Report Number: WUCS-97-18

1997-01-01

Noise-Tolerant Parallel Learning of Geometric Concepts

Nader H. Bshouty, Sally A. Goldman, and H. David Mathias

We present several efficient parallel algorithms for PAC-learning geometric concepts in a constant-dimensional space. The algorithms are robust even against malicious classification noise of any rate less than 1/2. We first give an efficient noise-tolerant parallel algorithm to PAC-learn the class of geometric concepts defined by a polynomial number of (d-1)-dimensional hyperplanes against an arbitrary distribution where each hyperplane has a slope from a set of known slopes. We then describe how boosting techniques can be used so that our algorithms' dependence on {GREEK LETTER} and {DELTA} does not depend on d. Next we give an efficient noise-tolerant parallel... **Read complete abstract on page 2.**

Noise-Tolerant Parallel Learning of Geometric Concepts

Nader H. Bshouty, Sally A. Goldman, and H. David Mathias

Complete Abstract:

We present several efficient parallel algorithms for PAC-learning geometric concepts in a constant-dimensional space. The algorithms are robust even against malicious classification noise of any rate less than 1/2. We first give an efficient noise-tolerant parallel algorithm to PAC-learn the class of geometric concepts defined by a polynomial number of (d-1)-dimensional hyperplanes against an arbitrary distribution where each hyperplane has a slope from a set of known slopes. We then describe how boosting techniques can be used so that our algorithms' dependence on {GREEK LETTER} and {DELTA} does not depend on d. Next we give an efficient noise-tolerant parallel algorithm to PAC-learn the class of geometric concepts defined by a polynomial number of (d-1)-dimensional hyperplanes (of unrestricted slopes) against a uniform distribution. We then show how to extend our algorithm to learn this class against any (unknown) product distribution. Next we defined a complexity measure of any set S of (d-1)-dimensional surfaces that we call the variant of S and prove that the class of geometric concepts defined by surfaces of polynomial variant can be efficienty learned in parallel under a product distribution (even under malicious classification noise). Furthermore, we show that the VC-dimension of the class of geometric concepts defined by a set of surfaces S of variant v is at least v. Finally, we give an efficient, parallel, noise-tolerant algorithm to PAC-learn any geometric concept defined by a set S of (d-1)-dimensional surfaces of polynomial area under a uniform distribution.

Noise-Tolerant Parallel Learning of Geometric Concepts

Nader H. Bshouty, Sally A. Goldman and H. David Mathias

WUCS-97-18

April 1997

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

Noise-Tolerant Parallel Learning of Geometric Concepts

NADER H. BSHOUTY bshouty@cpsc.ucalgary.ca
Dept. of Computer Science, University of Calgary, Calgary, Alberta, Canada, T2N 1N4

SALLY A. GOLDMAN sg@cs.wustl.edu
H. DAVID MATHIAS dmath@cs.wustl.edu
Dept. of Computer Science, Washington University, St. Louis, MO 63130

Abstract. We present several efficient parallel algorithms for PAC-learning geometric concepts in a constant-dimensional space. The algorithms are robust even against malicious classification noise of any rate less than 1/2. We first give an efficient noise-tolerant parallel algorithm to PAC-learn the class of geometric concepts defined by a polynomial number of $(d-1)$-dimensional hyperplanes against an arbitrary distribution where each hyperplane has a slope from a set of known slopes. We then describe how boosting techniques can be used so that our algorithms' dependence on ϵ and δ does not depend on d. Next we give an efficient noise-tolerant parallel algorithm to PAC-learn the class of geometric concepts defined by a polynomial number of $(d-1)$-dimensional hyperplanes (of unrestricted slopes) against a uniform distribution. We then show how to extend our algorithm to learn this class against any (unknown) product distribution. Next we define a complexity measure of any set S of $(d-1)$-dimensional surfaces that we call the *variant* of S and prove that the class of geometric concepts defined by surfaces of polynomial variant can be efficiently learned in parallel under a product distribution (even under malicious classification noise). Furthermore, we show that the VC-dimension of the class of geometric concepts defined by a set of surfaces S of variant v is at least v. Finally, we give an efficient, parallel, noise-tolerant algorithm to PAC-learn any geometric concept defined by a set S of $(d-1)$-dimensional surfaces of polynomial area under a uniform distribution.

Keywords: computational learning theory, geometric concepts, parallel computation

1. Introduction

We present several efficient parallel algorithms for PAC-learning geometric concepts over $[0,1]^d$ (for d any constant). These algorithms are robust even against malicious classification noise of any rate less than 1/2. We note that our algorithms directly apply to any domain of the form $[c_1, c_2]^d$ for any constants c_1 and c_2 by just rescaling the points[1]. We present all of our algorithms as statistical query (SQ) algorithms. Furthermore, we show that our algorithms can all be modified so that they can tolerate malicious classification noise. In this noise model, an adversary can arbitrarily label some randomly chosen examples and thus the noise rate seen on different portions of the domain can be different (as opposed to the *random* classification noise handled by any SQ algorithm).

We first present an efficient noise-tolerant parallel algorithm to PAC-learn geometric concepts defined by a polynomial number of $(d-1)$-dimensional hyperplanes

against an arbitrary distribution where each hyperplane has a slope from a set of known slopes. We refer to this class as *R-linear geometric concepts* since they are defined by hyperplanes restricted to be of one of r known slopes. We demonstrate that this algorithm can easily be modified to handle malicious classification noise. We then describe how hypothesis boosting can be used so that our algorithms' dependence on ϵ and δ does not depend on d (the number of dimensions).

Next we present an efficient noise-tolerant parallel algorithm that PAC-learns any geometric concept defined by a polynomial number of $(d-1)$-dimensional hyperplanes (of unrestricted slopes) against a uniform distribution. We refer to this class as *linear geometric concepts* since they are defined by hyperplanes. Next, we show how to modify our algorithm that learns linear geometric concepts against a uniform distribution so that it works against any product distribution.

We then consider the class of geometric concepts defined by any set S of $(d-1)$-dimensional surfaces. We refer to this class as *non-linear geometric concepts* since these surfaces are not restricted to be linear. We define a complexity measure of any set of $(d-1)$-dimensional surfaces S that we call the *variant* of S. We then give a noise-tolerant parallel algorithm that PAC-learns the class of geometric concepts defined by surfaces of polynomial variant under a product distribution. Considering this class under the product distribution is of particular interest since it is well known that even surfaces with variant 1 have infinite VC-dimension (see, for example, Baum [8]) and thus, by the results of Ehrenfeucht, *et.al.* [20], this class is not efficiently learnable under an arbitrary distribution.

Finally, we give an efficient parallel noise-tolerant algorithm to learn 2-dimensional geometric concepts defined by a set of 1-dimensional surfaces (or curves) of polynomial length under the uniform distribution.

We note that not only do we give efficient parallel PAC-algorithms that tolerate malicious classification noise of any rate $\eta < 1/2$, but to our knowledge no comparable sequential noise-free PAC-learning algorithms for most of these classes were known prior to our work[2]. Also since our algorithms are simple, tolerate noise, and can take advantage of unlabelled as well as labelled data, they may have practical value.

2. Preliminaries

The learning model we use in this work is the *probably approximately correct* (PAC) model of Valiant [36]. In this model, the learner is presented with examples, chosen randomly from instance space $\mathcal{X}$ according to some unknown probability distribution $\mathcal{D}$. Let f be an unknown target function from known concept class $\mathcal{C}$. The learner must return a hypothesis h that classifies at least $(1-\epsilon)$ of $\mathcal{X}$ consistent with f, with probability at least $(1-\delta)$, where ϵ and δ are given parameters. That is, with high probability, the hypothesis must correctly classify most of the instances (by weight under distribution $\mathcal{D}$).

