A)$ for which `Equiv( $h(G_{\mathcal{H}}, A)$ )` returns “yes”, the algorithm is correct upon returning a hypothesis.

We now analyze the query and time complexity of `LEARN-WITH-MQS`. Recall that since there are only m boxes in the target concept, there are at most $2md$ hyperplanes in the final hypothesis. Furthermore, since no hyperplane is ever rediscovered and every binary search (which uses $O(\log n)$ membership queries) discovers a hyperplane, we know that $O(md \log n)$ membership queries are used during all of the binary searches made by the algorithm. Also, as given in Equation (1), there are at most $(4m + 1)^d$ regions in the final hypothesis, and thus the number of membership queries used for querying the upper and lower opposing corners is at most $2 \cdot (4m + 1)^d = O((4m)^d)$. Since these are the only two places in which membership queries are performed, the total number of membership queries made by our algorithm is $O((4m)^d + md \log n)$.

From Lemma 1 we know that each counterexample enables `LEARN-WITH-MQS` to find at least two new, distinct hyperplanes of the target concept. Thus since there are at most

LEARN-WITH-MQS:

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 $G_{\mathcal{H}}$, let $A[r] = 0$

While $\text{Equiv}(h(G_{\mathcal{H}}, A)) \neq \text{"yes"}$

 Let x be the counterexample where x is in region r of $h(G_{\mathcal{H}}, A)$

 Let z_1 and z_2 be the lower and upper 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($h(G_{\mathcal{H}}, A), Q, i, c$)

$h(G_{\mathcal{H}}, A) \leftarrow \text{MAKE-CONSISTENT-HYPOTHESIS}(h(G_{\mathcal{H}}, A), Q)$

Return $h(G_{\mathcal{H}}, A)$

MAKE-CONSISTENT-HYPOTHESIS($h(G_{\mathcal{H}}, A), Q$)

While $Q \neq \emptyset$

$r \leftarrow Q.\text{DEQUEUE}$

 Perform binary search between x_- and x_+ from r

 to find the hyperplane $y_i = c$

 ADD-HYPERPLANE($h(G_{\mathcal{H}}, A), Q, i, c$)

Figure 3: Algorithm for learning unions of d -dimensional axis-parallel boxes using membership and equivalence queries.

$2md$ hyperplanes comprising the m boxes, at most md counterexamples can be received and thus at most $md + 1$ equivalence queries are made.

The time needed to evaluate $h(G_{\mathcal{H}}, A)(x)$ for an unlabeled example x is $O(d \log m)$ since the key operation is performing d searches in balanced search trees of depth $O(\log m)$. Thus, it is easily seen that the time complexity of this algorithm is $O((4m)^d + md \log n)$. $\square$

Finally, it is easily seen that this algorithm extends to learn any geometric concept with slopes from $S = \{e_i\}$ (the standard basis) while receiving at most $C_S(g)/2$ counterexamples, and using time and membership queries polynomial in $C_S(g)$ and $\log n$ for d any constant.

5.3 Using a Hypothesis Class of Unions of Boxes

We now describe how a consistent hypothesis can be converted to the union of $O(md \log m)$ boxes from BOX_n^d . Since all equivalence queries are made with consistent hypotheses, such a conversion enables our algorithm to learn the union of m boxes from BOX_n^d using as a hypothesis class the union of $O(md \log m)$ boxes from BOX_n^d .

Recall that a consistent 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 of the positive regions. We now describe how to formulate this problem as a set covering problem for which we can then use the standard greedy set covering heuristic [13] to perform the conversion. The set X of objects to cover will simply contain all positive regions in h . Thus $|X| \leq (4m + 1)^d$. 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 $b \in B$, if b contains any negative region, then throw it out. Otherwise, place in $\mathcal{F}$ the set of regions contained within b . Thus $|\mathcal{F}| \leq (2m)^{2d}$ since there are at most $2m$ values in each dimension that can form the two sides of the box. Furthermore, $\mathcal{F}$ contains a subset of size m that covers all items in X . Finally, we can apply the greedy set covering heuristic to find a set of at most $m(\ln |X| + 1) = m(d \ln(4m + 1) + 1) = O(md \log m)$ boxes that cover all positive regions. The time to perform the conversion is $O((2m)^{2d})$. Thus, since at most $md + 1$ equivalence queries are made, the total time spent in converting the internal hypotheses into hypotheses that are unions of boxes is at most $O(md \cdot (2m)^{2d})$.

