Testing Problems with Sub-Learning Sample Complexity

Michael Kearns
AT&T Labs Research
180 Park Avenue
Florham Park, NJ, 07932
mkearns@research.att.com

Dana Ron*
Laboratory for Computer Science, MIT
545 Technology Square
Cambridge, MA, 02138
danar@theory.lcs.mit.edu

October 19, 1998

Abstract

We study the problem of determining, for a class of functions H, whether an unknown target function f is contained in H or is "far" from any function in H. Thus, in contrast to problems of *learning*, where we must *construct* a good approximation to f in H on the basis of sample data, in problems of *testing* we are only required to determine the *existence* of a good approximation. Our main results demonstrate that, over the domain $[0,1]^d$ for constant d, the number of examples required for testing grows only as $O(s^{1/2+\delta})$ (where δ is any small constant), for both decision trees of size s and a special class of neural networks with s hidden units. This is in contrast to the $\Omega(s)$ examples required for learning these same classes. Our tests are based on combinatorial constructions demonstrating that these classes can be approximated by small classes of coarse partitions of space, and rely on repeated application of the well-known Birthday Paradox.

*Supported by an ONR Science Scholar Fellowship at the Bunting Institute.

1 Introduction

A considerable fraction of the computational learning theory literature is devoted to a basic and natural question: for a given class of functions H and a distribution P on inputs, how many random examples of an unknown function f are required in order to *construct* a good approximation to f in H? In this paper, we consider a natural and potentially important relaxation of this problem: how many random examples of an unknown function f are required in order to simply *test* whether a good approximation to f *exists* in H? Thus, in contrast to the standard learning problem, in problems of testing we are not required to actually construct a good hypothesis, but only to assert its existence — so under the appropriate definitions, the resources required for testing are always at most those required for learning. In this work, we show that for certain natural classes H, the number of examples required for testing can actually be considerably less than for learning. Even more dramatic gaps are shown to hold when the measure is the number of queries required.

The motivation for studying learning problems is by now obvious. Why study testing problems? In addition to its being a basic statistical question, if we can find fast and simple solutions for testing problems that require little data, we may be able to use them to choose between alternative hypothesis representations without actually incurring the expense of running the corresponding learning algorithms. For example, suppose that in a setting where data is expensive, but the final accuracy of our learned hypothesis is paramount, we are considering running C4.5 (a fast algorithm) to find a decision tree hypothesis (a relatively weak representation). But we may also want to consider running backpropagation (a slow algorithm) to find a multilayer neural network (a relatively powerful representation, requiring more data, but with perhaps greater accuracy). Ideally, we would like a fast, low-data test that informs us whether this investment would be worthwhile.

The results we present here are far from providing tests of such practicality, but they do examine natural and common hypothesis representations, and introduce some basic tools for testing algorithms that may point the way towards further progress. Specifically, our main results demonstrate tests for s-node decision trees, and for a special class of neural networks of s hidden units (both over $[0,1]^d$), that require only $O(s^{1/2+\delta})$ (for any small constant δ) random examples when the input dimension d is held constant and the underlying distribution is uniform. This is in contrast to the $\Omega(s)$ examples required, under the same conditions, to learn a hypothesis that is even a weak approximation to such functions.

The tests we describe will "accept" any function that is a size s decision tree or neural network, and "reject" any function that is "far" from all size s' decision trees or neural networks, where s' is not much larger than s. Thus, even though acceptance ensures the existence of a small decision tree or neural network nontrivially approximating the target function, we have far fewer examples than are necessary to actually construct the approximation. We also provide tests using membership queries in which the difference between testing and learning is even more dramatic, from $\Omega(s)$ queries required for learning to $\mathrm{poly}(\log(s))$ or even a constant number of queries required for testing.

What form do these tests have? We begin by noting that they must look quite different from the standard learning algorithms. With only roughly $\sqrt{s}$ examples, if we begin by seeing how well we can fit the data with a size s function, we will always be able to achieve zero training error, even if the labels were generated randomly. The tests we describe are based on two central ideas: *locality* and the *Birthday Paradox*. Roughly speaking, for both decision trees and neural networks, we show that there are different notions of two input points being "near" each other, with the property that for any size s

function, the probability that a pair of near points have the same label significantly exceeds $1/2$. It is not hard to construct notions of nearness for which this will hold — for instance, calling two points near only if they are identical. The trick is to give the *weakest* such notion, one sufficiently weak to allow the application of the Birthday Paradox. In particular, we use the Birthday Paradox to argue that a small sample is likely to contain a pair of near points. Thus, all of the resulting tests are appealingly simple: they involve taking a small sample or making a small number of queries, pairing nearby points, and checking the fraction of pairs in which the two points have common labels.

The heart of our proofs are purely combinatorial lemmas in which we prove that certain notions of locality yield relatively coarse partitions of space that can approximate the partition induced by any small decision tree or neural network, respectively. We believe these combinatorial lemmas are of independent interest and may find application elsewhere. Variations on these combinatorial constructions can hopefully yield improved tests. There are two main aspects of our results that call for improvement: the exponential dependence on the dimension d (thus limiting their interest to a fixed, small dimension), and the distance (from any decision tree or neural network of size s') at which we can ensure that a tested function is rejected, which is $1/2 - c$ for a constant c. The two cases for which we can provide tests that work for any distance are decision trees of dimension 1 (interval functions), and classes of functions that are defined by all labelings of a fixed set of partitions of the domain.

Prior Work

There are several lines of prior work that inspired the current investigation. Problems of testing and their relationship to learning were recently studied by Goldreich et al. [GGR96], whose framework we follow and generalize. They were in turn inspired by the PAC learning model, and built on the model of Rubinfeld and Sudan [RS96] that emerged in the context of program testing. However, the testing algorithms described in these papers, as well as in other related work [BLR93, Rub94, GR97, GR98], all utilize queries, and except for illustrative examples we are not aware of any testing algorithms that use only random examples and have lower sample complexity than that required for learning.

The function classes studied in these earlier works are defined by algebraic or graph-theoretic properties. In particular, there are algorithms for testing whether a function is multilinear (or far from any such function) [BLR93, BCH$^+$95] and for testing whether a function is a multivariate polynomial [RS96]. These classes demonstrate very large gaps between the complexity of testing and of learning, when queries are available and the input distribution is uniform.

Our work can also be viewed as a study of the sample complexity of classical *hypothesis testing* [Kie87] in statistics, where one wishes to accept or reject a "null hypothesis", such as "the data is labeled by a function approximable by a small decision tree". Other related works from the learning literature along these lines includes papers by Kulkarni and Zeitouni [KZ93], and Yaminishi [Yam95].

Outline

The outline of the paper is as follows: in Section 2, we introduce several related notions of testing. Section 3 illuminates the basic ideas of locality and the Birthday Paradox on a simple example, interval functions on the real line. In Sections 4 and 5 we give our main testing results for decision trees and a special class of neural networks, respectively. In Section 6 we give a general statement regarding the complexity of testing classes of functions that can be approximated by families of labeled partitions.

In Section 7 we prove a connection between testing and the standard notion of weak learning from the computational learning theory literature. In Section 8 we show a lower bound on the number of examples required for testing the classes we consider, which matches our upper bounds, in terms of the dependence on s, up to logarithmic factors.

2 Definitions

We start by noting that though we consider only Boolean function, our definitions easily generalize to real-valued functions. We begin with a needed definition for the distance of a function from a class of functions.

Definition 1 *Let f and f' be a pair of functions over domain X, H a class of functions over X, and P a distribution over X. The* distance *between f and f' with respect to P is $\mathrm{dist}_P(f, f') \stackrel{\text{def}}{=} \Pr_{x \sim P}[f(x) \neq f'(x)]$, and the distance between f and H (with respect to P) is $\mathrm{dist}_P(f, H) \stackrel{\text{def}}{=} \min_{f' \in H} \mathrm{dist}_P(f, f')$. For $\epsilon \in [0, 1]$, if $\mathrm{dist}_P(f, H) > \epsilon$, then we say that f is ϵ-*far *from H (with respect to P). Otherwise, it is ϵ-*close*. We use $\mathrm{dist}(\cdot, \cdot)$ as a shorthand for $\mathrm{dist}_U(\cdot, \cdot)$, where U is the uniform distribution over X.*

Before giving our definitions for testing problems, we reiterate the point made in the introduction that, under the appropriate definitions, the resources required for testing for H will always be at most those required for learning H (see Proposition 14). Our interest is in cases where testing is considerably *easier* than learning.

In our first definition of testing we generalize the definition given by Goldreich et al. [GGR96]. There the task was to determine whether an unknown function f belongs to a particular class of functions H or is ϵ-far from H. We relax this definition to determining whether $f \in H$ or f is ϵ-far from a class H', where $H' \supseteq H$. This relaxation is especially appropriate when dealing with classes of functions that are indexed according to *size*. In such a case, H might contain all functions of size at most s (for instance, decision trees of at most s leaves) in a certain family of functions $\mathcal{H}$ (all decision trees), while H' might contain all functions in $\mathcal{H}$ that have size at most s', where $s' \geq s$. An ideal test would have $H' = H$ ($s' = s$), and ϵ arbitrarily small, but it should be clear that relaxations of this ideal are still worthwhile and nontrivial.

Definition 2 *Let H be a class of functions over X, let $H' \supseteq H$, let $\epsilon \in (0, 1/2]$, and let P be a distribution over X. We say that H* is testable with rejection boundary (H', ϵ) in m examples (respectively, m queries) with respect to P *if there is an algorithm T such that:*

- *If T is given m examples, drawn according to P and labeled by any $f \in H$ (respectively, T makes m queries to such an f), then with probability $2/3$, T accepts.*
- *If T is given m examples, drawn according to P any labeled by any function f that is ϵ-far from H' with respect to P (respectively, T makes m queries to such an f), then with probability $2/3$, T rejects.*

If neither of the above conditions on f holds, then T may either accept or reject.

Note that our insistence that the success probability of the algorithm be at least $2/3$ is arbitrary; any constant bounded away from $1/2$ will do, as the success probability can be amplified to any desired value $1 - \delta$ by $O(\log(1/\delta))$ repetitions of the test.

Our next definition can be viewed as pushing the rejection boundary of the previous definition to the extreme of truly random functions.

Definition 3 *Let H be a class of functions over X, and let P be a distribution over X. We say that H* is testable against a random function in m examples with respect to *P if there is an algorithm T such that:*

- *If T is given m examples drawn according to P and labeled by any $f \in H$, then with probability $2/3$, T accepts.*
- *If T is given m examples drawn according to P and labeled randomly, then with probability $2/3$, T rejects. The probability here is taken both over the choice of examples and their random labels.*

Note that whenever H is testable with rejection boundary (H, ϵ) in m examples (queries), and H is such that with high probability a random function is ϵ-far from H (for some $\epsilon < 1/2$), then it is testable against a random function in m examples (queries).

Our final definition has a slightly different flavor than the previous two. Here there are two classes of functions, H_1 and H_2, and the task is to determine whether f belongs to H_1 or to H_2.