The basic PAC model assumes that the examples given to the learner are drawn randomly from $\mathcal{D}$ and labeled *correctly* based on the target concept. In this work

we also consider a variant of the PAC model in which the labeled examples that the learner receives are corrupted by *random classification noise* [2]. In this noise model, each example is still drawn at random from $\mathcal{D}$. However, with probability η (where $0 \leq \eta < 1/2$ is called the *noise rate*), the learner receives an incorrect label and with probability $1-\eta$, the learner receives the correct label. Thus the example drawn is labeled incorrectly, at random, with probability η. In the *malicious classification noise* model [35], each example is still drawn at random according to $\mathcal{D}$. however, with probability η an adversary selects the label provided and with probability $1-\eta$ the learner receives the correct label. In the more general *malicious noise* model [37], with probability η the adversary can provide an example and label of its choice.

To achieve noise-tolerance we use the *statistical query* model [27], [18], [3], [4]. In this model, rather than sampling labeled examples, the learner requests from an oracle the value of various statistics. The additive statistical query oracle returns the probability, within some additive constant, that some predicate is true relative to the distribution. An additive statistical query takes the form $\text{STAT}_{\mathcal{D}}(\chi, \alpha)$ where χ is a predicate over labeled examples drawn from $\mathcal{D}$ and α is the relative error bound. For target function f, let $P_\chi = \Pr_{\mathcal{D}}[\chi(x, f(x)) = 1]$. Then $\text{STAT}_{\mathcal{D}}(\chi, \alpha)$ must return an estimate $\hat{P}_\chi$ such that $P_\chi - \alpha \leq \hat{P}_\chi \leq P_\chi + \alpha$. The learner may also request unlabeled examples. Kearns [27] has shown that all statistical query algorithms are robust against *random* classification noise for any noise rate $\eta < 1/2$. He has also shown that statistical query algorithms are robust against small amounts of malicious errors.

Let $\mathcal{Z}$ be the set of integers. Let $\mathcal{X}$, the instance space, be $[0,1]^d$. We use the column vectors $\vec{y} = (y_1, \ldots, y_d)$ to denote the d dimension variables, and $\vec{x} = (x_1, \ldots, x_d)$ to denote an element of $\mathcal{X}$. Let $\vec{a} = (a_1, \ldots, a_d)$ be a row vector where $a_i \in \mathcal{Z}$. A d-dimensional *hyperplane* is $\vec{a} \cdot \vec{y} = b$ for some $b \in \mathcal{Z}$. A d-dimensional *halfspace* is $\vec{a} \cdot \vec{y} \succ b$ where $\succ \in \{>, \geq, <, \leq\}$. We call $G \subseteq \mathcal{X}$, a *subspace*. $\mathcal{D}(G)$ denotes the weight of the points in G under distribution $\mathcal{D}$.

In this paper we study geometric concept classes defined over $[0,1]^d$ where we assume that d, the number of dimensions of the space, is a constant. The class of *R-Linear Geometric Concepts*, denoted $\mathcal{C}_s^{\text{R-linear}}$, is the class of geometric concepts defined by any Boolean function over at most s, (d–1)-dimensional halfspaces each of which has a slope restricted to come from one of r known slopes. More formally, we associate a Boolean variable v_i for $1 \leq i \leq s$ with each of the s halfspaces. For $\vec{x} \in \mathcal{X}$, we define $v_i(\vec{x}) = 1$ if and only if $\vec{x}$ is in the halfspace associated with v_i. Then $\mathcal{C}_s^{\text{R-linear}} = \{f(v_1, \ldots, v_s) \mid f \text{ is any Boolean function on } s \text{ halfspaces over } v_1, \ldots, v_s\}$. We define $\mathcal{C}^{\text{R-linear}} = \cup_s \mathcal{C}_s^{\text{R-linear}}$. We define the complexity, $C(f)$ of $f(v_1, \ldots, v_s)$ as $|v_1| + \cdots + |v_s| + |f|$ where $|v_i|$ (respectively, $|f|$) denotes the representation size of the halfspace (respectively, the Boolean function). Given that a real number can be represented in unit space, for this class $|v_i| \leq \lceil \lg r \rceil \leq \lg r + 1$. Observe that assuming all halfspaces define a border of the target then $|f| \geq s$. We define the complexity $C(g)$ of $g \in \mathcal{C}_s^{\text{R-linear}}$ to be the minimum over all $f(v_1, \ldots, v_s) = g$ of $C(f)$. The class of *Linear Geometric Concepts*, denoted $\mathcal{C}_s^{\text{linear}}$, is exactly like

$C_s^{\text{R-linear}}$ except that each halfspace can be of an arbitrary unknown slope. For ease of exposition, for both of these classes we assume that the learner knows s. If s is unknown standard doubling techniques can be applied.

We now define a new complexity measure for a $(d-1)$-dimensional surface. We say dimension i *contains* surface S if there exists a constant c such that for all points of S, $y_i = c$. We define the complexity of the set of surfaces S in the following manner. We say that any (possibly non-continuous) surface S is a *1-variant surface* if any axis-parallel line, along any dimension *not* containing S, intersects S at most once. In other words, S is a 1-variant surface if for each i, for which dimension i does not contain S, and for every $(d-1)$-dimensional point, $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_d$ there is at most one x_i for which $x_1, \ldots, x_d$ is contained within S. Thus any $(d-1)$-dimensional hyperplane is a 1-variant surface. For $d = 2$, any monotone function[3] defines a 1-variant surface.

For any arbitrary set S of $(d-1)$-dimensional surfaces, we define the *complexity or variant* of S, $\mathcal{V}(S)$, as the minimum v such that S can be expressed as a union of v 1-variant surfaces. For example, a circle has variant 4 since if you divide it into four equal size arcs using diametric, axis-parallel cuts, each arc is a 1-variant curve. The class of *Non-Linear Geometric Concepts*, which we denote by $C_{\mathcal{V}(S)}^{\text{non-linear}}$, is the class of geometric concepts defined by a Boolean function over any $(d-1)$-dimensional surface S of variant at most $\mathcal{V}$. We assume, without loss of generality, that the learner is given an upperbound for $\mathcal{V}$. When studying $C_{\mathcal{V}(S)}^{\text{non-linear}}$ we use $\mathcal{V}(S)$ as a measure for the complexity of the target concept.

We also study, for $d = 2$, the class $C_L^{\text{non-linear}}$ in which we use the length of the curves defining the target concept as the measure of complexity. This result is better than the result based on variant in some cases, such as a very small tight spiral where the length can be relatively small even though it has a high variant.

Our algorithms consist of a first stage in which an unlabelled sample is used to gather information about the unknown distribution, and a second stage in which a random sample is used to compute the conditional probability of the form Pr[random point $\vec{x}$ is positive | $\vec{x}$ is in a subspace of $\mathcal{X}$]. Thus it follows from known results about the noise tolerance of statistical query (SQ) algorithms [27], [18], [3] that our algorithms can tolerate *random* classification noise of any noise rate bounded above by $1/2$. Because of the simplicity of the statistical queries we make, rather than using this general technique, we can directly compute the sample complexity required to obtain significantly better bounds. Furthermore, it is easily seen that our noise-tolerant algorithms can handle *malicious* classification noise of any noise rate bounded above by $1/2$.

We also note that no efficient parallel algorithm exists to exactly learn[4] the union of s axis-parallel boxes over $\{1, \ldots, n\}^d$ (which is the discretized version of a subclass of C_s^{linear}). We use an adversarial argument. The adversary answers all membership queries "no". Only when the learner asks an equivalence query is the adversary forced to reveal a positive instance. The learner can then use membership queries to learn the remainder of the box containing that instance. Thus, s equivalence queries are necessary. Combined with the work of Bshouty and Cleve [13] we get

that any parallel algorithm to exactly identify this class must have $\Omega(s)$ parallel time which is not efficient (*i.e.* it is not poly-logarithmic).

3. Previous Work

Considerable work has been done on learning geometric concepts in the PAC model. In particular, unions and intersections of halfspaces have been considered. Blum and Rivest [10] show that there does not exist an efficient proper[5] learning algorithm for unions of s halfspaces, unless $P = NP$. (That is, any such algorithm must have exponential dependence on d.) Baum [7] gives an algorithm that efficiently learns a union of s halfspaces in a constant number of dimensions. Blumer *et al.* [11] give a similar result. Both algorithms return hypotheses containing $O(s \lg m)$ halfspaces where m is the size of the sample. Baum gives efficient algorithms for learning several classes with infinite VC-dimension (such as convex polyhedral sets) under the uniform distribution [8]. Haussler [25] also gives distribution specific algorithms for several classes of functions.