6 Learning Unions of Boxes With Only Equivalence Queries

We now describe a simple method to remove the use of membership queries in LEARN-WITH-MQS. First observe that the use of membership queries in this algorithm can easily be reduced to only their use within the binary searches. Instead of querying opposing corners of new regions created, we can instead use the classification of the single point known or

otherwise a default of negative for the classification of the region. Then the counterexamples from the equivalence queries can be used to obtain a positive and negative point in the region that can be used for the binary search. (Of course, this modification dramatically increases the number of equivalence queries used.)

Now to remove the use of membership queries in a binary search between x_+ and x_- we simply take the midpoint between x_+ and x_- (i.e. the first point on which a membership query would be made) and insert a hyperplane going through that point for each of the d dimensions. Note that at most $2d \log n$ hyperplanes will be added in each dimension for each hyperplane of the target and thus the number of regions in the final hypothesis will be at most $(8dm \log n + 1)^d$. It is easily seen that for d constant this algorithm exactly learns $\bigcup_m \text{BOX}_n^d$ using only equivalence queries with both time and equivalence queries polynomial in m and $\log n$.

7 Extending S to Arbitrary Known Slopes

We now present a modification of the equivalence query algorithm described in Section 6 that handles the situation in which S can be an arbitrary set of known slopes versus just being the standard basis. We let s denote the number of distinct slopes in S (i.e. $s = |S|$).

Let $S = \{a_1, \dots, a_s\} \subset \mathcal{Z}^d$ be a set of slopes and $B = \{B_1, \dots, B_s\}$ where $B_i \subset \mathcal{R}$. Together S and B define a set of hyperplanes

$$\mathcal{H} = \{a_i y^T = b_{i,j} \mid y = (y_1, \dots, y_d), i = 1, \dots, s, b_{i,j} \in B_i\}.$$

The hypothesis $h = h(G_{\mathcal{H}}, A)$ is that which classifies each $x \in \mathcal{N}_n^d$ according to the unique classification of all known points (if any) in the region of $G_{\mathcal{H}}$ that contains x and if there are no known points in the region containing x then negative is used as the default.

As in our basic equivalence query algorithm, this algorithm begins with the entire region classified as negative. After the first two equivalence queries are made (the first with $h = \emptyset$ and the second with $h = \mathcal{N}_n^d$) the algorithm will have positive counterexample x and a negative counterexample u . Thus the straight line between x and u must intersect one of the hyperplanes that define g . Let $v = (x + u)/2$ be the midpoint of the line between x and u . Without the ability to make membership queries it is not possible to find a $+/-$ pair. Furthermore, even if we could find a $+/-$ pair we would not be able to determine the slope of the hyperplane that created that $+/-$ pair. As in the previous section, we address this problem by adding to our set of hyperplanes $\mathcal{H}$ a hyperplane passing through the midpoint v for *each* slope in S . We repeatedly use this process until an equivalence query made with $h(G_{\mathcal{H}}, A)$ is correct.

For ease of exposition, in this section we will not discuss the details of how to represent $\mathcal{H}$ so that $h(G_{\mathcal{H}}, A)$ can be efficiently evaluated (in terms of all parameters). However, the

```

LEARN-GENERAL-SLOPES( $S = \{a_1, \dots, a_s\}$ )

 $\mathcal{H} \leftarrow \emptyset$ 
Let  $r$  be the single region of  $G_{\mathcal{H}}$ , let  $A[r] = 0$ 
While  $\text{Equiv}(h(G_{\mathcal{H}}, A)) \neq \text{"yes"}$ 
    Let  $x$  be the counterexample where  $x$  is in region  $r$  of  $h(G_{\mathcal{H}}, A)$ 
     $A[r] = 1 - h(G_{\mathcal{H}}, A)(x)$ 
    If there is a known point  $u$  in  $r$  classified opposite  $x$ 
         $v = (x + u)/2$ 
        For  $i = 1$  to  $s$ 
             $\mathcal{H} \leftarrow \mathcal{H} \cup a_i y^T = a_i v^T$ 
        Update  $A$  to incorporate the new regions created
Return  $h(G_{\mathcal{H}}, A)$ 

```

Figure 4: Algorithm for exactly learning a discretized geometric concept defined by slopes in S using only equivalence queries.

technique of Section 5 of using s balanced search tree, one for each element of S generalizes in the obvious manner.

Our algorithm, at a high level, is shown in Figure 4. In this algorithm, LEARN-GENERAL-SLOPES, S is the set of the possible slopes of the hyperplanes. We initialize $\mathcal{H}$ to be the empty set and the classification of the single region to be 0. (And thus the initial hypothesis, $h(G_{\mathcal{H}}, A)$, is simply the always false hypothesis.) We ask the equivalence query $h(G_{\mathcal{H}}, A)$ and use the counterexample x to update A . If this counterexample is the first counterexample in its region then we just update A . Otherwise, if this counterexample is in some region for which we have already seen a point u , then x and u have different classifications in g and the line that passes through x and u must intersect a defining hyperplane of g . We then define $v = (x + u)/2$ and add all possible hyperplanes that pass through v to the set $\mathcal{H}$. We then repeat this process until $h(G_{\mathcal{H}}, A)$ is logically equivalent to g .