Definition 4 *Let H_1 and H_2 be classes of functions over X, and let P be a distribution over X. We say that (H_1, H_2)* are testable in m examples (respectively, m queries) with respect to *P if there is an algorithm T such that:*

- *If T is given m examples, drawn according to P and labeled by any $f \in H_1$ (respectively, T makes m queries to such an f), then with probability $2/3$, T outputs 1.*
- *If T is given m examples, drawn according to P and labeled by any $f \in H_2$ (respectively, T makes m queries to such an f), then with probability $2/3$, T outputs 2.*

If neither of the above conditions on f holds, then T may output either 1 or 2.

Note that in the above definition it is implicitly assumed that there is a certain *separation* between the classes H_1 and H_2 — that is, that there exists some $\epsilon \in (0, 1]$ such that for every $h_1 \in H_1$ and $h_2 \in H_2$, $\mathrm{dist}_P(h_1, h_2) > \epsilon$. In particular, this implies that $H_1 \cap H_2 = \emptyset$. Otherwise, it would not be possible to distinguish between the classes in any number of examples. An alternative definition would require that the testing algorithm be correct only when the function f belongs to one class and is ϵ-far from the other.

3 Interval Functions

We start by describing and analyzing a testing algorithm for the class of *interval functions*. The study of this simple class serves as a good introduction to subsequent sections.

For any size s, the class of interval functions with at most s intervals, denoted INT_s, is defined as follows. Each function $f \in \mathrm{INT}_s$ is defined by $t \leq s - 1$ *switch points*, $a_1 < \ldots < a_t$, where $a_i \in (0, 1)$. The value of f is fixed in each *interval* that lies between two switch points, and alternates between 0 to 1 when going from one interval to the next.

It is not hard to verify that learning the class INT_s requires $\Omega(s)$ examples (even when the underlying distribution is uniform). In fact, $\Omega(s)$ is also a lower bound on the number of membership queries

necessary for learning. As we show below, the complexity of *testing* under the uniform distribution is much lower — it suffices to observe $O(\sqrt{s})$ random examples, and the number of queries that suffice for testing is independent of s.

Theorem 1 *For any integer $s > 0$ and $\epsilon \in (0, 1/2]$, the class of interval functions INT_s is testable with rejection boundary $(\mathrm{INT}_{s/\epsilon}, \epsilon)$ under the uniform distribution in $O(\sqrt{s}/\epsilon^{1.5})$ examples or $O(1/\epsilon)$ queries. The running time of the testing algorithm is linear in the number of examples (respectively, queries) used.*

The basic property of interval functions that our testing algorithm exploits is that *most* pairs of *close* points belong to the same interval, and thus have the same label. The algorithm scans the sample for such close pairs (or queries the function on such pairs), and accepts only if the fraction of pairs in which both points have the same label is above a certain threshold. In the proof below we quantify the notion of closeness, and analyze its implications both on the rejection boundary for testing and on the number of examples needed. Intuitively, there is the following tradeoff: as the distance between the points in a pair becomes smaller, we are more confident that they belong to the same interval (in the case that $f \in \mathrm{INT}_s$); but the probability that we observe such pairs of points in the sample becomes smaller, and the class H' in the rejection boundary becomes larger.

Proof: We first describe the testing algorithm in greater detail. Let $s' = s/\epsilon$, and consider the partition of the domain $[0, 1]$ imposed by a one-dimensional grid with s' equal-size *cells* (intervals) $c_1, \ldots, c_{s'}$. Given a sample S of size $m = O(\sqrt{s'}/\epsilon)$ (=$O(\sqrt{s}/\epsilon^{1.5})$), we sort the examples $x_1, \ldots, x_m$ into bins $B_1, \ldots, B_{s'}$, where the bin B_j contains points belonging to the cell c_j. Within each (non-empty) bin B_j, let $x_{i_1}, x_{i_2}, \ldots, x_{i_t}$ be the examples in B_j, ordered according to their appearance *in the sample*, and let us pair the points in each such bin according to this order (thus, x_{i_1} is paired with x_{i_2}, x_{i_3} with x_{i_4}, and so on). We call these pairs the *close* pairs, and we further call a pair *pure* if it is close *and* both points have the same label. The algorithm accepts f if the fraction of pure pairs (among all close pairs) is at least $1 - 3\epsilon/4$; otherwise it rejects. When the algorithm is instead allowed queries, it uniformly selects $m' = O(1/\epsilon)$ of the grid cells, uniformly draws a pair of points in each cell chosen, and queries f on these pairs of points. The acceptance criteria is unaltered.

Our first central observation is that by our choice of m, with high probability (say, $5/6$), the number m'' of close pairs is at least $m' = O(1/\epsilon)$. To obtain this lower bound on m'', assume we restricted our choice of pairs by breaking the random sample into $4m'$ random subsamples, each of size $2\sqrt{s'}$, and considered only close pairs that belong to the same subsample. We claim that by the well-known *Birthday Paradox*, for each subsample, the probability that the subsample contains a close pair is at least $1/2$. To see why this is true, think of each subsample S' as consisting of two parts, S'_1 and S'_2, each of size $\sqrt{s'}$. We consider two cases: In this first case, S'_1 already contains two examples that belong to a common cell and we are done. Otherwise, each example in S'_1 belongs to a different cell. Let this set of $\sqrt{s'}$ cells be denoted C and recall that all cells have the same probability mass $1/s$. Thus, the probability that S'_2 does *not* contain any example from a cell in C is

$$\left(1 - |C| \cdot \frac{1}{s}\right)^{|S'_2|} = \left(1 - \frac{1}{\sqrt{s'}}\right)^{\sqrt{s'}} < e^{-1} < 1/2 \qquad (1)$$

as claimed. Hence, with very high probability, at least a fourth of the subsamples (that is, at least m') will contribute a close pair, in which case $m'' \geq m'$. Since the close pairs are equally likely to fall in

each cell c_j and are uniformly distributed within each cell, the correctness of the algorithm when using examples reduces to its correctness when using queries, and so we focus on the latter. To establish that the algorithm is a testing algorithm we need to address two cases.

CASE 1: $f \in \mathrm{INT}_s$. For $t = 1 \ldots, m'$, let χ_t be random variable that is 0 if the tth close pair is pure, and 1 otherwise. Thus χ_t is determined by a two-stage process: (1) The choice of the t'th grid cell c_t; (2) The selection of the two points inside that cell. When c_t is a subinterval of some interval of f, then the points always have the same label, and otherwise they have a different label with probability at most $1/2$. Since f has at most s intervals, the number of cells that intersect intervals of f (that is, are *not* subintervals of f's intervals) is at most s, and since there are s/ϵ grid cells, the probability of selecting such a cell is at most ϵ. It follows that for each t,

$$\mathrm{E}[\chi_t] \quad \leq \quad \epsilon \cdot (1/2) + (1-\epsilon) \cdot 0 \quad = \quad \epsilon/2 \; . \tag{2}$$

By a multiplicative Chernoff bound, with probability at least $2/3$, the average of the χ_t's (which is just the fraction of close pairs that are not pure), is at most $3\epsilon/4$, as required.

CASE 2: $\mathrm{dist}(f, \mathrm{INT}_{s'}) > \epsilon$. In order to prove that in this case the algorithm will reject with probability at least $2/3$ we prove the contrapositive: If the algorithm accepts with probability greater than $1/3$ then there exists a function $f' \in \mathrm{INT}_{s'}$ that is ϵ-close to f.

Let $f' \in \mathrm{INT}_{s'}$ be the (equally spaced) s'-interval function that gives the majority label according to f to each grid cell. We claim that if f is accepted with probability greater than $1/3$ then $\mathrm{dist}(f, f') \leq \epsilon$. For contradiction assume that $\mathrm{dist}(f, f') > \epsilon$. For each grid cell c_j let $\epsilon_j \in [0, 1/2]$ be the probability mass of points in c_j that have the minority label of f among points in c_j. Thus, $\mathrm{dist}(f, f') = \mathrm{E}_j[\epsilon_j]$, and so, by our assumption, $\mathrm{E}_j[\epsilon_j] > \epsilon$. On the other hand, if we define χ_t as in Case 1, then we get that

$$\mathrm{E}[\chi_t] \quad = \quad \mathrm{E}_j\left[2\epsilon_j(1-\epsilon_j)\right] \quad \geq \quad \mathrm{E}_j[\epsilon_j] \tag{3}$$

where the second inequality follows from $\epsilon_j \leq 1/2$. By our assumption on f, $\mathrm{E}[\chi_t] > \epsilon$, and by applying a multiplicative Chernoff bound, with probability greater than $2/3$, the average over the χ_t's is greater than $3\epsilon/4$ (which causes the algorithm to reject). ■

4 Decision Trees

In this section we study the more challenging problem of testing for decision trees over $[0,1]^d$. Given an input $\vec{x} = (x_1, \ldots, x_d)$, the (binary) decision at each node of the tree is whether $x_i \geq a$ for some $i \in \{1, \ldots, d\}$ and $a \in [0,1]$. The labels of the leaves of the decision tree are in $\{0,1\}$. We define the *size* of such a tree to be the number of leaves, and we let DT_s^d denote the class of decision trees of size at most s over $[0,1]^d$. Thus, every tree in DT_s^d determines a partition of the domain $[0,1]^d$ into at most s axis aligned rectangles, each of dimension d (the leaves of the tree), where all points belonging to the same rectangle have the same label.

As in the testing algorithm for interval functions, our algorithm for decision trees will decide whether to accept or reject a function f by pairing "nearby" points, and checking that such pairs have common labels. The naive generalization of the interval function algorithm would consider a "grid" in d-dimensional space with $(s/\epsilon)^d$ cells, each of uniform length in all dimensions. Unfortunately, in

order to observe even a single pair of points that belong to the same grid cell, the size of the sample must be $\Omega(\sqrt{(s/\epsilon)^d})$, which for $d \geq 2$ is linear in s or larger, and thus represents no savings over the sample size for learning.

Instead of considering this very refined and very large grid, our algorithm will instead consider *several* much *coarser* grids. The heart of our proof is a combinatorial argument, which shows that there exists a (not too large) set of (relatively coarse) d-dimensional grids $G_1, \ldots, G_k$ for which the following holds: for every function $f \in \mathrm{DT}_s^d$, there exists a grid G_i such that a "significant" fraction of the cells in G_i "fit inside" the leaves of f — that is, there are not too many cells of G_i that intersect a decision boundary of f.

Theorem 2 *For any size s, dimension d and constant $C \geq 1$, let $s' = s'(s,d,C) \stackrel{\mathrm{def}}{=} 2^{d+1}(2s)^{1+1/C}$. Then the class of decision trees DT_s^d is testable with rejection boundary $\left(\mathrm{DT}_{s'}^d, \frac{1}{2} - \frac{1}{2^{d+5}(Cd)^{d-1}}\right)$ with respect to the uniform distribution in $\tilde{O}\left((2Cd)^{2.5d} \cdot s^{\frac{1}{2}(1+1/C)}\right)$ examples, or $O\left((2Cd)^{2d+1} \cdot \log(s)^{d+1}\right)$ queries. The time sufficient for testing with examples is at most $(2\log(2s))^d$ larger than the number of examples used, and the time for testing with queries is linear in the number of queries performed.*

In order for the sample sizes of Theorem 2 to represent a substantial improvement over those required for learning, we must think of the input dimension d as being a constant. In this case, for a sufficiently large constant C, Theorem 2 says that it is possible to distinguish between the case in which a function is a decision tree of size s, and the case in which the function is a constant distance from any tree of size s' (where s' is not much bigger than s), using only on the order of $\sqrt{s}$ examples or on the order of $\log(s)$ queries. Again, it is easy to verify that $\Omega(s)$ examples or queries are required for learning in any of the standard models.