Research has also been done on the learnability of unions of axis-parallel boxes. Blumer *et al.* present an algorithm to PAC-learn an s-fold union of boxes in E^d by drawing a sufficiently large sample of size $m = poly\left(\frac{1}{\epsilon}, \lg\frac{1}{\delta}, s, d\right)$, 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 [29] 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 $s(2d)^s$ boxes consistent with the sample. Thus both the time and sample complexity of their algorithm depend polynomially on $s, d^s, \frac{1}{\epsilon}$, and $\lg\frac{1}{\delta}$. So for s constant this yields an efficient PAC algorithm. We note that either of these PAC algorithms can be applied to the class $\bigcup_s \text{BOX}_n^d$ giving efficient PAC algorithms for this class for either d constant or s constant.

Finally, under a variation of the PAC model in which membership queries can be made, Frazier *et al.* [21] 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. Their algorithm learns this subclass of general unions of boxes in time polynomial in both s and d. Recall that since $\bigcup_s \text{BOX}_n^d$ generalizes DNF, a polynomial-time algorithm for arbitrary d and s 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. Bshouty, *et al.* [12] give a PAC algorithm to learn the discretized version of R-linear geometric concepts with random classification noise.

A number of results [31], [32], [33], [28], [5], [30], [17], [15], [26], [16], [24], [14] have been obtained for geometric classes in Angluin's query learning model [1] as well.

There has also been work on learning in parallel [39], [9], [40], [13], [6]. Of particular relevance is the work of Vitter and Lin [39], [40]. They say that a concept class

$\mathcal{C}$ is NC-learnable (respectively, NC_η^{MC}-learnable) if there exists a PAC-learning algorithm for $\mathcal{C}$ in RNC that runs in poly-logarithmic time with a polynomial number of processors on an arithmetic CRCW PRAM[6] in the noise-free setting (respectively, when the examples are corrupted with malicious classification noise of rate η). Along with giving several non-geometric results, they prove that the following geometric classes are NC-learnable: non-axis parallel rectangles (for $d = 2$), linearly separable functions (for constant d), simple k-gons (k constant), unions of s axis-parallel rectangles in the plane[7]. Furthermore they prove that the class of axis-parallel rectangles, linear separators and simple k-gons are NC_η^{MC}-learnable for $\eta < 1/2$. Berger, Rompel and Shor [9] gave NC approximation algorithms for the unweighted and weighted set cover problems. They use these approximation algorithms to prove the NC-learnability of concept classes formed by taking either finite unions or finite intersections of a fixed base class of finite VC-dimension for which there is a NC hypothesis finder.

4. Learning $C_s^{\text{R-linear}}$ Under an Arbitrary Distribution

In this section we describe a parallel algorithm to NC_η^{MC}-learn $C_s^{\text{R-linear}}$ for any $\eta < \frac{1}{2}$. For clarity, we introduce our algorithm as a sequential algorithm and analyze the sample complexity. We then explain our method for handling noise. Next we parallelize the algorithm, and analyze the parallel time complexity.

4.1. A Sequential Algorithm

The algorithm we present runs in two stages. First it draws unlabeled sample S_1 of size m_1 that is used to partition $[0,1]^d$ into a set of subspaces. This is done by passing through each point of S_1 a hyperplane with each of the possible r slopes, where each of these hyperplanes defines three regions (the hyperplane itself and the two open halfspaces on either side). The learner then draws labeled sample S_2 of size m_2 that is used to determine the classification of each of the subspaces created. Pseudo-code for this learning algorithm is shown in Figure 1.

We now analyze the sample complexity of the sequential algorithm.

4.2. Analysis

Let $\mathcal{D}$ be the distribution from which points are drawn. Let f be the target concept generated from s hyperplanes with r distinct, known slopes. Let $\vec{a}_i\vec{y} = b_i$, $i = 1, \ldots, s$ be the set of hyperplanes and let $\mu_1, \ldots, \mu_r$ be the set of slopes. For each hyperplane in the target, we define two parallel, bounding hyperplanes – one "above" and one "below" the target hyperplane. Recall that $\mathcal{D}(G)$ represents the weight under distribution $\mathcal{D}$ of the points contained within subspace $G \subseteq \mathcal{X}$. Specifically, for hyperplane i we define

Learn-$\mathcal{C}_s^{\text{R-linear}}$:
Draw an unlabeled sample S_1 of size $m_1 = \frac{4s}{\epsilon} \ln \frac{4s}{\delta}$
For $i = 1$ to m_1
For $j = 1$ to r
Pass a hyperplane of slope j through point i
Draw a labeled sample S_2 of size $m_2 = \frac{2(4sr)^d}{\epsilon^{d+1}} \ln^d \frac{4s}{\delta} \ln \left(\frac{2}{\delta} \left(\frac{4sr}{\epsilon} \right)^d \ln^d \frac{4s}{\delta} \right)$
For $k = 1$ to t
Label subspace k with the label of any point in subspace k

Figure 1. A sequential algorithm for learning geometric concepts defined by hyperplanes of arbitrary, known slopes against an arbitrary distribution. t is the number of regions into which the hypothesis divides $[0,1]^d$.

$$w_{1,i} = \min \left\{ w \mid \mathcal{D}(b_i \leq \vec{a}_i \cdot \vec{y} \leq b_i + w) \geq \frac{\epsilon}{(4s)} \right\}$$

and

$$w_{2,i} = \min \left\{ w \mid \mathcal{D}(b_i - w \leq \vec{a}_i \cdot \vec{y} \leq b_i) \geq \frac{\epsilon}{(4s)} \right\}$$

as the distances of the bounding hyperplanes from the target hyperplane. We define

$$A_{1,i} = \{ \vec{x} \in \mathcal{X} \mid b_i \leq \vec{a}_i \cdot \vec{x} \leq b_i + w_{1,i} \}$$

and

$$A_{2,i} = \{ x \in \mathcal{X} \mid b_i - w_{2,i} \leq \vec{a}_i \cdot \vec{x} \leq b_i \}$$

as the sets of points contained between the target hyperplane and the bounding hyperplanes. Note that by the definition of $w_{1,i}$ and $w_{2,i}$, $\mathcal{D}(A_{1,i}) \geq \frac{\epsilon}{(4s)}$ and $\mathcal{D}(A_{2,i}) \geq \frac{\epsilon}{(4s)}$.

Let $\mathcal{A} = \{A_{1,1}, \ldots, A_{1,s}, A_{2,1}, \ldots, A_{2,s}\}$. We want to ensure that, with high probability, all $A \in \mathcal{A}$ contain at least one point from S_1. The following lemma addresses this.

LEMMA 1 *A sample S_1 of size $m_1 = \frac{4s}{\epsilon} \ln \frac{4s}{\delta}$ is sufficient to ensure that with probability at least $1 - \frac{\delta}{2}$ each $A \in \mathcal{A}$ contains at least one point from S_1.*

Proof: The probability that a particular $A \in \mathcal{A}$ does not contain a point of S_1 is at most $(1 - \epsilon/(4s))^{m_1}$ and thus the probability that any $A \in \mathcal{A}$ does not contain a point of S_1 is at most $2s(1 - \epsilon/(4s))^{m_1}$. Thus, setting

$$2s\left(1-\frac{\epsilon}{4s}\right)^{m_1} \leq \frac{\delta}{2},$$

and solving for m_1 yields

$$m_1 = \frac{4s}{\epsilon}\ln\frac{4s}{\delta}$$

suffices for the size of the first sample. (The other half of the confidence bound, as well as the other half of the error bound, is reserved for the second stage of the algorithm.) □

The total number of subspaces created (where each of the $m_1 r$ hyperplanes divides each region it intersects into three parts) is

$$t \leq (m_1 r)^d = \Theta((m_1 r)^d) = \Theta\left(\left(\frac{sr}{\epsilon}\right)^d \log^d \frac{s}{\delta}\right)$$

where the inequality holds for $d > 1$. (See, for example, Edelsbrunner [19].) In the second stage of the algorithm we must determine the classification of all of the subspaces that contain at least $\tau = \frac{\epsilon}{2t}$ of the distribution. The following lemma addresses the size of the second sample.