7.1 Analysis

We now prove the correctness of our algorithm and analyze its complexity. We use the following lemma to bound the maximum number of regions that will be contained in $G_{\mathcal{H}}$.

Lemma 2 *Any t , d -dimensional hyperplanes in a $(d + 1)$ -dimensional space divide the space into at most $t^d + 1$ regions.*

Proof: The proof is by induction on d . For $d = 0$, t points in a line divide the line into $t + 1$ regions. For any $d \geq 0$ let $H_1, \dots, H_t$ be d -dimensional hyperplanes. If $H_1, \dots, H_t$ are parallel then they divide a $(d + 1)$ -dimensional space into $t + 1 \leq t^d + 1$ regions.

Suppose $H_1, \dots, H_t$ are not parallel. Then each region in the $(d + 1)$ -dimensional space is bounded by at least two hyperplanes in $H_1, \dots, H_t$. For each H_i let $R_i = \{H_i \cap H_j \mid H_i \cap H_j \neq \emptyset, j \neq i\}$ be $(d - 1)$ -dimensional hyperplanes in H_i . Since $|R_i| \leq t - 1$, by the induction hypothesis the hyperplanes in R_i intersect H_i in at most $(t - 1)^{d-1} + 1$ regions. Each such region corresponds to at most two regions generated by $H_1, \dots, H_t$ in the $(d + 1)$ -dimensional space (one from each side). However, since each region generated by $H_1, \dots, H_t$ is bounded by at least two hyperplanes, the number of regions is at most the number of regions in $\cup_{i=1}^t R_i$ which is at most

$$((t - 1)^{d-1} + 1)t \leq t^d + 1.$$

□

We are now ready to analyze our algorithm LEARN-GENERAL-SLOPES

Theorem 2 *Let S be a set of slopes. Then LEARN-GENERAL-SLOPES exactly learns any target concept g from the class of discretized geometric concepts generated from hyperplanes with slopes from S using time and equivalence queries polynomial in $\|S\|, C_S(g)$, and $\log n$ for d any constant.*

Proof: The correctness follows trivially since the algorithm only returns a hypothesis $h(G_{\mathcal{H}}, A)$ for which $\text{Equiv}(h(G_{\mathcal{H}}, A))$ returns “yes”, the algorithm is correct upon returning a hypothesis.

We now analyze the query and time complexity. Let $m = C_s(g)$. Recall that $\|S\|$ denotes the sum of the logarithms of the absolute values of the integers in S . Thus, by the definition of $\|S\|$, any $a_i \in S$ can be represented using at most $\|S\|$ bits. Thus there are at most $2^{\|S\|}$ possible values for a slope. Also, $y \in \mathcal{N}_n^d = \{1, \dots, n\}$ and thus has $n \cdot d$ possible values. Let $H_1, \dots, H_m$ be a minimal set of hyperplanes with slopes from S that generate the boundary of the geometric concept g . Since each hyperplane defining g is of the form $b = a_i y^T$ we have that the maximum number of values that any b can have is

$$\|b_i\| \leq \|a_i y^T\| \leq 2^{\|S\|} \cdot n \cdot d = \gamma.$$

At any time during execution of our algorithm, for each of $H_1, \dots, H_m$, of those hyperplanes in $\mathcal{H}$ there are two closest hyperplanes (one for each side) with the given slope. We now show that for every counterexample that causes $\mathcal{H}$ to be modified there exists some H_i ($i = 1, \dots, m$) for which the distance between it and one of its closest hyperplanes is reduced by at least a factor of 2.

Suppose x is a counterexample to the current hypothesis $h(G_{\mathcal{H}}, A)$ and that the region r of $G_{\mathcal{H}}$ that contains x already contains the point u (otherwise $\mathcal{H}$ is not modified). By the definition of $h(G_{\mathcal{H}}, A)$ it follows that $g(u) \neq g(x)$. Therefore the line segment between x and u must intersect some hyperplane, say $H_* \equiv (a_i y^T = b_*)$, of g for $a_i \in S$ and $b_* \in \mathcal{Z}$. Let $b_1 = a_i y^T$ and $b_2 = a_i y^T$ be the two nearest hyperplanes to H_* with slope a_i . Let $v = (x + u)/2$ be the midpoint between x and u . Without loss of generality we assume that the hyperplane $b'_1 = a_i y^T$ that passes through v with slope a_i is between $b_1 = a_i y^T$ and $b_* = a_i y^T$ as illustrated in Figure 5. We denote the hyperplane that passes through x (respectively u) with slope a_i by $b_x = a_i y^T$ (respectively, $b_u = a_i y^T$). Thus we have that $b_1 \leq b_x \leq b'_1 \leq b_* \leq b_u \leq b_2$.