A possible criticism of the above result is that the distance parameter in the rejection boundary implies that any function that has a significant bias towards either 1 or 0 (and in particular, a biased coin) will pass the test. In Subsection 4.1 we briefly discuss how our testing algorithm can be modified to address this issue.

When queries are allowed we can also obtain the following theorem.

Theorem 3 *For any s and for any ϵ, the class of decision trees DT_s^d is testable with rejection boundary $\left(\mathrm{DT}_{(sd/\epsilon)^d}^d, \epsilon\right)$ and with respect to the uniform distribution with $O(1/\epsilon)$ queries and in time $O(1/\epsilon)$.*

We now turn to prove Theorem 2. The proof of Theorem 3, which is very similar to the proof of Theorem 1, is provided after the proof of Theorem 2.

The combinatorial lemma below is the main tool in proving Theorem 2. We shall need the following notation: For a d-tuple $R = (R_1, \ldots, R_d)$, where $R_j \in [0,1]$, we consider the d-dimensional rectangle whose length (projected) in dimension j is R_j. Thus, in what follows, our notion of a rectangle R is independent of the position of R in space. We let $V(R)$ denote the *volume* of R, so

$$V(R) \;=\; \prod_{j=1}^{d} R_j \;. \tag{4}$$

If Q and R are d-dimensional rectangles, we say that Q *fits* in R if $Q_i \leq R_i$ for all i.

Lemma 4 *Let $R^1, \ldots, R^t$ be rectangles in $[0,1]^d$, each of volume $v \in [0,1]$. Then for any natural number $k \geq d$, there exists a rectangle Q in $[0,1]^d$ such that $V(Q) \geq v^{1+(d-1)/k}$ and Q fits in at least a fraction $k^{-(d-1)}$ of $R^1, \ldots, R^t$.*

Proof: We shall prove the lemma by induction. For $d = 1$ the "rectangles" $R^1, \ldots, R^t$ are simply line segments of length at least v, and so the line segment of length exactly v fits in all of them. Assuming the induction hypothesis holds for $d-1$, we prove it for d. For each rectangle R^i and $1 \leq j \leq d$, we denote by R^i_j the length of R^i in dimension j. By our assumption on the volume of the rectangles, $\prod_{j=1}^{d} R^i_j \geq v$. Let

$$V_{d-1}(R^i) \stackrel{\text{def}}{=} \prod_{j=1}^{d-1} R^i_j \tag{5}$$

be the volume of the projection of R^i to the first $d-1$ dimensions. Thus, for each R^i, $v \leq V_{d-1}(R^i) \leq 1$. Assume, without loss of generality, that $R^1, \ldots, R^t$ are ordered according to $V_{d-1}(R^i)$, so that $V_{d-1}(R^1)$ is largest.

Given a natural number $k \geq d$, we partition the rectangles $R^1, \ldots, R^t$ into k *bins* as follows. For $\ell = 1, \ldots, k$, let $b_\ell = v^{\frac{\ell}{k}}$, and let the ℓth bin, denoted B_ℓ, consist of all rectangles R^i such that $b_\ell \leq V_{d-1}(R^i) < b_{\ell-1}$ (where $b_0 \stackrel{\text{def}}{=} \infty$). Since there are only k bins, there exists a bin, denoted B_g, that contains at least k^{-1} of the rectangles $R^1, \ldots, R^t$. We focus on the rectangles in this bin.

Consider the set, denoted B'_g, of $(d-1)$-dimensional rectangles containing the projections of the rectangles in B_g to the first $d-1$ dimensions. Then, by definition of the bins, each of the $(d-1)$-dimensional rectangles in B'_g has volume at least b_g. Assume without loss of generality that they all have volume *exactly* b_g. [1] By applying the induction hypothesis on the rectangles in B'_g, we have that there exists a rectangle Q' (of dimension $d-1$) that has volume at least $b_g^{1+(d-2)/k}$ and fits in at least $k^{-(d-2)}$ of the rectangles in B'_g.

On the other hand, for each $R^i \in B_g$, we also have that $R^i_d \geq \frac{v}{b_{g-1}}$ (recall that $V(R^i) = v$ and $V_{d-1}(R^i) < b_{g-1}$). Combining this with the above application of the induction hypothesis, we know that there exists a d-dimensional rectangle Q such that

$$V(Q) \geq \frac{v}{b_{g-1}} \cdot b_g^{1+\frac{d-2}{k}} \tag{6}$$

and Q fits in at least $k^{-(d-1)}$ of all rectangles $R^1, \ldots, R^t$. If we now substitute b_{g-1} and b_g in the above lower bound on $V(Q)$, we get that

$$\begin{aligned} V(Q) &\geq \frac{v}{v^{\frac{g-1}{k}}} \cdot \left(v^{\frac{g}{k}}\right)^{1+\frac{d-2}{k}} \\ &= v^{\left(1-\frac{g-1}{k}+\frac{g\cdot(k+d-2)}{k^2}\right)} \\ &= v^{\left(1+\frac{k+g\cdot(d-2)}{k^2}\right)} \end{aligned} \tag{7}$$

which for $g <= k$ (and $v \leq 1$) is at least $v^{1+\frac{d-1}{k}}$ ■

[1] If this is not the case, we can consider a modified set, denoted B''_g, of $(d-1)$-dimensional rectangles, which we define as follows. For each rectangle $R = (R_1, \ldots, R_{d-2}, R_{d-1})$ in B'_g we have a rectangle $R' = (R_1, \ldots, R_{d-2}, R'_{d-1})$ such that $R'_{d-1} = R_{d-1} \cdot b_g / V(R)$ so that R' has volume *exactly* b_g. Clearly, if some $(d-1)$-dimensional rectangle fits in a certain fraction of rectangles in B''_g, then it fits in at least the same fraction in B'_g.

Lemma 4 shows that some "large" rectangle Q must fit inside "relatively many" rectangles in a given collection; this statement ignores the absolute position of the rectangles under consideration. We now translate this to a statement about decision trees, where the rectangles defined by the leaves do in fact have absolute positions. We show that if we now take a "grid" in which every cell is an identical (scaled) copy of Q, then there won't be too many cells that intersect a decision boundary of the decision tree — that is, a non-negligible fraction of the cells are purely labeled by the function.

Lemma 5 *Let f be a decision tree in DT_s^d, and let $R^1, \ldots, R^s$ be the d-dimensional rectangles defined by the leaves of f (ignoring their position in space). Let $\beta \in [0,1]$, and suppose there exists a rectangle $Q = (Q_1, \ldots, Q_d)$ such that the total volume of the rectangles among $R^1, \ldots, R^s$ in which Q fits is at least β. Consider a rectilinear partition G over $[0,1]^d$ where for every $j \in \{1, \ldots, d\}$, each cell in G is of length $Q_j/2$ in dimension j. Then the fraction of grid cells in G that fall entirely inside leaves of f is at least $\beta \cdot 2^{-d}$.*

Proof: For each rectangle R^i, let L^i be the leaf in f to which R^i corresponds. We say that R^i (respectively, L^i) is *good with respect to* Q if Q fits inside R^i. We shall show that for each good leaf L^i the total volume of all grid cells that fit inside it is at least $2^{-d} \cdot V(R^i)$. Since the total volume of all good leaves is at least β, the lemma follows.

Consider any good rectangle $R^i = (R_1^i, \ldots, R_d^i)$ and the corresponding leaf L^i. For each dimension j, let $r_j^i \stackrel{\text{def}}{=} R_j^i / Q_j$. Hence, by definition,

$$V(R^i) \;=\; V(Q) \cdot \prod_j r_j^i \; . \tag{8}$$

Let $\bar{R}^i = (\bar{R}_1^i, \ldots, \bar{R}_d^i)$ be the d-dimensional rectangle defined by the grid cells of G that fit inside L^i. Since the grid cells have length at most $Q_j/2$ in each dimension j, we have that

$$\bar{R}_j^i \;\geq\; R_j^i - Q_j/2 \tag{9}$$

and so

$$V(\bar{R}^i) \;\geq\; V(Q) \cdot \prod_j (r_j^i - 1/2) \tag{10}$$

Therefore

$$V(\bar{R}^i)/V(R^i) \;\geq\; \prod_j (r_j^i - 1/2)/r_j^i \; . \tag{11}$$

Since Q fits inside R^i, $r_j^i \geq 1$ for each j, and so $V(\bar{R}^i)/V(R^i) \geq 2^{-d}$, as claimed. ■

Proof of Theorem 2: We prove the correctness of the algorithm described in Figure 1. As in the proof of Theorem 1, we first show that based on our choice of the sample size m, with probability at least $5/6$, for *every* grid, the number of pairs that belong to a common grid cell, is at least m'. Details follow. As in the proof of Theorem 1, we break the sample into $4m'$ random samples each of size $2\sqrt{s'}$ and consider only close pairs that belong to the same subsample. By the same argument used in that proof, if we now fix a particular grid, the probability that any given subsample contains a pair of points from a common cell, is at $1/2$. By a multiplicative Chernoff bound, the probability that less than a fourth of the subsamples (that is, less than m') contribute such a pair, is at most $\exp(-m'/12)$. Since

Testing Algorithm for Decision Trees

Input: size s, dimension d, and a constant C.

1. For Examples Version of the algorithm let S be a uniformly chosen sample of size

$$m = O(2^{3d/2}(Cd)^{2d+1}(2s)^{\frac{1}{2}(1+1/C)} \cdot \log\log(s)).$$

2. Let $n = \lceil (1+1/C) \cdot \log(2s) \rceil$. For each setting of $(i_1, \ldots, i_d)$ such that the i_j's are integers ranging between 1 and n, and $\sum i_j = n$, consider the *grid* $G = G(i_1, \ldots, i_d)$ over $[0,1]^d$ whose grid cells have length $2^{-(i_j+1)}$ in dimension j and do the following:

   - Query Version

   (a) Uniformly select $m' = O((2Cd)^{2d+1} \cdot \log\log(s))$ grid cells in G, and for each cell chosen, uniformly select a pair of points in the cell and query f on these points.

   (b) If the fraction of selected pairs that have the same label is at least $\frac{1}{2} + \frac{1}{2^{d+4}(Cd)^{(d-1)}}$, then ACCEPT.

   - Examples Version

   (a) For each cell c in G, let $x_1^c, \ldots, x_t^c$ be the examples in the sample S that belong to c (according to their order in S). For each $1 \leq i \leq \lfloor t/2 \rfloor$, the pair of points x_{2i-1}^c and x_{2i}^c, are referred to as a *close pair*.