LEMMA 2 *A sample S_2 of size $m_2 = \frac{2(4sr)^d}{\epsilon^{d+1}} \ln^d \frac{4s}{\delta} \ln\left(\frac{2}{\delta}\left(\frac{4sr}{\epsilon}\right)^d \ln^d\left(\frac{4s}{\delta}\right)\right)$ is sufficient to ensure that with probability at least $1-\frac{\delta}{2}$ every subspace G such that $D(G) \geq \tau$ contains at least one point of S_2.*

Proof: Let $G_1, \ldots, G_\rho$ be the subspaces of the hypothesis for which $D(G_i) \geq \tau$. The probability that any particular G_i, $i = 1, \ldots, \rho$, does not contain any point from the second sample is $(1-\tau)^{m_2}$. Thus the probability that any of the G_i does not contain any point from S_2 is $\rho(1-\tau)^{m_2} \leq t(1-\tau)^{m_2}$. We want to ensure that, with probability at least $\frac{\delta}{2}$, each G_i contains at least one point from S_2. This gives us

$$t(1-\tau)^{m_2} \leq \frac{\delta}{2}.$$

Solving for m_2 yields that

$$\begin{aligned} m_2 &= \frac{1}{\tau}\ln\frac{2t}{\delta} = \frac{2t}{\epsilon}\ln\frac{2t}{\delta} = \frac{2(m_1 r)^d}{\epsilon}\ln\frac{2(m_1 r)^d}{\delta} \\ &= \frac{2(4sr)^d}{\epsilon^{d+1}}\ln^d\frac{4s}{\delta}\ln\left(\frac{2}{\delta}\left(\frac{4sr}{\epsilon}\right)^d \ln^d\left(\frac{4s}{\delta}\right)\right) \end{aligned}$$

suffices for the size of the second sample. □

We now show that with high probability our algorithm produces a hypothesis with low error.

LEMMA 3 *Our algorithm to learn $C_s^{\text{R-linear}}$ returns, with probability at least $1-\delta$, a hypothesis that has error at most ϵ. The sample complexity is polynomial in $s, r, \frac{1}{\epsilon}$, and $\log\frac{1}{\delta}$,*

Proof: After processing S_1 we have that, with probability at least $1 - \frac{\delta}{2}$, each hyperplane in the target is bounded by two parallel hyperplanes (one from each side) and the weight *strictly* between the two bounding hyperplanes is at most $2\frac{\epsilon}{4s} = \frac{\epsilon}{2s}$.

Let B_i, $i = 1, \ldots, s$, be the set of points strictly between the closest parallel hyperplanes (in the hypothesis) of the form $\vec{a}_i \cdot \vec{y} = b_i$. Therefore, $D(B_i) \leq \frac{\epsilon}{(2s)}$. Thus, for $B = B_1 \cup \cdots \cup B_s$, we have $D(B) \leq \frac{\epsilon}{2}$. In this first phase of the algorithm the domain has been divided into at most $\Theta((m_1 r)^d)$ subspaces. Note that in all subspaces R, except $R \subseteq B$, all of the points in R have the same sign. Thus, if $R \not\subseteq B$ it is sufficient to get one point in R in order to find the sign of all of the points in R. Furthermore, since $D(B) \leq \frac{\epsilon}{2}$ we can afford to improperly label all subspaces $R \subseteq B$.

In the second stage of the algorithm, with probability at least $1 - \frac{\delta}{2}$, all subspaces with weight at least τ are properly classified. Misclassifying those subspaces with weight less than τ adds at most $\tau t = \frac{\epsilon}{2}$ to the learning error. □

Our algorithm, therefore, returns an ϵ-good approximation of the target concept, with probability at least $1-\delta$, with sample complexity polynomial in $s, r, \frac{1}{\epsilon}$ and $\log \frac{1}{\delta}$. As previously noted, this algorithm is very easily stated as an algorithm in the statistical query (SQ) model.

COROLLARY 1 *There exists a statistical query algorithm that efficiently learns* $\mathcal{C}_s^{\text{R-linear}}$.

Proof Sketch: The first stage of the algorithm is unchanged. In the second stage we replace drawing sample S_2 with statistical queries. The predicate χ in the queries is [random point $\vec{x}$ is positive | $\vec{x}$ is in a subspace of $\mathcal{X}$]. We use the output of these querues to label the regions of the hypothesis. □

Thus our algorithm is able to handle random classification noise. Next we prove that our algorithm can tolerate malicious classification noise.

4.3. Handling Malicious Classification Noise

In this section we show that our algorithm for learning $\mathcal{C}_s^{\text{R-linear}}$ is robust against malicious classification noise of rate $\eta < 1/2$.

Allowing malicious classification noise requires only two changes to our first algorithm. The first stage of the algorithm is unaffected since the learner does not use the labels of the first sample and thus, noise has no effect. In the second stage of the algorithm the size of the sample drawn must depend on the upperbound, η_b, for the noise rate. The second change concerns determining the signs of the subspaces created in the first stage of the algorithm. In the analysis of our sequential, noise-free algorithm we noted that in every subspace with weight at least τ all points must have the same sign. When points are misclassified this is no longer the case. Thus, rather than simply returning as the label for a subspace the label of any point in the subspace, we return the result of a majority vote of the labels of all the points in the subspace.

Let η_b be the upper bound on the noise rate. Thus, a point is mislabeled with probability at most η_b and is properly labeled with probability $p \geq 1 - \eta_b$. We now determine the necessary size of the second sample in the presence of malicious classification noise.

LEMMA 4 *A sample of size m_2, where m_2 is*

$$O\left(\frac{1}{(1-2\eta_b)^2}\left(\log\frac{sr}{\epsilon}+\log\frac{s}{\delta}\right)+\frac{(sr)^d}{\epsilon^{d+1}}\log^d\frac{s}{\delta}\left(\log\frac{1}{\delta}+\log\frac{sr}{\epsilon}+\log\log\frac{s}{\delta}\right)\right)$$

is sufficient to ensure that with probability at least $1-\frac{\delta}{4}$ more than half of the the points in each subspace are properly labeled.

Proof: Let m_G be the number of points in subspace G, and S be the number of points that appear properly labeled of those m_G points. We want prove that

$$\Pr\left[S \leq \frac{1}{2}m_G\right] \leq \frac{\delta}{4t}.$$

Using Chernoff bounds we get that

$$\Pr\left[S \leq \frac{1}{2}m\right] \leq e^{-2m(\frac{1}{2}-(1-\eta_b))^2}.$$

Thus, we have

$$e^{-2m\left(\frac{1}{2}-(1-\eta_b)\right)^2} \leq \frac{\delta}{4t}.$$

Solving for m we find that

$$m = \frac{1}{2(1-2\eta_b)^2}\log\frac{4t}{\delta} = \frac{1}{2(1-2\eta_b)^2}\left(\ln\frac{4}{\delta}+d\ln\frac{4sr}{\epsilon}+d\ln\frac{4s}{\delta}\right)$$

suffices to ensure that with probability at least $1-\delta/4$ for *all* subspaces greater than half of the points in that subspace are properly labeled. We must still ensure that all of the subspaces of sufficiently high weight contain an point from the sample. Thus, it suffices to select S_2 of size

$$m_2 = max\ \left\{\frac{1}{2(1-2\eta_b)^2}\left(\ln\frac{4}{\delta}+d\ln\frac{4sr}{\epsilon}+d\ln\frac{4s}{\delta}\right),\ \frac{2(4sr)^d}{\epsilon^{d+1}}\ln^d\frac{4s}{\delta}\ln\left(\frac{2}{\delta}\left(\frac{4sr}{\epsilon}\right)^d\ln^d\left(\frac{4s}{\delta}\right)\right)\right\}.$$

□

We now present an efficient parallel implementation of our algorithm for learning this class. While the overall sample complexity remains unchanged, the parallel time complexity is much improved over that of the sequential algorithm.