Figure 5: Here x is a counterexample to the current hypothesis and $b_1 = a_i y^T$ and $b_2 = a_i y^T$ are the nearest hyperplanes. The hyperplane $b'_1 = a_i y^T$ is added to $\mathcal{H}$ by LEARN-GENERAL-SLOPES.

Let $\Delta = b_* - b_1$. (Thus Δ is proportional to the distance between H_* and $b_1 = a_i y^T$.) Observe that LEARN-GENERAL-SLOPES will add the hyperplane $b'_1 = a_i y^T$ to $\mathcal{H}$ and this will now replace $b_1 = a_i y^T$ as a closest hyperplane of slope a_i to H_* . Let $\Delta' = b_* - b'_1$. (Thus Δ' is proportional to the new distance between H_* and its closest hyperplane in $\mathcal{H}$ in that direction.) We now show that $\Delta' \leq \Delta/2$. Observe that

$$\begin{aligned} \Delta' &= b_* - b'_1 \\ &= b_* - \left(\frac{(b_u + b_x)}{2} \right) \\ &\leq b_* - \left(\frac{(b_* + b_1)}{2} \right) \end{aligned}$$

$$= \frac{b_* - b_1}{2} = \frac{\Delta}{2}$$

Thus, the distance between H_* and one of its two nearest hyperplanes is reduced by a factor of two. Finally, when the distance between H_* and both of its two nearest hyperplanes is less than 1 then the algorithm has determined the discretized hyperplane H_* and no other hyperplanes will be added for H_* .

Since the distance between each of $H_1, \dots, H_m$ with both of its closest hyperplanes in $\mathcal{H}$ is at most γ it follows that the number of counterexamples needed to find one hyperplane is

$$\lceil 2 \log(\gamma) \rceil = \lceil 2 \log(2^{\|S\|} \cdot n \cdot d) \rceil = O(\|S\| + \log dn).$$

The number of hyperplanes is $m = C_S(g)$ and at each iteration we add $s = |S| \leq \|S\|$ hyperplanes to the hypothesis. Therefore the number of hyperplanes generated by our algorithm is $O(ms(\|S\| + \log dn))$. Thus by Lemma 2 the number of regions of the hypothesis is

$$O(ms(\|S\| + \log dn))^d.$$

The number of iterations that do not add any hyperplanes is bounded by the number of regions, and thus the number of equivalence queries made by LEARN-GENERAL-SLOPES is

$$O(ms(\|S\| + \log dn))^d$$

and clearly the time complexity is also polynomial in $m, \|S\|$, and $\log n$ for any constant d as desired. $\square$

8 Handling Lies in the Counterexamples

In this section we consider the case in which the learner is provided with an l -liar teacher that can lie on up to l of the counterexamples. To learn the true classification of the points about which the environment has lied, it is necessary for the learner to isolate these instances. That is, the learner must create a region that consists of the single point that was the counterexample.

A degenerate region (a region consisting of fewer than d dimensions) is created whenever our algorithm adds to the hypothesis a hyperplane, passing through a given point, for each slope in S . We want to isolate the counterexample on which the lie occurred in a region of dimension 0. Notice, however, that given some set of slopes, it is possible that the hyperplanes defined by the slopes, passing through the counterexample, do not create a 0-dimensional region.

Therefore, to ensure that the 0-dimensional region is created, we include in S the slopes of the elementary vectors $\{e_i\}$. Then the faulty points will be bounded by hyperplanes and at

Figure 6: These concepts illustrate why number of boxes is an inadequate measure of complexity. (a) is more complex than (b) because of the large number of intersections of the sides of the boxes.

the end the oracle must give their real values. This changes $C_S(g)$ to $C_S(g) + dl$, the number of slopes to $|S| + d$ and the complexity to

$$O((C_S(g) + dl)(|S| + d)(\|S\| + d + \log dn))^d.$$

9 A Return to Learning Unions of Boxes

Observe that by extending the hyperplane defined by a $+/-$ pair across the entire domain, our initial algorithm LEARN-WITH-MQS may unnecessarily split a consistent region into a large number of smaller regions all of which make the same prediction. The algorithm we present here is motivated by the goal of reducing this unnecessary splitting by only splitting the region in which the counterexample is contained. We show that this algorithm runs in polynomial time for *either* m or d constant.