   (b) If the fraction of close pairs (among all close pairs in all cells) that have the same label is at least $\frac{1}{2} + \frac{1}{2^{d+4}(Cd)^{(d-1)}}$, then ACCEPT.

3. If no grid caused to ACCEPT then REJECT.

Figure 1: Testing Algorithm for Decision Trees

the number of grids is bounded by $n^d < (2\log(2s))^d$ (while $m' = \Omega(d \log\log s)$, the probability that for some grid there are less than m' pairs of points from a common cell, is bounded by a small constant as required. It thus follows that we can restrict our attention to analyzing the query version of the algorithm.

The proof that for any f such that $\mathrm{dist}(f, \mathrm{DT}_{s'}^d) > \frac{1}{2} - \frac{1}{2^{d+5}(Cd)^{(d-1)}}$, the algorithm rejects f with probability at least $2/3$, (or, equivalently, that any function that is accepted with probability greater than $1/3$ is $(\frac{1}{2} - \frac{1}{2^{d+5}(Cd)^{(d-1)}})$-close to $\mathrm{DT}_{s'}^d$), is analogous to the special case of the interval functions proved in Theorem 1. In particular, if f is accepted with probability greater than $1/3$ then there must be at least one grid $G = G(i_1, \ldots, i_d)$ that, when considered, causes the algorithm to accept with probability greater than $1/(3k)$, where $k < n^d < (2\log(2s))^d$ is the total number of grids considered. Let f' be the decision tree (of size s') whose leaves correspond to the cells of G, and where the label of each leaf is the majority label according to f. We define χ_t and ϵ_j analogously to the way they were defined in the proof of Theorem 1. Namely, χ_t is a 0/1 random variable that is 1 if an only if the t'th pair of points (uniformly selected in a uniformly chosen grid cell of G) have a different label; $\epsilon_j \in [0, 1/2]$ is the probability mass of points in j'th cell of G that have the minority label according

to f. Then we get that

$$\mathrm{E}[\chi_t] \;=\; \mathrm{E}_j[2\epsilon_j(1-\epsilon_j)] \;\geq\; \mathrm{E}_j[\epsilon_j] \;=\; \mathrm{dist}(f,f') \tag{12}$$

Since f is accepted with probability greater than $1/(3k)$ when the grid G is considered, then by applying an additive Chernoff bound (and our choice of m'), it follows that

$$\mathrm{dist}(f,f') \;\leq\; \frac{1}{2} - \frac{1}{2^{d+5}(Cd)^{(d-1)}} \,. \tag{13}$$

In the remainder of the proof we analyze the case in which the function f is in fact a decision tree in DT_s^d. We show that for each $f \in \mathrm{DT}_s^d$, there exists a grid $\hat{G} = G(i_1,\ldots,i_d)$, such that fraction of grid cells that fits inside leaves of f is at least $2^{-(d+2)} \cdot (Cd)^{-d}$. We prove the existence of $\hat{G}$ momentarily, but first show how this the proof that f is accepted with probability at least $2/3$ follows from the existence of $\hat{G}$. If we define χ_t as above (with respect to $\hat{G}$), then similarly to the proof of Theorem 1, we have that

$$\begin{aligned}\mathrm{E}[\chi_t] \;&\leq\; \left(1 - 2^{-(d+2)} \cdot (Cd)^{-d}\right) \cdot \frac{1}{2} \\ &=\; \frac{1}{2} - 2^{-(d+3)} \cdot (Cd)^{-d} \,.\end{aligned} \tag{14}$$

By an additive Chernoff bound (and our choice of m'), the function f is accepted with probability at least $2/3$ as required. We now turn to proving the existence of $\hat{G}$.

Let $R^1,\ldots,R^s$ be the rectangles corresponding to the leaves of f. Consider those rectangles among $R^1,\ldots,R^s$ that have volume at least $1/(2s)$, and assume, without loss of generality that these are $R^1,\ldots,R^t$. Thus, the total volume of $R^1,\ldots,R^t$ is at least $1/2$. We would like to apply Lemma 4 to these rectangles; however, they do not all have exactly the same volume. We therefore "cut them up" into rectangles of volume exactly $1/(2s)$. More precisely, for each rectangle $R^i = (R^i_1,\ldots,R^i_d)$ such that $V(R^i) \geq 1/(2s)$, let

$$r^i \;\stackrel{\mathrm{def}}{=}\; \lfloor V(R^i)/(1/(2s)) \rfloor \tag{15}$$

be the number of (whole) rectangles with volume $1/(2s)$ that can fit ("side-by-side") in R^i. For each $1 \leq \ell \leq r^i$, let $R^{i,\ell} = (R^i_1,\ldots,\bar{R}^i_d)$, where

$$\bar{R}^i_d \;\stackrel{\mathrm{def}}{=}\; (1/(2s))/\prod_{j=1}^{d-1} R^i_j \,. \tag{16}$$

Thus, for each ℓ, $V(R^{i,\ell}) = 1/(2s)$, and $R^{i,1},\ldots,R^{i,r^i}$ can all fit "side-by-side" in R^i (with a possible "left-over" smaller rectangle). The rectangles

$$R^{1,1},\ldots,R^{1,r^1},\ldots,R^{t,1},\ldots,R^{t,r^t}$$

all have volume exactly $1/(2s)$, and their total volume is at least $1/4$.

Suppose we now apply Lemma 4 to these (at most $2s$) rectangles, setting $k = C \cdot d$. Then, by the lemma, there exists a rectangle $Q = (Q_1,\ldots,Q_d)$, that has volume at least $(1/(2s))^{1+1/C}$ and fits inside at least a fraction $(Cd)^{-(d-1)}$ of $R^{1,1},\ldots,R^{t,r^t}$. Recall that $R^{1,1},\ldots,R^{t,r^t}$ are simply

sub-rectangles of a subset of $R^1, \ldots, R^s$, their total volume is at least $1/4$, and they all have equal volume. Therefore, the total volume of the rectangles among $R^1, \ldots, R^s$ into which Q fits is at least $\frac{1}{4} \cdot (Cd)^{-(d-1)}$. Since

$$V(Q) \;\geq\; (1/2s)^{-(1+1/C)} \;\geq\; 2^{-n} \tag{17}$$

(where n is as set in Step (2) of the algorithm), there must be at least one iteration of the algorithm in which the grid $G(i_1, \ldots, i_d)$ has cells with length at most $Q_j/2$ in each dimension j. Let us denote this grid by $\hat{G}$. By applying Lemma 5 we obtain that the fraction of grid cells (in $\hat{G}$) that fit inside leaves of f is at least $2^{-(d+2)} \cdot (Cd)^{-(d-1)}$. ■

Proof of Theorem 3: The algorithm on which the correctness of this theorem is based, is a generalization of the query version of the algorithm for testing interval functions. The algorithm considers a single grid G, whose cells have length $\epsilon/(sd)$ in each dimension. It then uniformly chooses $m' = O(1/\epsilon)$ grid cells, uniformly selects two points in each chosen cell, and accepts only if the fraction of pairs selected by the algorithm that have the same label is at least $1 - 3\epsilon/4$.

CASE 1 ($f \in \mathrm{DT}^d_s$). We show that in this case, the fraction of grid cells in G that fit inside leaves of f is at least $1 - \epsilon$. Similarly to the argument made in the proof of Theorem 2, this implies that the expected fraction of pairs that have the same label is at least $1 - \epsilon/2$, and so with probability at last $2/3$ the fraction is at least $1 - 3\epsilon/4$ as desired. To obtain the bound on the fraction of grid cells that fit inside leaves of f, consider any axis-aligned hyperplane (of $d-1$ dimensions), which separates two leaves of f. Each such hyperplane is either aligned with the grid, or it intersects $(sd/\epsilon)^{d-1}$ grid cells. Since the number of such separating hyperplanes is at most $s \cdot d$, while the number of grid cells is $((sd)/\epsilon)^d$, the fraction of grid cells that are intersected by leaves of f is at most ϵ, as claimed above.

CASE 2 ($\mathrm{dist}(f, \mathrm{DT}^d_{(sd/\epsilon)^d}) > \epsilon$). This case follows analogously to Case 2 in the proof of Theorem 1. Here we have that for any function f that is accepted with probability greater than $1/3$, the function $g \in \mathrm{DT}_{(sd/\epsilon)^d}$ whose leaves correspond to the cells of the grid G and the label of each leaf (cell) is the majority label according to f, is at distance at most ϵ from f.

4.1 Biased Functions

As noted previously, Theorem 2 implies that any function that has a significant enough bias towards either 1 or 0 (and in particular, a biased coin) will pass the test with high probability. The following theorem is a refinement of Theorem 2, and can be used to address this issue (as discussed below).

Theorem 6 *For any constant $\gamma \in [0, 1/2]$, let $\mathrm{DT}^d_{s,\gamma}$ be the class of decision tree over $[0,1]^d$ that have size at most s and* bias *at least γ. That is, for every function $f \in \mathrm{DT}^d_{s,\gamma}$, either $\Pr[f(x) = 1] \geq \frac{1}{2} + \gamma$, or $\Pr[f(x) = 0] \geq \frac{1}{2} + \gamma$. Then for any γ, the class $\mathrm{DT}^d_{s,\gamma}$ is testable with rejection boundary $\left(\mathrm{DT}^d_{s'}, \frac{1}{2} - \gamma^2 - \frac{1-2\gamma}{2^{d+5}(Cd)^{d-1}}\right)$, where s' is as set in Theorem 2 (with the same sample/query complexity).*

Thus, in order to avoid accepting any function f that is biased away from $1/2$, the algorithm can first approximate the bias of f and then run the testing algorithm for $\mathrm{DT}^d_{s,\gamma}$ with the appropriate γ.

Proof: The testing algorithm for $\mathrm{DT}^d_{s,\gamma}$ is the same as the testing algorithm described in Figure 1, except that the acceptance criterion is slightly altered: a function is accepted only if the fraction of

pairs with the same label is at least $\frac{1}{2} + \gamma^2 - \frac{1-\gamma}{2^{d+4}(Cd)^{d-1}}$. Showing that every f that is beyond the rejection boundary is in fact rejected (or more precisely, proving the contrapositive statement), is done as in Theorem 2.