4.4. A Parallel Algorithm

In this section we give a parallel algorithm for the class $C_s^{\text{R-linear}}$. We have chosen, whenever possible, to increase the number of processors in order to decrease the running time. We do not claim that our algorithm is optimal in its use of processors or time. It is intended to illustrate how our learning algorithm can be implemented to run in parallel in poly-logarithmic time.

The parallel version of our algorithm is quite similar to the sequential version shown in Figure 1. Recall that in the first stage of our algorithm, for each point in the sample we add a hyperplane with each possible slope. Thus, we add $m_1 r$ hyperplanes. To parallelize this stage of the algorithm we use $m_1 r$ processors. We divide these processors into m_1 groups of r processors each. For each group we choose a point $x \in S_1$. Each processor in a group is assigned a slope and creates a hyperplane with that slope passing through x. Each processor then picks the minimum dimension i for which $a_i \neq 0$ and computes

$$x_i = \frac{b - (a_1 x_1 + \cdots + a_{i-1} x_{i-1} + a_{i+1} x_{i+1} \cdots a_d x_d)}{a_i}$$

where $a_1 x_1 + \cdots + a_d x_d = b$ is a hyperplane defined by the processor. Then, for each slope, all of the processors with that slope (from all groups) sort their hyperplanes with that slope according to the calculated values of x_i.

In the second stage of the algorithm we label each subspace created in the first stage. To accomplish this we use $O(m_1 r m_2 + (m_1 r)^d)$ processors. For each of the m_2 points in the second sample we have a processor. Each of these processors determines which subspace contains its point and then reports the label of its point to the processor for that subspace. To determine which subspace contains point x, we use $m_1 r$ processors for each of the m_2 points. Each of these processors corresponds to a hyperplane created in the first stage of the algorithm. The job of each processor is to decide if point x lies "above", on or "below" its associated hyperplane. Each processor then reports the result to its two nearest neighbors (one above and one below). Thus, the processor for each point knows the nearest hyperplanes of each slope. This information is enough to index into the array of processors for the subspaces and report the label of the point. The processors for the points in a subspace write their labels concurrently. The last value that is written is used as the label of that subspace. (This method works only in the noise-free case. If malicious classification noise is present then a majority vote must be used.) We now analyze our parallel algorithm.

THEOREM 1 *Let* $m_1 = \frac{4s}{\epsilon} \ln \frac{4s}{\delta}$ *and*

$$m_2 = O\left(\frac{(sr)^d}{\epsilon^{d+1}} \log^d \frac{s}{\delta} \left(\log \frac{sr}{\epsilon} + \log \frac{1}{\delta}\right)\right).$$

There is a parallel algorithm to NC-learn $C_s^{\text{R-linear}}$ *using* $m_1 + m_2$ *points. The algorithm uses* $O(m_1 r m_2 + (m_1 r)^d)$ *processors and* $O(\log m_1)$ *parallel time.*

Proof: In Lemma 1 we prove that m_1 is sufficient for the size of sample S_1 and in Lemma 2 we prove that m_2 is sufficient for the size of sample S_2. Lemma 3 proves that the algorithm returns an ϵ-good hypothesis with probability at least $1-\delta$.

All that remains is to prove the running time. In the first stage we create r groups each containing m_1 hyperplanes. We must calculate an intercept for each hyperplane and then sort the hyperplanes by these intercepts. Since we have a processor for each hyperplane the calculation of the intercepts is performed in constant parallel time. The sorting requires $O(\log m_1)$ parallel time. In the second stage the determination of the nearest pair of hyperplanes of each slope for each point requires constant time. The concurrent write into the appropriate location for the subspace containing each point also requires constant parallel time. Thus, the entire second stage requires $O(1)$ parallel time. The total running time for the algorithm is $O(\log m_1)$. □

The following corollary addresses malicious classification noise in the parallel algorithm.

COROLLARY 2 *There is a parallel algorithm to NC_η^{MC}-learn $C_s^{\text{R-linear}}$, for any $\eta < 1/2$. The sample complexity and number of processors are poly$(r, s, 1/\epsilon, \log 1/\delta)$ and the parallel time complexity is poly$(\log r, \log s, \log 1/\epsilon, \log\log 1/\delta)$.*

Proof: There are only two relevant differences from the sequential algorithm. The first is the size of the second sample which was addressed in Lemma 4. The second difference is the running time of the second stage. Due to the noise, instead of using a concurrent write and taking the result for the label of a subspace, we must do a majority vote. This can be achieved in parallel time that is logarithmic in the number of examples in the subspace (using a parallel prefix computation). □

5. Boosting To Reduce the Dependence on ϵ and δ

In this section we show how hypothesis boosting techniques can be used to modify all of our algorithms so that their dependence on ϵ and δ do not depend on d.

THEOREM 2 *Let $\mathcal{A}$ be an algorithm that learns a class C of Boolean functions under any distribution with $m(s, d, \epsilon, \delta)$ examples for any function m. Then there exists an algorithm that learns C with $m(s, d, 1/4, 1/2)poly(1/\epsilon, \log(1/\delta))$ examples.*

Proof: We handle δ by running $\mathcal{A}$ with $\delta = 1/2$ and $\epsilon/2$, $\ell = \log(2/\delta)$ times to get ℓ hypotheses $h_1, \ldots, h_\ell$. The sample size is $m(s, d, \epsilon/2, 1/2)\log(2/\delta)$ and with probability at least $1 - \delta/2$ one of the hypotheses is $\epsilon/2$-good. Using hypothesis testing on $h_1, \ldots, h_\ell$ with probability at least $1 - \delta/2$ we can find one that is ϵ-good using $poly(1/\epsilon, \log(1/\delta))$ more examples. This shows that the sample size $m(s, d, \epsilon, \delta)$ can be changed to $m(s, d, \epsilon/2, 1/2)poly(1/\epsilon, \log(1/\delta))$. Let $\mathcal{B}$ be the resulting algorithm.

We now handle ϵ using boosting. We run $\mathcal{B}$ for $\epsilon = 1/4$ to get a weak approximation of the target. Using boosting techniques we need to generate $\log(1/\epsilon)$ weak

hypotheses to get an ϵ-good approximation [34], [22], [23]. The blow up in δ is $poly(\log(1/\delta))$. Therefore, the boosting algorithm generates an ϵ-good approximation of the target with sample size

$$m(s, d, 1/4, 1/2)poly(1/\epsilon, \log(1/\delta)).$$

□

6. Learning $\mathcal{C}_s^{\text{linear}}$ Under a Uniform Distribution

In this section we present an efficient noise-tolerant parallel algorithm for learning $\mathcal{C}_s^{\text{linear}}$ under a uniform distribution. A key step of the algorithms we present in the remainder of this paper is to partition $\mathcal{X}$ into ϕ^d *subspaces* by partitioning each dimension of $\mathcal{X}$ into ϕ pieces. We use $I_{i,1}, \ldots, I_{i,\phi}$ to denote the intervals used to partition $[0, 1]$ in the ith dimension where each $I_{i,j}$ is the interval such that $c_1 \leq y_i < c_2$ for constants c_1, c_2. We use $\mathcal{G}_\phi^d$ to denote the grid defined by

$$\{I_{1,1}, \ldots, I_{1,\phi}\} \times \{I_{2,1}, \ldots, I_{2,\phi}\} \times \cdots \times \{I_{d,1}, \ldots, I_{d,\phi}\}.$$

So $G \in \mathcal{G}_\phi^d$ is one of the ϕ^d subspaces that are in $\mathcal{G}_\phi^d$.

