Testing Problems with Sub-Learning Sample Complexity

Michael Kearns
AT&T Labs
mkearns@research.att.com

Dana Ron
M.I.T.
danar@theory.lcs.mit.edu

January 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 $\tilde{O}(\sqrt{s})$ 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.

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 can 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 also want to entertain 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 $\tilde{O}(\sqrt{s})$[1] random examples when the input dimension d is held constant and the underlying distribution is uniform. This is in stark 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 too 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 necessary to 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.

How are such tests possible? We begin by noting that they must look quite different from the standard learning algorithms. With only $\tilde{O}(\sqrt{s})$ 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 points in the domain 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 coincide. 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

[1] The $\tilde{O}(\cdot)$ notation is used to hide poly-logarithmic factors. For simplicity, in this introduction we use it for a slightly faster growing function (in the case of decision trees). For a precise statement see our theorems.

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. They have an unfortunate exponential dependence on the input dimension, which is why our tests are of interest only for fixed dimension, and will reject only functions that are rather far from the reference class H. Improvements or variations on these combinatorial constructions and the resulting tests are interesting open problems.

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. [1], whose framework we follow and generalize; their positive results are for graph-theoretic problems not typically examined in the learning literature, and their tests all require queries. Our work can be viewed as a study of the sample complexity of classical *hypothesis testing* [4] 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".

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 toy 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 prove an interesting connection between testing and the standard notion of weak learning from the computational learning theory literature. In Appendix A 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 $\mathrm{dist}_P(f, f') \stackrel{\mathrm{def}}{=} \Pr_{x \sim P}[f(x) \neq f'(x)]$, and the distance between f and H (with respect to P) is $\mathrm{dist}_P(f, H) \stackrel{\mathrm{def}}{=} \min_{f' \in H} \mathrm{dist}_P(f, f')$. 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 the resources required for testing for H will always be at most those required for learning H in the standard models. 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. [1]. 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 such that with high probability a random function is ϵ-far from H (for some $\epsilon < 1/2$), and H is testable with rejection boundary (H, ϵ) in m examples (queries), 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 $h_2 \in H_2$, $\mathrm{dist}_P(h_1, h_2) > \epsilon$. 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 from 0 to 1 (or 1 to 0) 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 $O(\sqrt{s}/\epsilon^{2.5})$ examples or $O(1/\epsilon^2)$ 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 $m = O(\sqrt{s'}/\epsilon^2)$, 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 according to their order in the sample, and let us pair the points in each such bin according to this order (i.e., 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 is at least $1 - 3\epsilon/4$; otherwise it rejects. When the algorithm is instead allowed queries, it uniformly selects $m' = O(1/\epsilon^2)$ 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 $m' = O(1/\epsilon^2)$. 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 cells be denoted C and recall that all cells have the same probability mass. Thus, the probability that S'_2 does *not* contain any example from a cell in C is $(1 - 1/|C|)^{|S'_2|} = (1 - (1/\sqrt{s'}))^{\sqrt{s'}} < e^{-1} < 1/2$, as claimed. Hence, with very high probability, at least a fourth of the subsamples (i.e., 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. Note that had we made the grid more refined (and so decreased the distance between each close pair) the number of examples required would grow respectively.

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 1 if tth close pair is pure, and 0 otherwise. Thus χ_t is determined by a two stage process: (1) The choice of the t'th grid cell c_i; (2) The selection

of the two points inside that cell. When c_t is a subinterval of some interval of f, then the points alway have the same label, and otherwise they have the same label with probability at least $1/2$. Since f has at most s intervals, the number of cells that intersect intervals of f (i.e., 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, $\mathrm{E}[\chi_t] \geq (1-\epsilon)\cdot 1 + \epsilon \cdot (1/2) = 1 - \epsilon/2$. By an additive 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 pure), is at least $1 - 3\epsilon/4$, 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 counterpositive: 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 $\mathrm{E}[\chi_t] = \mathrm{E}_j\left[(1-\epsilon_j)^2 + \epsilon_j^2\right] \leq 1 - \mathrm{E}_j[\epsilon_j]$ where the second inequality follows from $\epsilon \leq 1/2$. By our assumption on f, $\mathrm{E}[\chi_t] < 1 - \epsilon$, and by applying an additive Chernoff bound, with probability greater than $2/3$, the average over the χ_t's is less than $1 - 3\epsilon/4$. ■