We begin by examining how one might measure the complexity of a concept from $\bigcup_m \text{BOX}_n^d$. Observe that the number of boxes m used to form the target concept is not a good measure of the complexity of the target concept. For example, consider the two examples shown Figure 6. While both targets are composed of 6 boxes, the first is clearly more complex than the second. Thus the complexity of an algorithm should depend on some quantity other than just the number of boxes and dimension of the target concept. We now introduce such a new complexity measure, σ , to better capture the complexity of the target concept. We define a *segment* of the target concept g as a maximum portion of one of the sides of g that lies either entirely inside or entirely outside of each of the other halfspaces defining the target. For example the target concept shown in Figure 1 has 20 segments. For a concept $g \in \bigcup_m \text{BOX}_n^d$, we let σ denote the number of segments in the target concept corresponding to g . Observe that $m \leq \sigma \leq (2m)^d$, where the first inequality is based on the assumption that none of the boxes are contained within the union of the other $m - 1$ boxes.

The hypothesis class we use in this algorithm is a decision tree over the halfspaces defining the target concept. Namely, each hypothesis T is a rooted binary tree where each internal node is labeled with a halfspace and whose leaves are labeled from $\{0, 1\}$. We evaluate T recursively by starting at the root and evaluating the left subtree if the root's halfspace does not contain the point, and right subtree otherwise. When a leaf is reached its label is output. Observe that each node of T corresponds to a sub-region of the domain, with the root corresponding to the entire domain. The halfspace H associated with each internal node divides its region r into two sub-regions, with the left child being the sub-region given by $\overline{H} \cap r$, and the right child being the sub-region given by $H \cap r$. The leaves correspond to a set of nonoverlapping boxes that cover the entire region where the label for a given region is given by the label for the corresponding leaf. Observation 1 shows that the height of the final decision tree will be at most $2md$. Thus the hypothesis can be evaluated in time polynomial in both m and d .

Observation 1 *The height of the final decision tree constructed by ALT-LEARN-WITH-MQS is at most $2md$ since each of the at most $2md$ halfspaces defining the target polyhedron can appear at most once on any path from the root to any leaf.*

We now describe our algorithm that has complexity polynomial in σ and $\log n$ as well as being an efficient algorithm for either m or d constant. We initialize T to be a single 0 leaf node. (Again we implicitly use the instance space $\{0, 1, \dots, n, n+1\}^d$.) When a counterexample is received, we first search T to find the leaf v containing it. Let r be the sub-region corresponding to v . Then as in LEARN-WITH-MQS we use a binary search to find a $+/-$ pair contained in r that defines a halfspace H . We replace v with an internal node labeled with H , having left child v_L corresponding to the region given by $\overline{H} \cap r$ and right child v_R corresponding to the region given by $H \cap r$. At this point we call a procedure that recursively visits all newly created leaves in a depth-first manner and checks if the corresponding region is a consistent region. If the region r' associated with leaf v' is consistent then the classification field is filled, otherwise we use a binary search to obtain a halfspace H' for r' and replace v' by an internal node labeled with H' . We generate two new leaves: v'_L corresponding to the region $\overline{H'} \cap r'$ and v'_R corresponding to the region $H' \cap r'$. Then a recursive call is made to validate (if necessary) each of these new regions. The algorithm is shown in Figure 7. One possible final hypothesis that could be constructed by this algorithm, for the target concept shown in Figure 1, is shown in Figure 8. Figure 9 shows the decomposition of $\mathcal{N}_n^d$ that corresponds to the decision tree shown in Figure 8.

ALT-LEARN-WITH-MQS

Initialize T to be the single 0-leaf

While $\text{Equiv}(T) \neq \text{"yes"}$

 Let x be the counterexample

 Search in T to find the leaf v corresponding to region containing x

$T \leftarrow \text{SPLIT-REGION}(T, v, x)$

Return T

$\text{SPLIT-REGION}(T, v, x)$:

 Perform binary search between x and corner of region r corresponding to v

 Let H be the hyperplane found

 Let v_L , and v_R correspond to the new regions created

 Make v an internal node labeled with H and having left child v_L , and right child v_R

 Let r_L be the region $\overline{H} \cap r$ corresponding to v_L

 Let r_R be the region $H \cap r$ corresponding to v_R

 For each $r' \in \{r_L, r_R\}$

 Let v' be the leaf corresponding to region r'

 Perform a membership query on the upper and lower opposing corners of r'

 If both corners have classification $b \in \{0, 1\}$ let v' be a b -leaf of T

 Let $v'.prediction \leftarrow v.prediction$

 Else

$\text{SPLIT-REGION}(h_T, upper, lower)$

Figure 7: Alternate algorithm for learning unions of d -dimensional axis-parallel boxes that runs in polynomial time for *either* m or d constant.