In this section we make $\mathcal{G}_\phi^d$ a *uniform grid* by selecting $I_{i,j}$ for $1 \leq i \leq d$ and $1 \leq j \leq \phi$ be the interval $(j-1)\phi \leq y_i < j\phi$. Thus we partition each dimension into ϕ intervals each of width $1/\phi$. We now prove the following lemma that is used by most of our remaining results.

LEMMA 5 *There exist at most $d \cdot \phi^{d-1}$ subspaces in the uniform grid $\mathcal{G}_\phi^d$ that are intersected by a $(d-1)$-dimensional hyperplane through $\mathcal{G}_\phi^d$.*

Proof: A $(d-1)$-dimensional hyperplane in $\mathcal{X}$ can be written as

$$y_1 = \frac{b}{a_1} - \frac{a_2}{a_1}y_2 - \cdots - \frac{a_d}{a_1}y_d$$

where $a_1 = \max\{a_1, \ldots, a_d\}$. Since a_1 is the largest coefficient a change of at most ℓ in one of $y_2, \ldots, y_d$, will cause a change of at most ℓ in y_1.

Think of $\mathcal{G}_\phi^d$ as ϕ slices of $\mathcal{G}_\phi^{d-1}$ where $\mathcal{G}_\phi^{d-1}$ is obtained by projecting dimension 1 out of $\mathcal{G}_\phi^d$. That is, $\mathcal{G}_\phi^{d-1}$ is defined by

$$\{I_{2,1}, \ldots, I_{2,\phi}\} \times \{I_{3,1}, \ldots, I_{3,\phi}\} \times \cdots \times \{I_{d,1}, \ldots, I_{d,\phi}\}.$$

We now focus on one of the ϕ^{d-1} subspaces $G \in \mathcal{G}_\phi^{d-1}$ and the corresponding ϕ slices in $\mathcal{G}_\phi^d$. Note that in all points in G each of the $d-1$ variables change by at most $1/\phi$. Thus it follows that y_1 changes by at most $(d-1)/\phi$. Finally, since the "depth" of each slice is $1/\phi$ it follows that at most $\dfrac{(d-1)/\phi}{1/\phi} + 1 = d$ slices of $\mathcal{G}_\phi^d$ corresponding to G can be cut by any one hyperplane. Summing over all ϕ^{d-1} subspaces of $\mathcal{G}_\phi^{d-1}$ yields the desired result. □

Now for the main result of this section.

THEOREM 3 *There exists a parallel algorithm to NC-learn any concept from* $\mathcal{C}_s^{\text{linear}}$ *under the uniform distribution with a sample complexity of size* $\left(\frac{sd}{\epsilon}\right)^d \left(d \ln \frac{sd}{\epsilon} + \ln \frac{1}{\delta}\right)$ *where* s *is the number of hyperplanes defining the target concept.*

Proof: Let $\phi = (sd/\epsilon)$, and let $\mathcal{G}_\phi^d$ be a uniform grid. Thus for each of the ϕ^d subspaces $G \in \mathcal{G}_\phi^d$ it follows that $\mathcal{D}(G) = (1/\phi)^d = (\epsilon/(2s))^d$ where $\mathcal{D}$ is the uniform distribution.

We draw a sample S_1 of size m_1 such that with probability at least $1-\delta$ at least one point from S_1 falls into each $G \in \mathcal{G}_\phi^d$. For any given subspace $G \in \mathcal{G}_\phi^d$, the probability that m_1 points are drawn none of which are in G is $(1-(1/\phi)^d)^{m_1}$. Thus setting $\Pr[\exists\ G \in \mathcal{G}_\phi^d \mid \text{ no point in } S_1 \text{ is in } G] \leq \phi^d \left(1 - \left(\frac{1}{\phi}\right)^d\right)^{m_1} \leq \delta$, and solving for m_1 yields that a sample of size

$$\begin{aligned} m_1 &= \phi^d \left(d \ln \phi + \ln \tfrac{1}{\delta}\right) \\ &= \left(\tfrac{sd}{\epsilon}\right)^d \left(d \ln \tfrac{sd}{\epsilon} + \ln \tfrac{1}{\delta}\right) \\ &= O\left(\left(\tfrac{s}{\epsilon}\right)^d \left(\log \tfrac{s}{\epsilon} + \log \tfrac{1}{\delta}\right)\right) \end{aligned}$$

suffices.

For each $G \in \mathcal{G}_\phi^d$ our algorithm classifies all points in G based on the label of a point from S_1 that is in G. If S_1 has multiple points in G, we arbitrarily choose one. We now prove that given there is a sample point in each $G \in \mathcal{G}_\phi^d$, which occurs with probability at least $1-\delta$, the error of our hypothesis is at most ϵ. By Lemma 5, we have that each hyperplane of the target concept can intersect at most $d\phi^{d-1}$ subspaces of $\mathcal{G}_\phi^d$. Thus a total of at most $sd\phi^{d-1}$ subspaces are intersected by the hyperplanes defining the target concept. For each $G \in \mathcal{G}_\phi^d$ not intersected, all points in G are classified in the same manner and thus our algorithm predicts correctly. Our hypothesis may misclassify those subspaces that are intersected. However, since each subspace has weight at most ϕ^{-d} and at most $sd\phi^{d-1}$ subspaces are intersected the overall error is at most $\frac{sd\phi^{d-1}}{\phi^d} = sd/\phi = \epsilon$.

Finally, the techniques of Theorem 1 can be used to show that with a polynomial number of processors, the parallel implementation runs in poly-logarithmic time. □

By drawing a larger sample and using a majority vote of the points in each subspace of $\mathcal{G}_\phi^d$ to select the classification for each subspace, it is easily shown that this algorithm is robust against malicious classification noise.

COROLLARY 3 *The class* $\mathcal{C}_s^{\text{linear}}$ *is efficiently* NC_η^{MC}*-learnable under the uniform distribution for any* $\eta < 1/2$.

7. Learning $\mathcal{C}_s^{\text{linear}}$ Under a Product Distribution

In this section we give a parallel algorithm to NC-learn any concept from $\mathcal{C}_s^{\text{linear}}$ under a product distribution. A product distribution over points in d-dimensional

space is a distribution $\mathcal{D} = (\mathcal{D}_1, \ldots, \mathcal{D}_d)$ in which each dimension's distribution is independent. Thus, the probability of choosing point $p = (x_1, x_2, \ldots, x_d)$ is $\mathcal{D}_1(x_1) \cdot \mathcal{D}_2(x_2) \cdots \mathcal{D}_d(x_d)$ where $\mathcal{D}_i(x_i)$ is the probability that x_i is drawn under distribution $\mathcal{D}_i$.

In both methods we have two phases, in the first phase we use the sample to partition $\mathcal{X}$ into subspaces such that the weight of any subspace is not too large. Then in the second phase we draw a sample to classify each subspace in such a way that with high probability the total error is at most ϵ. The following lemma extends Lemma 5 when $\mathcal{G}_\phi^d$ may be non-uniform.

LEMMA 6 *There exist at most $d \cdot 2^d \cdot \phi^{d-1}$ subspaces of any grid $\mathcal{G}_\phi^d$ that are intersected by a $(d-1)$-dimensional hyperplane through $\mathcal{G}_\phi^d$.*

Proof: For each dimension i for $1 \leq i \leq d$ consider the $(d-1)$-dimensional grid $g_i = \mathcal{G}_\phi^{d-1}$ obtained by projecting out dimension i from $\mathcal{G}_\phi^d$. Now consider the ϕ^{d-1} points from g_i defined by $\{p_1, p_2, \ldots, p_\phi\}^{d-1}$ and imagine projecting each such point infinitely in both directions in dimension i. Clearly any $(d-1)$-dimensional hyperplane intersects each of these lines at at most 1 point. Furthermore, since each point of intersection can border at most 2^{d-1} regions it follows that the number of cut regions of $\mathcal{G}_\phi^d$ defined by i dimensional borders is at most $2^d\phi^{d-1}$. Finally, since every region intersected is intersected at some border, all regions are counted by adding up the number intersected in each of the d dimensions. Thus the number of regions of $\mathcal{G}_\phi^d$ intersected by a $(d-1)$-dimensional hyperplane is at most $d \cdot 2^d \cdot \phi^{d-1}$. □

We now describe our algorithm for learning linear geometric concepts under the product distribution. The key to this algorithm is to use a sample from the unknown product distribution to divide each dimension into ϕ intervals of nearly equal weight. Namely, with high probability we can guarantee that each of the ϕ intervals created have weight at least $1/(4\phi)$ and at most $4/\phi$. Then we can proceed with a second phase like that used when learning this class under the uniform distribution (with the only change being that our choice for ϕ must be adjusted slightly).