4 Decision Trees

In this section we study the 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 it has, 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 $d > 1$ is superlinear in s, and 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, very few of the cells of G_i fall on 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 $\left(\mathrm{DT}_{s'}^d, \frac{1}{2} - \frac{1}{2^{d+5}(Cd)^d}\right)$ with respect to the uniform distribution in $\tilde{O}\left((2Cd)^{2.5d} \cdot s^{\frac{1}{2}(1+1/C)}\right)$ examples, or $O\left((2Cd)^{2d} \cdot \log(s)^{d+1}\right)$ queries. The time sufficient for testing is at most $(2\log(2s))^d$ larger than the number of examples (respectively, queries) used.*

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 about an order of $\sqrt{s}$ examples or about an 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. Here we simply note that our testing algorithm can be slightly modified to address this problem, and defer details until the full paper.

As mentioned above, when queries are allowed we can generalize Theorem 2 as follows.

Theorem 3 *For any s and for any ϵ, the class of decision trees DT_s^d is testable with rejection boundary $\left(\mathrm{DT}^d_{(sd/\epsilon)^d}, \epsilon\right)$ and with respect to the uniform distribution with $O(1/\epsilon^2)$ queries and in time $O(1/\epsilon^2)$.*

Because the proof of Theorem 3 is very similar to the proof of Theorem 1, it is omitted, and we direct our attention to proving Theorem 2.

The following combinatorial lemma is the main step in proving Theorem 2. We shall need the following notation: For a d-dimensional rectangle $R = (R_1, \ldots, R_d)$, where $R_j \in [0,1]$, we let $V(R)$ denote the *volume* of R, so $V(R) = \prod_{j=1}^d R_j$. If Q and R are d-dimensional rectangles, we say that Q *fits* in R if $Q_i \leq R_i$ for all i. Note that this notion is independent of the position of Q and R in space.

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. Assume 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 $V_{d-1}(R^i) \stackrel{\mathrm{def}}{=} \prod_{j=1}^{d-1} R^i_j$ 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 $b_0 \stackrel{\mathrm{def}}{=} \infty$). Since there are only k bins, there exists a bin, denoted B_g, that contains at least $k^{-(d-1)}$ 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.[2] 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 $R^i_d \geq v/b_{g-1}$. Combining this with the above application of the induction hypothesis, we know that there exists a d-dimensional rectangle Q such that $V(Q) \geq \frac{v}{b_{g-1}} \cdot b_g^{1+\frac{d-2}{k}}$ 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

[2] 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 $R = (R_1, \ldots, R_{d-1})$ in B'_g we have a rectangle $R' = (R_1, \ldots, R'_{d-1})$ 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.

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

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

Lemma 4 shows that some "large" rectangle Q must fit inside "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 very few cells will intersect a decision boundary of the decision tree — that is, almost all cells are purely labeled by the function.

Lemma 5 *Let f be a decision tree in DT^d_s, and let $R^1, \ldots, R^s$ be the d-dimensional rectangles defined by the leaves of f. 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 $R^1, \ldots, R^t$ 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 $2^{-(d+1)} \cdot V(R^i)$. Since the total volume of all good leaves is at least β, the lemma follows.

Consider any good rectangle $R^i = (R^i_1, \ldots, R^i_d)$ and the corresponding leaf L^i. For each dimension j, let $r^i_j \stackrel{\text{def}}{=} R^i_j/Q_j$. Hence, by definition, $V(R^i) = V(Q) \cdot \prod_j r^i_j$. Let $\bar{R}^i = (\bar{R}^i_1, \ldots, \bar{R}^i_d)$ 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 $\bar{R}^i_j \geq R^i_j - Q_j/2$, and so $V(\bar{R}^i) \geq V(Q) \cdot \prod_j (r^i_j - 1/2)$. Therefore $V(\bar{R}^i)/V(R^i) \geq \prod_j (r^i_j - 1/2)/r^i_j$. Since Q fits inside R^i, $r^i_j \geq 1$ for each j, and so $V(\bar{R}^i)/V(R^i) \geq 2^{-d}$, as claimed. ■