9.1 Analysis

We now give two separate techniques for analyzing this algorithm. The first method of analysis gives that this algorithm uses queries and time polynomial in σ and $\log n$ (and thus polynomial in m and $\log n$ for d constant). The second method of analysis shows that our algorithm uses queries and time polynomial in d and $\log n$ for m any constant.

Theorem 3 *Given any target concept $g \in \bigcup_m \text{BOX}_n^d$, the algorithm ALT-LEARN-WITH-MQS achieves exact identification of g making at most $(\sigma/2 + 1)$ equivalence queries, and using $O(\sigma \log n)$ time and membership queries.*

Proof: Observe that each segment of g causes at most one region to be split. Thus the number of leaves in the decision tree created will be at most $\sigma + 1$.

By Lemma 1 we get that two segments are found from the counterexample to each equivalence query (here the second halfspace is implicitly found by the call to SPLIT-REGION). Thus at most $\frac{\sigma}{2} + 1$ equivalence queries will be made.

Furthermore, since there are at most 2 membership queries made to query the upper and lower corners of each leaf, and $\log n$ membership queries used in the binary searches for the σ halfspaces, it follows that the number of membership queries made is at most $2\sigma + \sigma \lceil \log n \rceil = O(\sigma \log n)$.

Observe that the depth of T is at most $2md$ since any of the $2md$ hyperplanes defined by the s boxes in the target concept will appear at most once on any path from a root to the leaf. Thus the time to locate the region to split is $O(md)$ and it immediately follows that the time complexity is $O(\sigma \log n)$. $\square$

Corollary 4 *The algorithm ALT-LEARN-WITH-MQS achieves exact identification for any $g \in \bigcup_m \text{BOX}_n^d$ using time and queries polynomial in m and $\log n$ for d constant.*

Proof: This follows immediately from Theorem 3 and the observation that $\sigma \leq (2m)^d$. $\square$

Finally, observe that just as we described in Section 5.3 our final hypothesis can be converted to the union of $O(md \log m)$ boxes from BOX_n^d . Recall that the time to perform the conversion is $O((2m)^{2d})$ and thus this will be efficient only if d is constant. Also, using the technique of Section 6 we can refine this algorithm to use only equivalence queries.

We now use a different method of analysis to show that ALT-LEARN-WITH-MQS uses time and queries polynomial in d and $\log n$ for m constant.

We begin by examining the number of regions created by this algorithm. Each region is represented by a single leaf in the hypothesis decision tree. Thus, we can find the number of regions by finding the number of leaves in our hypothesis. We now derive a recurrence relation for the number of leaves in the final decision tree. Let $g \in \bigcup_m \text{BOX}_n^d$ be the target concept and T be the final decision tree output by our algorithm. For each internal node T

there is an associated region of $\mathcal{N}_n^d$. For any node r in T , let h_r denote the height of the subtree of T rooted at r and let m_r be the minimum number of boxes needed to cover the region of $\mathcal{N}_n^d$ associated with r . So for the region r corresponding to the root of T we have that $m_r \leq m$ and $h_r \leq 2md$ since, by Observation 1, the height of T is at most $2md$.

Let $L(m, h)$ denote the maximum number of leaves in a decision tree rooted at a node r with $m_r = m$ and $h_r = h$. Then we have that $L(m, h) = L(m, h-1) + L(m-1, h-1)$ where for all $m \geq 0$, $L(m, 0) = 1$ and for all $h \geq 1$, $L(0, h) = 1$. To see this, observe that when we find, while building the hypothesis, a hyperplane that splits a node, the two subproblems that correspond to the left and right children both must have at most $h-1$ hyperplanes left to find since in the worst case, all other hyperplanes are split by this one and thus appear on both sides. Finally, since the hyperplane just found must be the side of one of the m boxes, that box will not appear in one of the recursive calls. (In the worst case all other boxes will be split).