THEOREM 4 *Consider the interval $[0,1]$ and let $\mathcal{D}$ be an unknown distribution over $[0,1]$. Using a sample of size $O(\phi \log(\phi/\delta))$ we can partition $[0,1]$ into ϕ intervals such that with probability at least $1-\delta$ each interval has weight at least $1/(4\phi)$ and at most $4/\phi$.*

The proof for this theorem follows from the Vapnik-Chervonenkis theory [38] and the fact that intervals have a VC-dimension of 2.

Here we summarize the basic technique used. The algorithm is very simple. Draw a sample of size $m = O\left(\phi \log \frac{\phi}{\delta}\right)$ and then divide $[0,1]$ into intervals such that each interval contains m/ϕ points. That each of these intervals has weight between $1/(4\phi)$ and $4/\phi$ then follows.

We now apply this theorem to obtain an algorithm to PAC-learn $\mathcal{C}_s^{\text{linear}}$ under a product distribution.

THEOREM 5 *There exists a parallel algorithm to NC-learn any concept from* $\mathcal{C}_s^{\text{linear}}$ *under a product distribution with a sample complexity of size*

$$O\left(\left(\frac{s}{\epsilon}\right)^d\left(\log\frac{s}{\epsilon}+\log\frac{1}{\delta}\right)\right).$$

Proof: Since we are working with a product distribution, we can apply Theorem 4 in each dimension by projecting the sample onto each dimension and then using a confidence parameter of $\delta/(2d)$. Thus it immediately follows that, with probability at least $d\left(\frac{\delta}{2d}\right)=\delta/2$, in each dimension each interval has weight between $1/(4\phi)$ and $4/\phi$. We now can proceed as in the proof of Theorem 3 except that we let $\phi=(sd4^d)/\epsilon$. Since each region has weight at least $(1/(4\phi))^d$ it is easily shown that by drawing a sample of size $(4\phi)^d\left(d\ln\phi+\frac{2}{\delta}\right)$, with probability at least $1-\delta/2$, there is at least one sample point in each region. Thus the only error caused is by regions that are cut by the hyperplanes defining the target concept. Since each subspace has weight at most $(4/\phi)^d$ and at most $sd\phi^{d-1}$ regions are intersected the overall error is at most ϵ. □

Then using the same techniques as in Section 4.3 both of these algorithms can be modified to tolerate malicious classification noise of any rate less than $1/2$.

COROLLARY 4 *The class* $\mathcal{C}_s^{\text{linear}}$ *is efficiently* NC_η^{MC}*-learnable under any product distribution for any* $\eta<1/2$.

8. Learning $\mathcal{C}_{\mathcal{V}}^{\text{non-linear}}$ Under a Product Distribution

In this section we present an efficient noise-tolerant parallel algorithm for learning $\mathcal{C}_{\mathcal{V}(S)}^{\text{non-linear}}$ under a product distribution where the target concept is defined by any set of surfaces of polynomial variant. As we mentioned in the introduction, it is well known that even a surface with a variant of 1 may have infinite VC-dimension. We now give a proof.

THEOREM 6 $VCD(\mathcal{C}_1^{\text{non-linear}})=\infty$.

Proof: For ease of exposition, we give the proof for $d=2$, it easily generalizes for arbitrary d. Take any number m and take the m points $x_i=(i/m,i/m)$ for $i=0,\ldots,m-1$. We now show that the set $\{x_i\}$ is shattered by a surface of variant 1. To see this take any $A\subseteq\{x_i\}$ and define a monotone function that passes above the points of A and below the points of $\{x_i\}-A$. (Line segments can even be used to define this monotone function). Finally, observe that the resulting surface has variant 1 and shatters the m points. □

Thus, by the results of Ehrenfeucht et al. [20], there is no efficient algorithm (even if computation time is unbounded) to PAC-learn $\mathcal{C}_{\mathcal{V}(S)}^{\text{non-linear}}$, even when $\mathcal{V}(S)=1$, against an arbitrary distribution. However, we are able to show that by modifying our algorithm from the previous section we can efficiently PAC learn a geometric

concept defined by any set of surfaces of polynomial variant against any product distribution (in parallel, with malicious classification noise).

The key to our result of this section is the following lemma.

LEMMA 7 *There exist at most $d \cdot 2^d \cdot \phi^{d-1}$ subspaces of $\mathcal{G}_\phi^d$ that are intersected by any $(d-1)$-dimensional surface of variant-1.*

Proof: This proof follows directly from the proof of Lemma 6 by noting that the only property of a hyperplane used in that proof is that any axis-parallel line intersects a hyperplane at most once. Since, by definition of a 1-variant surface, we have the property that an axis-parallel line intersects it at most once, the result follows. □

We now give the main result of this section.

THEOREM 7 *There exists a parallel algorithm to NC-learn $\mathcal{C}_{\mathcal{V}(S)}^{\text{non-linear}}$ under the product distribution with a sample complexity of size*

$$O\left(\left(\frac{v}{\epsilon}\right)^d \left(\log \frac{v}{\epsilon} + \log \frac{1}{\delta}\right)\right)$$

where $v = \mathcal{V}(S)$ is the variant of the surface S that defines the target concpet.

Proof: Here we use an algorithm like that used to learn $\mathcal{C}_s^{\text{linear}}$ against the product distribution where $\mathcal{V}(S)$ replaces s, the number of hyperplanes defining the target concept. Namely, let $\phi = (\mathcal{V}(S)d4^d)/\epsilon$. By applying Theorem 4 in each dimension, we get that, with probability at least $d\left(\frac{\delta}{2d}\right) = \delta/2$, in each dimension each interval has weight between $1/(4\phi)$ and $4/\phi$. Since each region has weight at least $(1/(4\phi))^d$ it is easily shown that by drawing a sample of size $(4\phi)^d\left(d\ln\phi + \frac{2}{\delta}\right)$, with probability at least $1-\delta/2$, there is at least one sample point in each region. Thus the only error caused is by regions that are cut by the hyperplanes defining the target concept. Since each subspace has weight at most $(4/\phi)^d$ and at most $\mathcal{V}(S)d\phi^{d-1}$ regions are intersected the overall error is at most ϵ. □

Then using the same techniques as in Section 4.3 this algorithm can be modified to tolerate malicious classification noise of any rate less than 1/2.

COROLLARY 5 *The class $\mathcal{C}_{\mathcal{V}(S)}^{\text{non-linear}}$ is efficiently NC_η^{MC}-learn under any product distribution for any $\eta < 1/2$.*

9. Learning Concepts Defined by Surfaces of Polynomial Length

We briefly describe a technique to learn any 2-dimensional geometric concept defined by any set of surfaces where L is the total length of the surfaces. The key to the result of this section is the following lemma

LEMMA 8 *Any 1-dimensional surface of length L can intersect at most $4L\phi$ subspaces of the uniform grid $\mathcal{G}_\phi^2$.*

Proof: Imagine cutting L into at most $L\phi$ pieces of length $1/\phi$. We now argue that each piece can intersect at most 4 regions of $\mathcal{G}_\phi^d$ and thus the total number of regions intersected is at most $4L\phi$.