Proof of Theorem 2: We start by describing the testing algorithm. Let $m = O(2^{3d/2}(Cd)^{2d+1}(2s)^{\frac{1}{2}(1+1/C)} \cdot \log\log(s))$ be the size of the sample provided to the algorithm, and 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 $\sum i_j = n$, the algorithm considers the *grid* $G(i_1, \ldots, i_d)$ over $[0,1]^d$ whose grid-cells have length $2^{-(i_j+1)}$ in dimension j. For each such grid, the algorithms checks whether, among the set of (disjoint) pairs of points in the sample that fall in a common grid-cell, the fraction that have the same label is at least $\frac{1}{2} + \frac{1}{2^{d+4}(Cd)^d}$. (In the query version of the algorithm, the algorithm uniformly selects $m' = O((2Cd)^{2d+1} \cdot \log\log(s))$ grid cells in each grid, and for each cell chosen, it uniformly selects two points in the cell and queries f on these points.)

The proof that for any f such that $\mathrm{dist}(f, \mathrm{DT}^d_{s'}) > \frac{1}{2} - \frac{1}{2^{d+5}(Cd)^{d-1}}$, 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 $\mathrm{DT}^d_{s'}$), is analogous to the special case of the interval functions proved in Theorem 1, and is hence omitted.

In the remainder of the proof we analyze the case in which the function f is in fact a decision tree in DT^d_s. We show that for each $f \in \mathrm{DT}^d_s$, there exists a grid $G(i_1, \ldots, i_d)$, such that fraction of grid cells that fits inside leaves of f is at least $2^{-(d+2)} \cdot (Cd)^{-(d-1)}$. From that point on the proof proceed as the proof of Theorem 1 (details omitted).

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

$r^i \stackrel{\text{def}}{=} \lfloor V(R^i)/(1/(2s)) \rfloor$, 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 $\bar{R}^i_d \stackrel{\text{def}}{=} (1/(2s))/\prod_{j=1}^{d-1} R^i_j$. 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 $V(Q) \geq (1/2s)^{-(1+1/C)} \geq 2^{-n}$ (where n is as set in Step (1) of the algorithm), there must be at least one iteration of the algorithm in which the grid $G(i_1, \ldots, i_d)$ (defined in Step (2) of the algorithm) 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 $2^{-(d+2)} \cdot (Cd)^{-(d-1)}$. From this point on we proceed as in the proof of Theorem 1. ■

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 $f(\vec{x}) = sign\left(\sum_{i=j}^{s} h_j(\vec{x})\right)$, 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* $a^j_i \in [0,1]$ and *orientations* $u^j_i \in \{+1,-1\}$. The hyperplanes $x_i = a^j_i$ define the rectilinear partition, and f is constant over each cell c: for any $\vec{x}$, we define $\#(\vec{x}) = \sum_{i=1}^{d} \sum_{j=1}^{s_i} sign(x_i - u^j_i a^j_i)$ (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, exponential in 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 6 *Let AVN^d_s denote the class of aligned voting networks of size at most s over $[0,1]^d$. Then for any s and d, AVN^d_s 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 $O(6^{2^{d+6}}\sqrt{s})$ examples (and time), or $O(6^{2^{d+6}})$ 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 our comments following Theorem 2, here too we can slightly modify the algorithm so that it will not automatically accept biased random functions (details omitted).

Before giving the proof of Theorem 6, 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 7 *Let f be an aligned voting network over $[0,1]^d$ of size at most s. If $\Pr_{[0,1]^d}[f(\vec{x}) = +1] \geq \frac{1}{2} - \gamma_d$ then there exists a dimension i such that the total probability mass (with respect to the uniform distribution) of the $slice(i,j)$ of f for which $P_i^j = \Pr_{slice(i,j)}[f(\vec{x}) = +1] \geq \frac{1}{2} + \gamma_d$ is at least $2\gamma_d$, where $\gamma_d = 1/(6^{2^{d+1}})$. (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

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

From this we obtain $\alpha_H \geq \frac{1}{(1/2)-\gamma_d}\left((\gamma_d + \gamma_{d-1})\alpha_L - 2\gamma_d\right)$. If α_L satisfies

$$\alpha_L \geq \frac{((1/2) - \gamma_d)2\gamma_d + 2\gamma_d}{\gamma_d + \gamma_{d-1}} \tag{3}$$