We now prove that if $f \in \mathrm{DT}_d^{s,\gamma}$ then it is accepted with probability at least $2/3$. Here too it suffices to prove this holds for the query version of the algorithm. As shown in the proof of Theorem 2, there exists a grid $\hat{G} = G(i_1, \ldots, i_d)$, such that fraction of grid cells that fit inside leaves of f is at least $2^{-(d+2)} \cdot (Cd)^{-d}$. We refer to these cells as *pure* and to the rest as *mixed*. Let N denote the number of cells in $\hat{G}$, and let $\alpha = 2^{-(d+2)} \cdot (Cd)^{-d}$. Assume, without loss of generality, that $\Pr[f(x) = 1] \geq \frac{1}{2} + \gamma$, and let $1/2 + \gamma_j$ be the probability that $f(x) = 1$ for x in grid cell j (so that γ_j can be negative). Then

$$\frac{1}{N} \sum_j \left(\frac{1}{2} + \gamma_j \right) \;\; \geq \;\; \frac{1}{2} + \gamma \; . \tag{18}$$

Let χ_t be defined as in the proof of Theorem 2. Then,

$$\begin{aligned} \mathrm{E}[\chi_t] \;\; &= \;\; \frac{1}{N} \cdot \sum_j 2 \left(\frac{1}{2} + \gamma_j \right) \left(\frac{1}{2} - \gamma_j \right) \\ &\leq \;\; \frac{1}{2} - \frac{1}{N} \cdot 2 \sum_j \gamma_j^2 \; . \end{aligned} \tag{19}$$

In order to upper bound the above expectation, we lower bound the term $\frac{1}{N} \cdot 2 \sum_j \gamma_j^2$. We know that the pure cells ($\gamma_j = 1/2$ or $\gamma_j = -1/2$), contribute a total of at least $\alpha/2$ to this term, and we would like to lower bound the contribution of the mixed cells whose set we denote M.

Let α_1 be the fraction of cells that are both pure and labeled 1 by f. Then by breaking the summation in Equation (18) into two parts we get that,

$$\alpha_1 + \frac{1}{N} \cdot \sum_{j \in M} \left(\frac{1}{2} + \gamma_j \right) \;\; \geq \;\; \frac{1}{2} + \gamma \; . \tag{20}$$

Since the fraction of mixed cells is at most $1 - \alpha$ (and $\alpha_1 \leq \alpha$), Equation (20) implies that

$$\frac{1}{N} \sum_{j \in M} \gamma_j \;\; \geq \;\; \gamma - \frac{\alpha}{2} \; . \tag{21}$$

Recall that we are interested in lower bounding $\frac{1}{N} \cdot 2 \sum_j \gamma_j^2$. This sum is minimized when all γ_j's are equal. Substituting in Equation (19) we obtain

$$\mathrm{E}[\chi_t] \;\; \leq \;\; \frac{1}{2} - \gamma^2 - \frac{\alpha}{2} \cdot (1 - \gamma) \; . \tag{22}$$

By substituting the value of α and applying an additive Chernoff bound we get that f is accepted with probability at least $2/3$. ■

5 Aligned Voting Networks

In this section we study a restricted class of neural networks over $[0, 1]^d$ called *Aligned Voting Networks*. These are essentially neural networks in which the hyperplane defining each hidden unit is constrained to be parallel to some coordinate axis, and the output unit takes a majority vote of the hidden units.

Definition 5 *An* aligned hyperplane *over* $[0,1]^d$ *is a function* $h : [0,1]^d \to \{+1,-1\}$ *of the form* $h(\vec{x}) = sign(x_i - a)$ *for some dimension* $i \in \{1,\ldots,d\}$ *and some* $a \in [-1,1]$ *(where* $sign(0)$ *is defined to be* $+1$*). An* aligned voting network *over* $[0,1]^d$ *is a function* $f : [0,1]^d \to \{+1,-1\}$ *of the form*

$$f(\vec{x}) = sign\left(\sum_{j=1}^{s} h_j(\vec{x})\right)$$

where each $h_j(\vec{x})$ *is an aligned hyperplane over* $[0,1]^d$*. The* size *of* f *is the number of voting hyperplanes* s*.*

An alternative way of viewing an aligned voting network f is as a constrained labeling of the cells of a rectilinear partition of $[0,1]^d$. For each dimension i, we have *positions* $a_i^j \in [0,1]$ and *orientations* $u_i^j \in \{+1,-1\}$. The hyperplanes $x_i = a_i^j$ define the rectilinear partition, and f is constant over each cell c: for any $\vec{x}$, we define

$$\#(\vec{x}) = \sum_{i=1}^{d}\sum_{j=1}^{s_i} sign(x_i - u_i^j a_i^j)$$

(where s_i is the number of aligned hyperplanes that project on dimension i), and then $f(\vec{x}) = sign(\#(\vec{x}))$. By extension, for each cell c of the partition, we define $\#(c)$ as the constant value of $\#(\vec{x})$ for all $\vec{x} \in c$, and $f(c)$ as the constant value of $f(\vec{x})$ for all $\vec{x} \in c$.

A decision tree of size s defines a partition of space into only s cells, each of which may be labeled arbitrarily. An aligned voting network of size s also naturally defines a partition of space, but into many more cells, on the order of s^d. Indeed, already in 3 dimensions, if $s/3$ of the aligned hyperplanes project into each dimension, the rectilinear partition defined by these hyperplanes has $(s/3)^3$ cells, and waiting for two points to fall in a common cell will take more than s examples. Instead, we will exploit the fact that the labels of the cells in this partition are far from arbitrary, but are instead determined by the vote over the hyperplanes. It will turn out that if instead of considering two points to be near only if they fall in the same *cell* of the partition, we consider them to be near even if they fall in the same *slice* of the partition (where a slice contains *all* the cells sharing some fixed range of values for a single dimension), we can obtain the desired balance: with a number of examples sublinear in s, we can get a near pair, and the chances that such a pair is purely labeled is significantly greater than $1/2$.

Theorem 7 *Let* AVN_s^d *denote the class of aligned voting networks of size at most* s *over* $[0,1]^d$*. Then for any* s *and* d*,* AVN_s^d *is testable with rejection boundary* $(\mathrm{AVN}^d_{2\cdot 6^{2^{d+1}}s}, \frac{1}{2} - \frac{1}{6^{2^{d+3}}})$ *with respect to the uniform distribution in* $O(6^{2^{d+4}}\sqrt{s})$ *examples (and time), or* $O(6^{2^{d+4}})$ *queries (and time).*

Again, the theorem is interesting in comparison to the resources required for standard learning only if d is a constant with respect to s. Along the lines of Theorem 6, here too we can slightly modify the algorithm so that it will not automatically accept biased random functions.

Before giving the proof of Theorem 7, let us make some simple but useful technical observations about aligned voting networks. For a given network $f \in \mathrm{AVN}_s^d$ defined by hyperplanes $\{a_i^j\}$ and orientations $\{u_i^j\}$, we define $slice(i,j)$ to consist of all those partition cells in which the ith component falls between a_i^j and a_i^{j+1} (that is, the cells falling between the jth and $j+1$st aligned hyperplanes in dimension i). Note that in going from $slice(i,j)$ to $slice(i,j+1)$, either the count $\#(c)$ of every cell c increases by 2, or the count of every cell decreases by 2 (depending on the orientation u_i^j), since the

only change is with respect to a_i^j. This implies that for any i, and for any j and j', the *ordering* of the cells by their counts is *preserved* between the parallel $slice(i,j)$ and $slice(i,j')$. This leads to a simple but important property that we call the *continuation* property: if c has the ℓth largest count in $slice(i,j)$, and at least ℓ of the cells have positive counts (and thus, f is $+1$ on them), then the projection c' of c into any parallel $slice(i,j')$ containing at least ℓ positive counts will also satisfy $f(c') = +1$. The following combinatorial lemma, which is central to our proof, exploits this property.

Lemma 8 *Let f be an aligned voting network over $[0,1]^d$ of size at most s, and let $\gamma_d = 1/(6^{2^{d+1}})$. If*

$$\Pr_{[0,1]^d}[f(\vec{x}) = +1] \;\geq\; \frac{1}{2} - \gamma_d \tag{23}$$

then there exists a dimension i such that the total probability mass (with respect to the uniform distribution) of the $slice(i,j)$ of f for which

$$P_i^j \;=\; \Pr_{slice(i,j)}[f(\vec{x}) = +1] \;\geq\; \frac{1}{2} + \gamma_d \tag{24}$$

is at least $2\gamma_d$. (Note that P_i^j is simply the "positive bias" of f on a random point from $slice(i,j)$.) Thus, as long as an aligned voting network is not significantly biased away *from $+1$, the probability mass of the slices on which the network in fact has a significant bias* towards $+1$ *is non-negligible.*

In fact, the lemma remains true (with a slightly different setting of γ_d) if we exchange $1/2$ by some $p \geq 1/2$. However, in our application, the worst case is when $p = 1/2$.

Proof: For $i \in \{1,\ldots,d\}$ let $a_i^1 < \ldots < a_i^{s_i}$ be the aligned hyperplanes of f. We prove the claim by induction on d. For the base case $d = 1$, we have at most s intervals of $[0,1]$ each labeled either $+1$ or -1. If the overall probability that f is $+1$ is at least $(1/2) - \gamma_1$, then in fact the total probability mass of the intervals labeled $+1$ is $(1/2) - \gamma_1$. Solving $2\gamma_1 = (1/2) - \gamma_1$ yields $\gamma_1 = 1/6$.

Now suppose the claim holds for every $d' < d$, and assume that the probability that f is $+1$ over $[0,1]^d$ is at least $(1/2) - \gamma_d$. Let $0 \leq \alpha_H \leq r$ (the subscript H stands for "high bias") denote the total probability mass of all $slice(d,j)$ satisfying $P_d^j \geq (1/2) + \gamma_d$, and let α_L be the total probability mass of $slice(d,j)$ satisfying $P_d^j \leq (1/2) - \gamma_{d-1}$ ("low bias"). Then we have that

$$\alpha_L\left((1/2) - \gamma_{d-1}\right) + (1 - \alpha_L - \alpha_H)\left((1/2) + \gamma_d\right) + \alpha_H \cdot 1 \;\geq\; (1/2) - \gamma_d \;\; . \tag{25}$$

From this we obtain

$$\alpha_H \;\geq\; \frac{1}{(1/2) - \gamma_d}\left((\gamma_d + \gamma_{d-1})\alpha_L - 2\gamma_d\right) \;\; . \tag{26}$$

If α_L satisfies

$$\alpha_L \;\geq\; \frac{((1/2) - \gamma_d)2\gamma_d + 2\gamma_d}{\gamma_d + \gamma_{d-1}} \tag{27}$$

then we have $\alpha_H \geq 2\gamma_d$, as desired.

Otherwise, let k be the index j that satisfies $P_d^j \geq (1/2) - \gamma_{d-1}$ while minimizing P_d^j; thus, $slice(d,k)$ is the slice that "comes closest" to being low bias without actually being low bias. Note that f restricted to $slice(d,k)$ meets the inductive hypothesis for $d-1$ dimensions; thus $slice(d,k)$

must contain "subslices" (in which now both x_d ranges between a_d^k and a_d^{k+1} and for some other dimension $d' < d$, and some k', $x_{d'}$ is between $a_{d'}^{k'}$ and $a_{d'}^{k'+1}$) whose relative probability (with respect to $slice(d,k)$) is at least γ_{d-1} and whose positive $(+1)$ bias exceeds $(1/2)+\gamma_{d-1}$ (that is, the probability that f is $+1$ exceeds $(1/2)+\gamma_{d-1}$ in each of these subslices). Since $slice(d,k)$ was chosen to have minimum positive bias among all the $slice(d,j)$ of positive bias at least $(1/2)-\gamma_{d-1}$, by the continuation property, if c is a cell of $slice(d,k)$ that is positively labeled by f, then the projection of c into any of these parallel $slice(d,j)$ must also be positively labeled by f. In other words, we may take each positive cell in the biased subslices of $slice(d,k)$, and when we project each such cell along dimension d, it remains positive.