Recall that each $G \in \mathcal{G}_\phi^2$ is a square where each side has length $1/\phi$. It is easily shown that there are at most 4 subspaces of $\mathcal{G}_2^d$ that the segment of curve of length $1/\phi$ can intersect. Since there are at most $L\phi$ such segments the result follows. □

THEOREM 8 *There exists a parallel algorithm (for $d = 2$) to NC-learn any concept from $\mathcal{C}_L^{\text{non-linear}}$ under the uniform distribution with a sample complexity of size*

$$\left(\frac{4L}{\epsilon}\right)^2 \left(2\ln\frac{4L}{\epsilon} + \ln\frac{1}{\delta}\right)$$

where L is the total length of the curves defining the target concept.

Proof: Here we use an algorithm like that to learn $\mathcal{C}_s^{\text{linear}}$ except that we let $\phi = 4L/\epsilon$. Using the same analysis as in the proof of Theorem 3 it follows that if we draw a sample of size

$$m = \phi^2\left(2\ln\phi + \ln\frac{1}{\delta}\right) = \left(\frac{4L}{\epsilon}\right)^2\left(2\ln\frac{4L}{\epsilon} + \ln\frac{1}{\delta}\right) = O\left(\frac{A^2}{\epsilon^2}\left(\ln\frac{A}{\epsilon} + \ln\frac{1}{\delta}\right)\right)$$

then, with high probability, there is least one point in each subspace of $\mathcal{G}_\phi^d$.

We now prove that the error of our hypothesis is at most ϵ. From Lemma 8 it follows that the surface intersects at most $2^d L\phi = 4L\phi$ regions of $\mathcal{G}_\phi^d$. For those regions not intersected by one of the hyperplanes defining the target, all points in that region are the same and thus the majority vote is correct. For those regions that are intersected the majority vote may produce the wrong classification. However, since each region has weight at most $1/\phi^2$ and at most $4L\phi$ regions are intersected the overall error is at most $4L/\phi = \epsilon$. □

Then using the same techniques as in Section 4.3 this algorithm can be modified to tolerate malicious classification noise of any rate less than 1/2.

10. Concluding Remarks

We have described a set of simple parallel algorithms for efficiently learning various classes of geometric concepts in constant-dimensional space even when there is a high rate of random classification noise. When the target concept is defined by taking boolean combinations of halfspaces, we provide an algorithm to learn this class (1) against any distribution when the hyperplanes defining the halfspaces use a set of known slopes, and (2) against product distributions when the slopes of

the hyperplanes defining the halfspaces are arbitrary. We then look at concepts defined by non-linear surfaces, and define the variant, a new complexity measure for this class. While the VC-dimension of the class of concepts defined by surfaces of variant one is infinite, we are still able to efficiently learn any concept defined by a set of surfaces of polynomial variant against any product distribution.

Since all of our algorithms are easily formulated as statistical query algorithms, in addition to handling labelling noise, known results allow us to handle small amounts of malicious noise, and various types of noise effecting the distribution of the random examples (*e.g.* see Decatur [18]). In addition, they can all be efficiently implemented in parallel.

Another nice feature of our algorithms is that they can take advantage of unlabelled as well as labelled data which may be of value for some real-life applications, especially when combined with the simplicity and robustness of the algorithms.

Acknowledgements

We thank Subhash Suri for his help in proving Lemma 6, and the anonymous COLT95 reviewers for their very valuable suggestions.

Notes

1. Furthermore, we note that all of these results can easily be converted to learn the corresponding geometric classes defined over $\Re^d$ by first drawing a sufficiently large sample to find a bounding box that has small weight outside.
2. There is a known sequential PAC algorithm to learn the discretized version of R-linear geometric concepts with random classification noise [12].
3. A function f is monotone if $y_1 > y_2 \Rightarrow f(y_1) > f(y_2)$.
4. An algorithm *exactly learns* a concept class if for any concept in the class it returns a hypothesis that correctly classifies every instance in the instance space.
5. A learning algorithm is proper if all hypotheses come from the concept class.
6. In the arithmetic CRCW PRAM model each memory location can hold a single real number.
7. This result can be extended to higher dimensions and other geometrical objects like circles, triangles, and polygons in which each side has one of a constant number of fixed orientations.

References

1. Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.
2. Dana Angluin and Philip Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.
3. J. A. Aslam and S. E. Decatur. General bounds on statistical query learning and PAC learning with noise via hypothesis boosting. In *34th Annual Symposium on Foundations of Computer Science*, pages 282–291, November 1993.
4. J. A. Aslam and S. E. Decatur. Specification and simulation of statistical query algorithms for efficiency and noise tolerance. In *Proceedings of the Eighth Annual ACM Conference on Computational Learning Theory*, pages 437–446, July 1995.

5. 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.
6. J. L. Balcázar, J. Díaz, R. Gavaldà, and O. Watanabe. An optimal parallel algorithm for learning DFA. In *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory*, pages 208–217. ACM Press, New York, NY, 1994.
7. E. B. Baum. On learning a union of half spaces. *Journal of Complexity*, 6(1):67–101, March 1990.
8. E. B. Baum. The perceptron algorithm is fast for nonmalicious distributions. *Neural Computation*, 2:248–260, 1990.
9. B. Berger, J. Rompel, and P. W. Shor. Efficient NC algorithms for set cover with applications to learning and geometry. *Journal of Computer and System Sciences*, 49(2):454–477, 1994.
10. Avrim L. Blum and Ronald L. Rivest. Training a 3-node neural network is NP-Complete. *Neural Networks*, 5:117–127, 1992.
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, Zhixiang Chen, and Steve Homer. On learning discretized geometric concepts. In *35th Annual Symposium on Foundations of Computer Science*, pages 54–63, November 1994.
13. Nader H. Bshouty and Richard Cleve. On the exact learning of formulas in parallel. In *33rd Annual Symposium on Foundations of Computer Science*, pages 1–15, October 1992.
14. Nader H. Bshouty, Paul W. Goldberg, Sally A. Goldman, and H. David Mathias. Exact learning of discretized concepts. Technical Report WUCS-94-19, Washington University, 1994.
15. 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.
16. Zhixiang Chen and Steven Homer. The bounded injury priority method and the learnability of unions of rectangles. Unpublished manuscript, May 1994.
17. 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.
18. S. E. Decatur. Statistical queries and faulty PAC oracles. In *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory*, pages 262–268. ACM Press, New York, NY, 1993.
19. Herbert Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer-Verlag, New York, 1987.
20. Andrzej Ehrenfeucht, David Haussler, Michael Kearns, and Leslie Valiant. A general lower bound on the number of examples needed for learning. *Information and Computation*, 82(3):247–251, 1989.
21. 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.
22. Yoav Freund. Boosting a weak learning algorithm by majority. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 202–216. Morgan Kaufmann, August 1990.
23. Yoav Freund. An improved boosting algorithm and its implications on learning complexity. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 391–398, July 1992.
24. 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.
25. David Haussler. Generalizing the PAC model: sample size bounds from metric dimension-based uniform convergence results. In *30th Annual Symposium on Foundations of Computer Science*, pages 40–45, October 1989.

26. Steven Homer and Zhixiang Chen. Fast learning unions of rectangles with queries. Unpublished manuscript, July 1993.
27. M. Kearns. Efficient noise-tolerant learning from statistical queries. In *Proc. 25th Annu. ACM Sympos. Theory Comput.*, pages 392–401. ACM Press, New York, NY, 1993.
28. Nick Littlestone. Learning when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.
29. 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.
30. 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.
31. 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.
32. 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.
33. Wolfgang Maass and György Turán. Lower bound methods and separation results for on-line learning models. *Machine Learning*, 9:107–145, 1992.
34. Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.
35. 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.
36. Leslie Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.
37. Leslie Valiant. Learning disjunctions of conjunctions. In *Proceedings of Ninth International Joint Conference on Artificial Intelleigence*, 1985.
38. V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications*, XVI(2):264–280, 1971.
39. Jeffrey S. Vitter and Jyh-Han Lin. Learning in parallel. In *Proceedings of the 1988 Workshop on Computational Learning Theory*, pages 106–124. Morgan Kaufmann, 1988.
40. Jeffrey S. Vitter and Jyh-Han Lin. Learning in parallel. *Information and Computation*, pages 179–202, 1992.