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 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

$$\alpha_L \leq \frac{\gamma_{d-1}-\gamma_d}{(1/2)+\gamma_{d-1}} \tag{4}$$

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

Returning to the earlier constraint on α_L given by Equation (3), we find that Equations (3) and (4) can both be satisfied provided

$$\frac{((1/2)-\gamma_d)2\gamma_d+2\gamma_d}{\gamma_d+\gamma_{d-1}} \leq \frac{\gamma_{d-1}-\gamma_d}{(1/2)+\gamma_{d-1}} \tag{5}$$

First note that

$$\frac{((1/2)-\gamma_d)2\gamma_d+2\gamma_d}{\gamma_d+\gamma_{d-1}} \leq \frac{(1/2)2\gamma_d+2\gamma_d}{\gamma_{d-1}} = \frac{3\gamma_d}{\gamma_{d-1}} \tag{6}$$

and

$$\frac{\gamma_{d-1}-\gamma_d}{(1/2)+\gamma_{d-1}} \geq \gamma_{d-1}-\gamma_d \geq \gamma_{d-1}/2 \tag{7}$$

(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 7.

Corollary 8 *Let f be an aligned voting network over $[0,1]^d$ of size s. For $i \in \{1,\ldots,d\}$ and $j \in \{0,\ldots,2\cdot 6^{2^{d+1}}\cdot s-1\}$, let $b_i^j = j/(2\cdot 6^{2^{d+1}}\cdot s)$, 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, where $\gamma_d = 1/(6^{2^{d+1}})$.*

All that is missing for the proof of Theorem 6 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 9 *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.*

The proof of Lemma 9 is quite straightforward, and is hence ommited.

Proof of Theorem 6: Let γ_d be as defined in Lemma 7. The algorithm first approximates the bias of the function using $O(1/\gamma_d^2) = (6^{2^{d+2}})$ uniformly chosen examples. If the fraction of examples labeled 0 (similarly, 1) is at least $1/2+5\gamma_d/4$, then the algorithm accepts. Otherwise, for each dimension $i \in \{1,\ldots,d\}$ the algorithm considers the slices defined in Corollary 8. If for some dimension i, among all (disjoint) pairs of points that belong to a common slice, the fraction that have the same label is at least $\frac{1}{2}-\frac{1}{6^{2^{d+2}}}$, then the algorithm accepts. Otherwise it rejects. (When queries are allowed the algorithm uniformly selects $O(1/\gamma_d^2)$ slices and queries f on a uniformly selected pair in the slice). By appealing to this algorithm, the proof of the theorem follows from Corollary 8 and Lemma 9 using the same arguments applied in the proof of Theorem 1. ■

6 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 [1]).

Proposition 10 *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. In case 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, that any class which is efficiently testable against a random function (see Definition 3), is efficiently weakly learnable.

Theorem 11 *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.*

The proof of Theorem 11, which is given in Appendix B, applies a technique (known as the *hybrid* or *probability walk* technique), which was first used in the context of Cryptography [2]. By using the same technique we get a more general result that translates testing a pair of function classes (see Definition 4) to weakly learning (almost all functions in) one of the classes. The Proof of Theorem 12 is also given in Appendix B.

Theorem 12 *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*

References

[1] 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.

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

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

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

A 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. To illustrate the common idea for the lower bound (which is based on the lower bound associated with the Birthday Paradox), we prove the claim below for interval functions.

Theorem 13 *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 (i.e., 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 $2/3$. ■

B Proofs for Section 6

Proof of Theorem 11: 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 [2]. 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

$$P_i \stackrel{\mathrm{def}}{=} \Pr_{P,\vec{r}}[T(\langle x_1, f(x_1)\rangle, \ldots, \langle x_i, f(x_i)\rangle, \langle x_{i+1}, r_1\rangle), \ldots \langle x_m, r_{m-i}\rangle\rangle) = \textsc{accept}] \tag{8}$$

where $\vec{r}$ is a uniformly chosen vector in $\{0,1\}^{m-i}$. 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 (see analysis in [3]) 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

$$\Pr_P[T(S_1, \langle x, f(x)\rangle, S_2) = \textsc{accept}] - \Pr_P[T(S_1, \langle x, \neg f(x)\rangle, S_2) = \textsc{accept}] = \Omega(1/m) \tag{9}$$

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. ■

Proof of Theorem 12: By a similar argument to the one given in the proof of Theorem 11, 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. ■