Since the total probability mass of slices (along dimension d) having bias at least $(1/2)-\gamma_{d-1}$ is at least $1-\alpha_L$, we obtain that the total probability of slices along dimension d', that have positive bias at least $((1/2)+\gamma_{d-1})(1-\alpha_L)$, is at least $2\gamma_{d-1}$. For this bias to be at least $(1/2)+\gamma_d$ we must have

$$\alpha_L \leq \frac{\gamma_{d-1}-\gamma_d}{(1/2)+\gamma_{d-1}} \tag{28}$$

and in addition we need $\gamma_{d-1} \geq \gamma_d$.

Returning to the earlier constraint on α_L given by Equation (27), we find that Equations (27) and (28) can both be satisfied provided

$$\frac{((1/2)-\gamma_d)2\gamma_d+2\gamma_d}{\gamma_d+\gamma_{d-1}} \leq \frac{\gamma_{d-1}-\gamma_d}{(1/2)+\gamma_{d-1}} \tag{29}$$

First note that

$$\frac{((1/2)-\gamma_d)2\gamma_d+2\gamma_d}{\gamma_d+\gamma_{d-1}} \leq \frac{(1/2)2\gamma_d+2\gamma_d}{\gamma_{d-1}} = \frac{3\gamma_d}{\gamma_{d-1}} \tag{30}$$

and

$$\frac{\gamma_{d-1}-\gamma_d}{(1/2)+\gamma_{d-1}} \geq \gamma_{d-1}-\gamma_d \geq \gamma_{d-1}/2 \tag{31}$$

(where the last inequality holds whenever $\gamma_d < \gamma_{d-1}/2$, which holds in our case assuming $\gamma_1 \leq 1/2$), so it suffices to enforce that $3\gamma_d/\gamma_{d-1} \leq \gamma_{d-1}/2$ or equivalently $\gamma_d \leq \gamma_{d-1}^2/6$. Thus we obtain the constraint $\gamma_d \leq \gamma_1^{2^d}/6^d$, which is satisfied by the choice $\gamma_d = 1/(6^{2^{d+1}})$ given in the lemma. ■

The corollary below follows directly from Lemma 8.

Corollary 9 *Let f be an aligned voting network over $[0,1]^d$ of size s, and let $\gamma_d = 1/(6^{2^{d+1}})$. For $i \in \{1,\ldots,d\}$ and $j \in \{0,\ldots,2\cdot s\cdot\gamma_d^{-1}-1\}$, let $b_i^j = j\cdot\gamma_d/(2\cdot s)$, and let $slice(i,j)$ be the slice between the hyperplanes $x_i = b_i^j$ and $x_i = b_i^{j+1}$. If $\Pr_{[0,1]^d}[f(\vec{x})=+1] \geq (1/2)-\gamma_d$ then there exists a dimension i such that the total probability mass (with respect to the uniform distribution) of the $slice(i,j)$ for which $\Pr_{slice(i,j)}[f(\vec{x})=+1] \geq (1/2)+\gamma_d$ is at least γ_d.*

All that is missing for the proof of Theorem 7 is to show that any function defined by an arbitrary labeling of some s' parallel slices (determined by a set of axis aligned parallel hyperplanes) can be computed by an aligned voting network of size s'.

Lemma 10 *For any integer s', dimension $i \in \{1,\ldots,d\}$, and values $\{b^j\}_{j=1}^{s'}$, consider a partition of $[0,1]^d$ into slices defined by the hyperplanes $x_i = b^j$. Then for any $\{+1,-1\}$ labeling of these slices, there exists an aligned voting network g of size at most s' that is consistent with this labeling.*

Proof: Consider only those hyperplanes $b^{j_1} \ldots, b^{j_t}$ that separate slices that have difference labels, and let $a^1 = b^{j_1}, \ldots, a^t = b^{j_t}$ be the aligned hyperplanes of g. Assume, without loss of generality, that the "last" slice defined by $a^t \leq x_i \leq 1$ is labeled $+1$. Let the orientation of the hyperplanes alternate so that $u^t = +1$, $u^{t-1} = -1$ and so on. If t is even, then for the last slice c, we have $\#(c) = 0$, and otherwise $\#(c) = +1$. In either case, the sign is $+1$ as required. It is easy to verify that for every other slice c' that should be labeled $+1$, $\#(c') = 0$ (respectively, $\#(c') = +1$) while for every slice c' that should be labeled -1, $\#(c') = -2$ (respectively, $\#(c') = -1$). The lemma follows. ■

Testing Algorithm for Aligned Voting Networks

Input: size s, dimension d.

1. Let $\gamma_d = 1/(6^{2^{d+1}})$. Using $O(1/\gamma_d^2)$ uniformly chosen examples, approximate the bias of the function: If the fraction of examples labeled $+1$ (similarly, -1) is at least $\frac{1}{2} + \frac{3}{4}\gamma_d$, then ACCEPT.

2. Otherwise, for $i \in \{1, \ldots, d\}$ and $j \in \{0, \ldots, 2 \cdot \gamma_d^{-1} \cdot s - 1\}$, let $b_i^j = j \cdot \gamma_d/(2 \cdot s)$, and let $slice(i, j)$ be the slice between the hyperplanes $x_i = b_i^j$ and $x_i = b_i^{j+1}$.

3. - Queries Version: Uniformly select $O(1/\gamma_d^6)$ slices and query f on a uniformly selected pair in the slice. If the fraction of pairs with the same label is at least $\frac{1}{2} + \gamma_d^3$, then ACCEPT. Otherwise, REJECT.
   - Examples Version: Uniformly select $O(\gamma_d^{-6}\sqrt{s})$ examples. For each $slice(i, j)$, let $x_1^{i,j}, \ldots, x_t^{i,j}$ be the examples in the sample that belong to $slice(i, j)$. For each $1 \leq k \leq \lfloor t/2 \rfloor$, the pair of points $x_{2k-1}^{i,j}$ and $x_{2k}^{i,j}$, are referred to as a *close pair*. If the fraction of close pairs with the same label is at least $\frac{1}{2} + \gamma_d^3$, then ACCEPT. Otherwise, REJECT.

Figure 2: Testing Algorithm for Aligned Voting Networks

Proof of Theorem 7: As argued in the proof of Theorem 1 (using the Birthday Paradox) it suffices to prove the correctness of the query version of the algorithm. Consider first the case where $f \in \mathrm{AVN}_s^d$. If $\Pr_{[0,1]^d}[f(\vec{x}) = +1] < (1/2) - \gamma_d$ (so that the probability that the value of the function is -1 is greater than $(1/2) + \gamma_d$), then with high probability f is accepted in Step 1 of the algorithm. Otherwise we may apply Corollary 9 to obtain that there exists a dimension i such that the total probability mass (with respect to the uniform distribution) of the $slice(i, j)$ for which $\Pr_{slice(i,j)}[f(\vec{x}) = +1] \geq (1/2) + \gamma_d$ is at least γ_d. For any such slice, the probability that two points selected uniformly in that slice have a different label is at most

$$2\left(\frac{1}{2} - \gamma_d\right)\left(\frac{1}{2} + \gamma_d\right) \;=\; \frac{1}{2} - 2\gamma_d^2 \,. \tag{32}$$

For any other slice, the probability of this event is at most $\frac{1}{2}$. Let $m' = O(1/\gamma_d^6)$ be the number of pairs of points uniformly selected by the algorithm, and for $t = 1, \ldots, m'$ let χ_t be a 0/1 random variable that is 1 if an only if the t'th pair of points have a different label. Then,

$$\mathrm{E}[\chi_t] \;\leq\; \gamma_d \cdot \left(\frac{1}{2} - 2\gamma_d^2\right) + (1 - \gamma_d) \cdot \frac{1}{2} \;=\; \frac{1}{2} - 2\gamma_d^3 \,. \tag{33}$$

By an additive Chernoff bound, with probability at least $2/3$, the average value of the χ_t's measured by the algorithm is at most $\frac{1}{2} - \gamma_d^3$, causing the algorithm to accept.

We now turn to the case that f is beyond the rejection boundary. To show that any such f is rejected with probability at least $2/3$ we again prove the contrapositive. Let f be a function that is accepted with probability greater than $1/3$. It follows that either f is accepted with probability greater that $1/6$ in the first step of the algorithm or is accepted with probability greater that $1/6$ in the third step. In the former case, it must have bias of at least $\frac{1}{2} + \frac{1}{2}\gamma_d$, and so it is $(\frac{1}{2} - \frac{1}{2}\gamma_d)$-close to the "trivial" network that has constant value on the whole domain. In the latter case consider the network f' defined by the slices considered in the algorithm, where the label of each slice is the majority label according to f. The existence of such a network follows from Lemma 10. If we define χ_t as above, and let $\epsilon_{i,j} \in [0, 1/2]$ be the probability mass of points in $slice(i, j)$ that have the minority value according to f, then we have (analogously to the proofs of Theorem 1 and Theorem 2),

$$\mathrm{E}[\chi_t] \;\geq\; \mathrm{E}_{i,j}[\epsilon_{i,j}] \;=\; \mathrm{dist}(f, f') \tag{34}$$

and the claim follows by applying an additive Chernoff bound. ■

6 A Generalization

The common theme of the results presented in the previous sections is that in all cases, we showed that the class H_s we would like to test can be "approximated by" a bounded number of fixed partitions of the domain. More precisely, if we consider the family of functions F defined by these partitions (when we allow all labelings of the cells of the partitions), then for every function in H_s there exists a function in F that approximates it. Furthermore, these partition functions can be implemented by a class $H_{s'}$ where $s' \geq s$. The algorithms described essentially performed the same task: for each fixed partition, pairs of points that belong to a common cell of the partition were considered, and if there was a sufficiently strong bias among these pairs towards having a common label, then the tested function was accepted. The following theorem formalizes this unifying view.

Theorem 11 *Let H and $H' \supseteq H$ be classes of functions over domain X and let $\mathcal{P} = \{\mathcal{P}^1, \ldots, \mathcal{P}^k\}$ be a set of partitions over X, where each $\mathcal{P}^i$ consists of s' equal-size components. Let $\mathrm{PAR}_{\mathcal{P}}$ be the class of all functions g, such that there exists a partition $\mathcal{P}^i = (X_1^i, \ldots, X_{s'}^i)$ in $\mathcal{P}$ and a vector $\vec{b} \in \{0,1\}^s$ such that for each $j \in \{1, \ldots, s'\}$, and for each $x \in X_j^i$, $g(x) = b_j$. Suppose that:*

1. *$\mathrm{PAR}_{\mathcal{P}} \subseteq H'$;*
2. *There exists $\alpha \in [0, 1]$ and $\beta \in [0, 1/2]$ such that for each $f \in H$, there is a partition $\mathcal{P}^i$ such that on at least a fraction α of the component X_j^i of $\mathcal{P}^i$, for some $b_j \in \{0, 1\}$, $\mathrm{Pr}_{x \in X_j^i}[f(x) = b_j] \geq 1/2 + \beta$.*

Then for any $\delta \in (0, 1/2]$, H is testable with rejection boundary $(H', 1/2 - (2\alpha\beta^2 - \delta))$ and with respect to the uniform distribution in $O(\log k \cdot \sqrt{s'}/\delta^2)$ examples or $O(k \log k/\delta^2)$ queries. The running time of the examples version of the algorithm is an order of k times the sample size, and the running time of the query version is linear in the number of queries.