Notice that there are no leaves in the decision tree on levels 0 to $m-1$. On each level from m to h there are leaves caused by the base case $m=0$ of the recurrence. The number of these leaves at level j is given by

$$\sum_{j=s}^h \binom{j-1}{m-1}$$

since the number of nodes, in the recursion tree, with $m=0$ at level j is equal to the number of nodes with $m=1$ at level $j-1$. There are also leaf nodes caused by the other base case of the recurrence, $h=0$. Note that this is the last level of the tree. The number of leaves here is given by

$$\sum_{j=0}^{s-1} \binom{h}{j}$$

since this gives the number of nodes for each non-zero value of m (the $m=0$ nodes on this level were already counted in the previous expression). Thus, the total number of leaves is:

$$\begin{aligned} \sum_{j=s}^h \binom{j-1}{m-1} + \sum_{j=0}^{m-1} \binom{h}{j} &= \sum_{j=m-1}^{h-1} \binom{j}{m-1} + \sum_{j=0}^{m-1} \binom{h}{j} \\ &= \sum_{j=0}^{h-1} \binom{j}{m-1} + \sum_{j=0}^{m-1} \binom{h}{j} \\ &= \binom{h}{m} + \sum_{j=0}^{m-1} \binom{h}{j} \\ &= \sum_{j=0}^m \binom{h}{j} \end{aligned} \tag{2}$$

Recall that $h \leq 2md$. In the following Lemma we show that $(2md)^m$ is an upper bound on the summation in Equation (2).

Lemma 3 *The number of leaves in the hypothesis decision tree constructed by ALT-LEARN-WITH-MQS is bounded above by $(2md)^m$ for $m > 1$ and is $2d + 1$ for $m = 1$.*

Proof: Let $n = md$ and $k = m$ for $m > 1$. Then the expression we have derived for the number of leaves is $\sum_{j=0}^k \binom{2n}{j}$. It is easily shown by induction on k that $\sum_{j=0}^k \binom{2n}{j} \leq (2n)^k$ for $n \geq k > 1$ and thus the result follows for $m > 1$. Finally, for $m = 1$ the number of leaves in the hypothesis is $2d + 1$. $\square$

We are now ready to prove the running time of our algorithm using this method of analysis. For ease of exposition we assume $m > 1$ in the remainder of this paper.

Theorem 5 *Given any target concept $g \in \bigcup_m \text{BOX}_n^d$, the algorithm ALT-LEARN-WITH-MQS achieves exact identification of g making at most $(2md)^m$ equivalence queries, making $O((2md)^m \cdot \log n)$ membership queries, and using $O((2md)^{m+1} \cdot \log n)$ time.*

Proof: Observe that the number of counterexamples received by ALT-LEARN-WITH-MQS is at most the number of internal nodes in our final decision tree. Thus the number of equivalence queries made by ALT-LEARN-WITH-MQS is at most the number of leaves in the final decision tree.

Equation (2) shows that $L(m, h) = \sum_{j=0}^m \binom{h}{j}$ and Lemma 3 proves a bound for the number of leaves of T of at most $\sum_{j=0}^m \binom{2md}{j} \leq (2md)^m$. Thus it immediately follows that at most $(2md)^m$ equivalence queries are made. Since at most $\lceil \lg n \rceil$ membership queries are used by the binary search procedure when splitting a node, it follows that the number of membership queries made by ALT-LEARN-WITH-MQS is $O((2md)^m \cdot \log n)$. Finally, since it takes $O(md)$ time to find the node corresponding to the region containing the counterexample and at most $O(\log n)$ time for each binary search, it follows that the time complexity of ALT-LEARN-WITH-MQS is $O((2md)^{m+1} \cdot \log n)$. $\square$

Thus ALT-LEARN-WITH-MQS achieves exact identification for any $g \in \bigcup_m \text{BOX}_n^d$ using time and queries polynomial in d and $\log n$ for m constant. We note that for $m \geq 6$ we can remove a factor of 2^m in the complexity by using the tighter upperbound that $\sum_{j=0}^m \binom{2md}{j} \leq (md)^m$. Finally, it is easily seen that if this algorithm is run in parallel with $(2md)^m$ processors then the number of parallel steps needed is at most $2md$.

10 Concluding Remarks

We have given an efficient algorithm that uses membership and equivalence queries to exactly identify any concept from $\bigcup_m \text{BOX}_n^d$ for d constant. Furthermore, this algorithm makes at most $md + 1$ equivalence queries all of which can be formulated as the union of $O(md \log m)$ axis-parallel boxes.

We have also shown how to extend our basic algorithm to efficiently learn, from *only* equivalence queries, any discretized geometric concept generated from any number of half-spaces with any number of known (to the learner) slopes in a constant dimensional space. In particular, our algorithm exactly learns (from equivalence queries *only*) unions of discretized axis-parallel boxes in constant dimensional space in polynomial time. Further, this algorithm can be modified to handle a polynomial number of lies in the counterexamples provided by the environment.

Finally, we have introduced a new complexity measure, σ , that better captures the complexity of the union of m boxes than simply the number of boxes and the dimension. We presented an algorithm that uses time and queries polynomial in σ and $\log n$. In fact, the time and queries (both membership and equivalence) used by this single algorithm are polynomial for *either* m or d constant.