We note that while the bound on the running time of the query version of the algorithm is always strictly smaller than the bound on the running time of the examples version, the bound on the query complexity is smaller than the bound on the sample complexity only when k is small relative to s'. In other words, when k is relatively large, we do not know of a way to exploit the power of the queries, and we are better off using the examples version of the algorithm (that is, asking uniformly chosen queries).

Classes of Partition Functions. A special case of the above theorem applies to classes of functions that are defined by all possible labelings of a fixed partition (or fixed set of partitions). In this case we can get the following stronger result.

Theorem 12 *Let $\mathcal{P} = \{\mathcal{P}^1, \ldots, \mathcal{P}^k\}$ be a set of partitions over a domain X, where each $\mathcal{P}^i$ consists of at most s components. Then, for any $\epsilon \in (0, 1/2]$, the class $\mathrm{PAR}_{\mathcal{P}}$ (as defined in Theorem 11) is testable with rejection boundary $(\mathrm{PAR}_{\mathcal{P}}, \epsilon)$ and with respect to any distribution in $O(\log k \cdot \sqrt{s}/\epsilon)$ examples, and $O(k \cdot \log k \cdot \sqrt{s}/\epsilon)$ time. In case the underlying distribution is such that its restriction to any part of the domain is efficiently sampleable, then testing can be performed using $O(k \cdot \log k/\epsilon)$ queries in time linear in the number of queries.*

Proof of Theorem 11: The algorithm we analyze is a generalization of the algorithms we presented in the previous sections: it works in k stages, where in each stage it considers a different partition $\mathcal{P}^i \in \mathcal{P}$. In the query version of the algorithm, in each stage it uniformly selects $m' = O(\log k/\delta^2)$ components of the partition considered, and in each component chosen it uniformly selects two points and queries the function f on these points. In the examples version it pairs the $m = O(\log k \cdot \sqrt{s'}/\delta^2)$ sample points according to the components they belong to (as done in the other algorithms). In both cases, if for some partition, the fraction of such *close* pairs that have the same label is at least $\frac{1}{2} + 2\alpha\beta^2 - \frac{\delta}{2}$ then the algorithm accepts, otherwise, it rejects.

The analysis of the algorithm uses similar arguments to those already presented. In particular, as shown in the proof of Theorem 2, based on our choice of the sample size m, and the number of components selected in each partition by the query version of the algorithm, m', it suffices to analyze the query version of the algorithm. First consider the case $f \in H$. By the theorem's premise concerning H, there exists a partition $\mathcal{P}^i$ such that on at least a fraction α of the component X^i_j of $\mathcal{P}^i$, for some $b_j \in \{0, 1\}$, $\Pr_{x \in X^i_j}[f(x) = b_j] \geq 1/2 + \beta$. Let χ_t be a 0/1 random variable that is 1 if an only if the t'th pair of points (uniformly selected in a uniformly chosen component of $\mathcal{P}^i$), have a different label. Then

$$\mathrm{E}[\chi_t] \;\leq\; \alpha \cdot \left(2\left(\frac{1}{2} + \beta\right)\left(\frac{1}{2} - \beta\right)\right) + (1 - \alpha) \cdot \frac{1}{2} \;=\; \frac{1}{2} - 2\alpha\beta^2 \tag{35}$$

The probability that the average of the χ_t's will deviate by more than $\delta/2$ from this expectation is $\exp(-2(\delta/2)^2 \cdot m') = O(1/k)$, and so the probability that the algorithm accepts when $\mathcal{P}^i$ is considered, is greater than $2/3$.

We now turn to show that in case f is far from H' then the algorithm rejects with probability at least 2/3. As in previous proofs we prove the contrapositive. Assume f is accepted with probability greater than $1/3$. Then there must be at least one partition $\mathcal{P}^i$ that causes the algorithm to accepts with probability greater than $1/(3k)$. Let f' be the function in $\mathrm{PAR}_{\mathcal{P}} \subseteq H'$ that labels the points in each component of $\mathcal{P}^i$ according to the majority label of f in that component. If we define χ_t as above,

then as shown before (see Equation (12) and preceding discussion), $\mathrm{E}[\chi_t] \geq \mathrm{dist}(f, f')$. Since with probability at least $1/(3k)$ the average of the χ_t's is at most $\frac{1}{2} - 2\alpha\beta^2 + \frac{\delta}{2}$ (which is the event in which that algorithm accepts when considering $\mathcal{P}^i$), given our choice of m', the expected value of χ_t, and hence $\mathrm{dist}(f, f')$, is at most $\frac{1}{2} - 2\alpha\beta^2 + \delta$. ■

Proof of Theorem 12: Here too the algorithm works in k stages, where in each stage it considers a different partition $\mathcal{P}^i$. In the query version of the algorithm, in each stage it randomly selects $m' = O(\log k/\epsilon)$ components, where each component is chosen according to its probability weight, and in each component it randomly selects two points. In the examples version of the algorithm it pairs the $m = O(\sqrt{s} \cdot \log k/\epsilon)$ sample points according to the components they belong to (as done in the other algorithms). If for some partition, *all* pairs belonging to common components have the same label, then the algorithm accepts. Otherwise it rejects.

Clearly, if $f \in \mathrm{PAR}_{\mathcal{P}}$ then the algorithm always accepts. Suppose $\mathrm{dist}(f, \mathrm{PAR}_{\mathcal{P}}) > \epsilon$, then we show that the algorithm rejects with probability at least $2/3$. Consider first the query version of the algorithm, and let us fix a partition $\mathcal{P}^i$. We next show that the probability that we obtain a pair of points in a common component having a different label is at least $1 - 1/(3k)$, and so the probability that we obtain such a pair for each partition, is at least $2/3$. For each component X^i_j in $\mathcal{P}^i$, let w_j be the probability weight (assigned by the underlying distribution) to X^i_j, and let γ_j be the relative weight of the points in X^i_j having the minority label according to f. Since for every $g \in \mathrm{PAR}_{\mathcal{P}}$, we have that $\mathrm{dist}(f, g) > \epsilon$, in particular this is true for the function g which labels the components of $\mathcal{P}^i$ according to the majority label of f. Since $\mathrm{dist}(f, g) = \sum_j w_j \cdot \gamma_j$, we have that $\sum_j w_j \cdot \gamma_j > \epsilon$. For $t = 1, \ldots, m'$, let χ_t be as defined in the previous proofs. Then,

$$\Pr[\chi_t = 1] \quad = \quad \sum_j w_j \cdot (2\gamma_j(1 - \gamma_j)) \quad \geq \quad \sum_j w_j \cdot \gamma_j \quad > \quad \epsilon \tag{36}$$

It follows that $\Pr[\sum_{t=1}^{m'} \chi_t = 0]$ (that is, the probability that all pairs have the same label), is bounded by $(1 - \epsilon)^{m'} < 1/(3k)$, as desired.

We now turn to the examples version of the algorithm. In case all components of each partition have exactly the same probability weight, then, as we have seen before, the analysis reduces to that of the query version, and we are done. Otherwise we appeal to the following technical lemma whose proof is a slight adaptation of a proof given in the paper [GGLR98], and is reproduced in the appendix.

Lemma 13 *Let $S_1, \ldots, S_s, T_1, \ldots, T_s$ be disjoint sets of elements over domain X. For each j, let the probability of selecting an element x in S_j (when x is chosen according to some fixed distribution on X), be p_j, and the probability of selecting an element in T_j, be q_j. Suppose that for all j, $q_j \geq p_j$, and that $\sum_j p_j \geq \epsilon$ for some $\epsilon > 0$. Then, for some constant c, if we uniformly select $c \cdot \sqrt{s}/\epsilon$ elements in X, then with probability at least $2/3$, for some j we shall obtain one element in S_j and one in T_j.*

In our application of the above claim, for each j, $S_j \cup T_j = X^i_j$, where S_j is the subcomponent of X^i_j having the minority label, and T_j is the one having the majority label. Thus, $p_j = w_j \cdot \gamma_j$, and $q_j = w_j \cdot (1 - \gamma_j)$, so that $q_j \geq p_j$, and $\sum p_j \geq \epsilon$ as required. Since the sample we use is $\log k$ times larger than that specified by the lemma, the probability that we obtain a pair with an opposite label is at least $1 - 1/(3k)$. The correctness of the examples version of the algorithm follows. ■

7 Testing and Weak Learning

The intuition that testing is easier (or at least not harder) than learning can be formalized as follows (generalizing a similar statement in Goldreich et al. [GGR96]).

Proposition 14 *Let F be a class of functions that is learnable by hypothesis class H, under distribution P, with confidence $5/6$ and accuracy $\epsilon \in (0, 1/2]$, in m random examples. Then for every $\epsilon' \in (0, 1/2]$, the class F is testable with rejection boundary $(H, \epsilon + \epsilon')$ with respect to P using $m + O(1/(\epsilon')^2)$ examples. If F is learnable with any accuracy ϵ in $m(\epsilon)$ examples, then F is testable with rejection boundary (H, ϵ) with respect to P using $m(\epsilon/2) + O(1/\epsilon)$ examples.*

Below we present a theorem concerning the reverse direction — namely, any class that is efficiently testable against a random function (see Definition 3) is efficiently weakly learnable. Recall that testing against a random function (with respect to a particular distribution P) is our least stringent definition whenever, with respect to P, a random function is far from any function in the class (with high probability). We expect this property to hold for any function class and distribution that emerge naturally in the context of learning.

Theorem 15 *Let H be a class of functions over domain X, and P a distribution over X. If H is testable against a random function in m examples with respect to P, then H is weakly learnable with respect to P with advantage $\Omega(1/m)$ and constant confidence in $\tilde{O}(m^2)$ examples.*

Proof: Let T be the testing algorithm that distinguishes between functions in H and a random function. We start by using a standard technique first applied in the cryptography literature [GM84]. Let us fix any function $f \in H$, and consider the behavior of the algorithm when it is given a random sample drawn according to P and labeled *partly by f and partly randomly*. More precisely, for $i = 0, \ldots, m$, let P_i be the probability, taken over a random sample $x_1, \ldots, x_m$ drawn according to P, and a vector $\vec{r}$ uniformly chosen vector in $\{0,1\}^{m-i}$, that the test T accepts when given as input $\langle x_1, f(x_1)\rangle, \ldots, \langle x_i, f(x_i)\rangle, \langle x_{i+1}, r_1\rangle\rangle, \ldots, \langle x_m, r_{m-i}\rangle\rangle$. Since $P_m \geq 2/3$, while $P_0 \leq 1/3$, there must exist an index $1 \leq i \leq m$ such that $P_i - P_{i-1} = \Omega(1/m)$. Thus, by observing $\tilde{O}(m^2)$ examples (and generating the appropriate number of random labels) we can find an index i such that T has significant sensitivity to whether the ith example is labeled by f or randomly. From this it can be shown [KLV95] that by taking another $\tilde{O}(m^2)$ examples, we can find a *fixed* sequence S_1 of i examples labeled according to f, and a fixed sequence S_2 of $m - i$ examples having an arbitrary (but fixed) $0/1$ labeling such that the difference between the probability that T accepts when given as input $S_1, \langle x, f(x)\rangle, S_2$ and the probability that it accepts when given as input $S_1, \langle x, \neg f(x)\rangle, S_2$, is $\Omega(1/m)$, where now the probability is taken only over the draw of x. Let $h(x)$ be the following probabilistic function. If $T(S_1, \langle x, 0\rangle, S_2) = T(S_1, \langle x, 1\rangle, S_2)$, then h outputs the flip of a fair coin. If for $b \in \{0,1\}$, $T(S_1, \langle x, b\rangle, S_2) = \text{ACCEPT}$ and $T(S_1, \langle x, \neg b\rangle, S_2) = \text{REJECT}$, then h outputs b. Then from the preceding arguments, h has an advantage of $\Omega(1/m)$ over a random coin in predicting f. ■

The following result translates testing a pair of function classes (see Definition 4) to weakly learning (almost all functions in) one of the classes.