A number of important open questions, that we have not answered, concern the necessity of membership queries to exactly learn the class $\bigcup_m \text{BOX}_n^d$ (or the more general class of a discretized geometric concept) in time polynomial in d and $\log n$ when the number of boxes (or defining hyperplanes) is constant.

While we have provided an algorithm to efficiently learn geometric concepts defined by hyperplanes that are *not* axis parallel, to achieve this goal it was necessary that the learner is given *a priori* knowledge as to the slopes of the hyperplanes. An interesting direction is to explore the learnability (even for fixed dimensions) of geometric concepts defined by hyperplanes whose exact slopes are not known to the learner.

Another interesting direction is to explore other complexity measures, besides σ , that better capture the complexity of the target polyhedron. In particular, we feel that the number of sides of the target polyhedron is a good measure of the complexity of this class. For example the target concept shown in Figure 1 has 14 sides.

Finally, it would be interesting to see if $\bigcup_m \text{BOX}_n^d$ can be efficiently learned in time polynomial in m and $\log n$ for $d = O(\log m)$ or in time polynomial in d and $\log n$ for $m = O(\log d)$ (i.e. a generalization of the Blum and Rudich [5] result that $O(\log n)$ -term DNF formulas are exactly learnable). Of course, since $\bigcup_m \text{BOX}_n^d$ generalizes the class of DNF formulas, it seems very unlikely that one could develop an algorithm for the unrestricted case of $\bigcup_m \text{BOX}_n^d$ that is polynomial in m , $\log n$, and d . It may be possible, however, to obtain a truly polynomial algorithm for some subclass of $\bigcup_m \text{BOX}_n^d$.

Acknowledgements

We thank Peter Auer and the COLT committee for their comments.

References

- [1] Dana Angluin and Philip Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.
- [2] Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.
- [3] 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.
- [4] Avrim Blum. Separating PAC and mistake-bound learning models over the Boolean domain. In *31st Annual Symposium on Foundations of Computer Science*, October 1990.
- [5] 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.
- [6] 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.
- [7] Nader H. Bshouty. On learning discretized geometric concepts. Unpublished manuscript, May 1994.
- [8] Nader H. Bshouty, Sally A. Goldman, Thomas R. Hancock, and Sleiman Matar. Asking questions to minimize errors. In *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory*, pages 41–50, July 1993.
- [9] William J. Bultman and Wolfgang Maass. Fast identification of geometric objects with membership queries. In *Fourth Workshop on Computational Learning Theory*, pages 337–353, 1991.
- [10] 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.
- [11] Zhixiang Chen and Steven Homer. The bounded injury priority method and the learnability of unions of rectangles. Unpublished manuscript, May 1994.
- [12] 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.

- [13] V. Chvatal. A greedy heuristic for the set covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.
- [14] Mike Frazier, Sally Goldman, Nina Mishra, and Leonard Pitt. Learning from a consistently ignorant teacher. In *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory*, July 1994.
- [15] Steven Homer and Zhixiang Chen. Fast learning unions of rectangles with queries. Unpublished manuscript, July 1993.
- [16] Paul W. Goldberg, Sally A. Goldman, and H. David Mathias. Learning unions of boxes with membership and equivalence queries. In *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory*, July 1994.
- [17] Michael Kearns and Ming Li. Learning in the presence of malicious errors. *SIAM Journal on Computing*, 22(4):807–837, August 1993.
- [18] Philip D. Laird. *Learning from Good and Bad Data*. Kluwer international series in engineering and computer science. Kluwer Academic Publishers, Boston, 1988.
- [19] Nick Littlestone. Learning when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.
- [20] Philip M. Long and Manfred K. Warmuth. Composite geometric concepts and polynomial predictability. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 273–287. Morgan Kaufmann, August 1990.
- [21] 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.
- [22] 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.
- [23] 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.
- [24] Wolfgang Maass and György Turán. Lower bound methods and separation results for on-line learning models. *Machine Learning*, 9:107–145, 1992.
- [25] Wolfgang Maass and György Turán. Algorithms and lower bounds for on-line learning of geometrical concepts. *Machine Learning*, 14:251–269, 1994.

- [26] Wolfgang Maass and György Turán. How fast can a threshold gate learn? *Computational Learning Theory and Natural Learning Systems: Constraints and Prospects*, G. Drastal, S.J. Hanson and R. Rivest eds., *MIT Press*, to appear.
- [27] Wolfgang Maass and Manfred K. Warmuth. Efficient learning with virtual threshold gates. Personal communication.
- [28] Robert H. Sloan. Types of noise in data for concept learning. In *Proceedings of the 1988 Workshop on Computational Learning Theory*, pages 91–96. Morgan Kaufmann, 1988.
- [29] Leslie Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.