Theorem 16 *Let H_1 and H_2 be finite classes of functions over domain X, and P a distribution over X. If (H_1, H_2) is testable in m examples with respect to P, then for any $\gamma > 0$, one of the following must hold:*

- *There exists an $i \in \{1,2\}$ and a subclass $H' \subseteq H_i$ such that $|H'| \geq (1-\gamma)|H_i|$, and H' is weakly learnable with advantage $\Omega(1/m)$ and constant confidence in $\tilde{O}(m^2)$ examples.*
- *There exists an $i \in \{1,2\}$ such that H_i is weakly learnable with advantage $\Omega(1/m)$ and constant confidence in $\tilde{O}(m^2/\gamma)$ examples.*

Proof: By a similar argument to the one given in the proof of Theorem 15, we can show that for any fixed $f_1 \in H_1$ and $f_2 \in H_2$ it is possible to construct (using $\tilde{O}(m^2)$ examples) a pair of (randomized) hypotheses h_1 and h_2, such that for either $i = 1$ or $i = 2$, h_i has an advantage of $\Omega(1/m)$ over random guessing in predicting f_i. When $i = 1$ we say that f_2 *loses* to f_1, and otherwise, f_1 loses to f_2. Fix $\gamma > 0$, and let us say that a function $f_1 \in H_1$ is *bad* if it loses to at least a fraction $1 - \gamma$ of the functions in H_2. Then if there exists a bad function in H_1, then by fixing f_1 to be this bad function in the above construction, we have an algorithm that can weakly learn the $1 - \gamma$ fraction of the functions in H_2 that f_1 loses to. On the other hand, if there is no bad function in H_1, then we can weakly learn any function in H_1: for any fixed $f_1 \in H_1$, if we randomly sample a function $f_2 \in H_2$ there is at least probability γ that we will draw a function that f_1 does not lose to. Thus, in $O(1/\gamma)$ tries, we will be able to weakly learn f_1. ■

8 Lower Bounds

For both function classes we consider, DT_s^d and AVN_s^d, we can show a lower bound of $\Omega(\sqrt{s})$ on the number of examples required for testing against a random function and with respect to the uniform distribution. Thus, in terms of the dependence on s, this lower bound almost matches our upper bounds, where note that in these case, testing against a random function is not harder than testing with the rejection boundaries we achieve. We prove the lower bound for the class of interval functions INT_s. Since $\mathrm{INT}_s = \mathrm{DT}_s^1$, and $\mathrm{INT}_s \subset \mathrm{AVN}_s^1$, this lower bound holds decision trees and aligned voting networks. As we shall see below, the proof is based on the lower bound associated with the Birthday Paradox.

Theorem 17 *The sample complexity for testing the class INT_s against a random function and with respect to the uniform distribution is $\Omega(\sqrt{s})$.*

Proof: We define the following distribution P_I over functions in INT_s. The distribution P_I is non-zero only on functions having switch point in the set $\{j/s\}_{j=1}^{s-1}$, and it assigns equal probability to each function that is constant over the equal-size subintervals, $(j/s, (j+1)/s]$. In other words, in order to draw a function in INT_s according to P_I, we randomly label the above subintervals (and then put switch points between any two subintervals that got opposite labels). Consider the following two distributions D_1 and D_2 over labeled samples S.

In both distributions, the examples are drawn uniformly. In D_1 they are labeled according to a function in INT_s chosen according to P_I, and in D_2 they are labeled randomly. Note that whenever the examples chosen do not include a pair of examples that belong to the same subinterval, then the distribution on the labels is the same in D_1 and D_2 (that is, it is uniform). It follows that the statistical difference between D_1 and D_2 is of the same order as the probability that the sample does include a pair of points that fall in the same subinterval. However, the probability that a sample of size m contains

such a pair of examples is bounded by $\binom{m}{2} \cdot (1/s)$, which for $m = \alpha\sqrt{s}$, is bounded by α^2. Thus, for any testing algorithm T, there exists at least one function $f \in \mathrm{INT}_s$, such that the probability that f is accepted (distinguished from random) when T is provided with $\alpha\sqrt{s}$ examples labeled by f, is $O(\alpha^2)$, which for an appropriate choice of α is less than $2/3$. ■

References

[BCH+95] M. Bellare, D. Coppersmith, J. Håstad, M. Kiwi, and M. Sudan. Linearity testing in characteristic two. In *Proceedings of the Thirty-Sixth Annual Symposium on Foundations of Computer Science*, pages 432–441, 1995.

[BLR93] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47:549–595, 1993.

[GGLR98] O. Goldreich, S. Goldwasser, E. Lehman, and D. Ron. Testing monotinicity. Long version of extended abstract that will appear in proceedings of FOCS98, available from `http://theory.lcs.mit.edu/~danar`, 1998.

[GGR96] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. In *Proceedings of the Thirty-Seventh Annual Symposium on Foundations of Computer Science*, pages 339–348, 1996.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[GR97] O. Goldreich and D. Ron. Property testing in bounded degree graphs. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, pages 406–415, 1997.

[GR98] O. Goldreich and D. Ron. A sublinear bipartite tester for bounded degree graphs. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing*, 1998.

[Kie87] Jack Carl Kiefer. *Introduction to Statistical Inference*. Springer Verlag, 1987.

[KLV95] M. Kearns, M. Li, and L. Valiant. Learning boolean formulae. *Journal of the Association for Computing Machinery*, 41(6):1298–1328, 1995.

[KZ93] S. R. Kulkarni and O. Zeitouni. On probably correct classification of concepts. In *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory*, pages 111–116, 1993.

[RS96] R. Rubinfeld and M. Sudan. Robust characterization of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.

[Rub94] R. Rubinfeld. Robust functional equations and their applications to program testing. In *Proceedings of the Thirty-Fifth Annual Symposium on Foundations of Computer Science*, 1994. To appear in SIAM Journal on Computing.

[Yam95] K. Yamanishi. Probably almost discriminative learning. *Machine Learning*, 18:23–50, 1995.

A Proof of Lemma 13

As a mental experiment, we partition the sample of elements into two parts of equal size, $c \cdot \sqrt{s}/(2\epsilon)$. Let J be a random variable denoting the (set of) indices of sets S_j hit by the first part of the sample. We show below that with probability at least $5/6$ over the choice of the first part of the sample,

$$\sum_{j \in J} p_j \quad \geq \quad \frac{\epsilon}{\sqrt{s}} \tag{37}$$

The claim then follows since conditioned on Equation (37) holding, and by the lemma's premise that that $q_j \geq p_j$ for all j, the probability that the second part of the sample does not include any elements from $\bigcup_{j \in J} T_j$, is at most

$$\left(1 - \sum_{j \in J} q_j\right)^{c \cdot \sqrt{s}/(2\epsilon)} \quad \leq \quad \left(1 - \frac{\epsilon}{\sqrt{s}}\right)^{c \cdot \sqrt{s}/(2\epsilon)} \quad < \quad \exp(-c/2)$$

which is less than $1/6$ for an appropriate choice of c.

To prove that Equation (37) holds with probability at least $5/6$, we assume without loss of generality that the sets S_j are ordered according to size. Let $S_1, \ldots, S_t$ be all sets with probability weight at least $\epsilon/(2s)$ each (i.e., $p_1 \geq \ldots \geq p_t \geq \epsilon/(2s)$). Then, the total probability weight of all other sets $S_{t+1}, \ldots, S_s$ is less than $\epsilon/2$, and $\sum_{j=1}^{t} p_j \geq \epsilon/2$ follows. We first observe that by a (multiplicative) Chernoff bound (for an appropriate choice of c), with probability at least $11/12$, the first part of the sample contains at least $4 \cdot \sqrt{s}$ elements in $\bar{S} \stackrel{\text{def}}{=} \bigcup_{i=1}^{t} S_j$.

Let $J' \stackrel{\text{def}}{=} J \cap \{1, \ldots, t\}$. That is, J' is a random variable denoting the indices of sets S_j, $j \in \{1, \ldots, t\}$ that are hit by the first part of the sample. Conditioned on there being at least $4 \cdot \sqrt{s}$ elements from $\bar{S}$ in the first part of the sample, we next show that with probability at least $11/12$, $\sum_{j \in J'} p_j \geq \frac{\epsilon}{\sqrt{s}}$ (from which Equation (37) follows). Since conditioned on an element belonging to $\bar{S}$ it is distributed according to the underlying distribution restricted to $\bar{S}$, we may bound the probability of the above event, when randomly selecting $4\sqrt{s}$ elements in $\bar{S}$ according to the underlying distribution. Consider the choice of the $\ell' th$ element from $\bar{S}$, and let $J'_{\ell-1}$ denote the indices of sets $S_j, j \in \{1, \ldots, t\}$, among the first $\ell - 1$ selected elements of $\bar{S}$. If

$$\sum_{j \in J'_{\ell-1}} p_j \quad \geq \quad \frac{2 \cdot \sum_{j=1}^{t} p_j}{\sqrt{s}}$$

then, since $\sum_{j=1}^{t} p_j \geq \frac{\epsilon}{2}$, we are done. Otherwise ($\sum_{j \in J'_{\ell-1}} p_j < (2 \sum_{j=1}^{t} p_j)/\sqrt{s}$), the probability that the $\ell' th$ element belongs to $J' \setminus J'_{\ell-1}$ (i.e., it hits a set in $\{S_1, \ldots, S_t\}$ that was not yet hit), is at least $1 - 2/\sqrt{s}$, which is at least $3/4$ for $s \geq 36$. Since we are assuming that the first part of the sample includes at least $4 \cdot \sqrt{s}$ elements from $\bar{S}$, with probability at least $11/12$, we succeed in obtaining a new element in at least $2 \cdot \sqrt{s}$ of these trials. Since the sets $S_1, \ldots, S_t$ all have probability weight at least $\epsilon/(2s)$, the claim follows. ■