Reconstruction and testing via decision trees

Guy Blanc *Stanford*

Jane Lange *MIT*

Li-Yang Tan *Stanford*

May 8, 2021

Abstract

We study sublinear and local algorithms for decision trees. Our main result gives the first *reconstruction algorithm* for decision trees: given query access to a function f that is opt_s-close to a size-s decision tree, this algorithm provides query access to a fixed decision tree T where:

- T has size $S \coloneqq s^{O((\log s)^2/\varepsilon^3)}$;
- $\mathrm{dist}(f, T) \leq O(\mathrm{opt}_s) + \varepsilon$;
- Every query to T is answered with $\mathrm{poly}((\log s)/\varepsilon) \cdot \log n$ queries to f and in $\mathrm{poly}((\log s)/\varepsilon) \cdot n \log n$ time.

This yields a *tolerant tester* that distinguishes functions that are ε-close to size-s decision trees from functions that are $\Omega(\varepsilon)$-far from size-S decision trees with $\mathrm{poly}((\log s)/\varepsilon) \cdot \log n$ queries and in $\mathrm{poly}((\log s)/\varepsilon) \cdot n \log n$ time. Existing testers distinguish functions that are exactly size-s decision trees from those that are ε-far from from size-s decision trees with $\tilde{O}(s/\varepsilon)$ queries and in $\mathrm{poly}(s^s, 1/\varepsilon) \cdot n$ time.

We complement these algorithms with a hardness result for distinguishing whether an unknown function is ε-close-to or $\Omega(\varepsilon)$-far-from size-s decision trees: we show that an efficient algorithm for this task would yield an efficient algorithm for properly learning decision trees, a central open problem of learning theory.

Since decision tree complexity is well known to be related to numerous other boolean function properties—Fourier degree, randomized and quantum query complexities, certificate complexity, sensitivity, etc.—our results provide a new approach to reconstructing and testing these properties.

1 Introduction

We study sublinear and local algorithms for decision trees, focusing on the related problems of *testing* [RS96, GGR98] and *reconstruction* [ACCL08, SS10, AT10]. In testing, we would like to determine if an unknown function f is close to a size-s decision tree. In reconstruction, we are given query access to a function f that is promised to be close to a size-s decision tree, and we would like to provide fast query access to a decision tree, ideally of size not much larger than s, that is close to f.

Both testing and reconstruction have strong connections to learning. Testers can be used as an exploratory precursor to learning, and reconstruction algorithms serve as "on-the-fly repair procedures" that restore properties of a dataset that have been lost due to noise. There is a formal sense in which testing and reconstruction are no harder than learning: it is easy to see that *proper* learning algorithms—in the case of decision trees, algorithms that return a decision tree hypothesis—yield testers and reconstruction algorithms of comparable efficiency. However, this way of designing testers and reconstruction algorithms runs counter to their very purpose: such a tester makes the exploratory testing phase no more efficient than the actual learning phase, and such a reconstruction algorithm does not operate "on the fly". Relatedly, standard information-theoretic arguments show that any algorithm for learning size-s decision trees has to make $\Omega(s)$ queries, and hence take $\Omega(s)$ time. All of this leads us to a motivating question in the study of sublinear and local computation algorithms:

Can we test and reconstruct more efficiently than we can learn?

In this work, we study this question for decision trees and provide both positive and negative answers. Since decision tree complexity is well known to be related to numerous other boolean function properties—Fourier degree, randomized and quantum query complexities, certificate complexity, sensitivity, etc.—our results also have implications for these properties.

1.1 Our results

Our main result gives the first reconstruction algorithm for decision trees. Our algorithm has a polylogarithmic dependence on s in its query and time complexities, exponentially smaller than the information-theoretic minimum required to learn.

Theorem 1 (Reconstruction of decision trees)**.** *There is a randomized algorithm which, given query access to $f : \{\pm 1\}^n \to \{\pm 1\}$ and parameters $s \in \mathbb{N}$ and $\varepsilon \in (0, 1)$, provides query access to a decision tree T where*

- *T has size $s^{O((\log s)^2/\varepsilon^3)}$;*
- *$\mathrm{dist}(T, f) \le O(\mathrm{opt}_s) + \varepsilon$ w.h.p., where opt_s denotes the distance of f to the closest size-s decision tree;*
- *Every query to T is answered with $\mathrm{poly}((\log s)/\varepsilon) \cdot \log n$ queries to f and in $\mathrm{poly}((\log s)/\varepsilon) \cdot n \log n$ time.*

Our reconstruction algorithm is furthermore *local* in the sense of [SS10], allowing queries to be answered in parallel assuming there is a shared random string; see Remark 3. In particular, once

f, s, ε and the random string are fixed, all queries are answered consistent with a single decision tree.

As a consequence of Theorem 1, we obtain a tolerant tester for decision trees:

Corollary 1 (Testing decision trees)**.** *There is a randomized algorithm which, given query access to $f : \{\pm 1\}^n \to \{\pm 1\}$ and parameters $s \in \mathbb{N}$ and $\varepsilon \in (0, 1)$,*

- *Makes $\mathrm{poly}((\log s)/\varepsilon) \cdot \log n$ queries to f, runs in $\mathrm{poly}((\log s)/\varepsilon) \cdot n \log n$ time, and*
- *Accepts w.h.p. if f is ε-close to a size-s decision tree;*
- *Rejects w.h.p. if f is $\Omega(\varepsilon)$-far from size-$s^{O((\log s)^2/\varepsilon^3)}$ decision trees.*

Decision trees have received significant attention in the testing literature [KR00, DLM+07, CGSM11a, BBM12, Bsh20]. We give a detailed overview of prior work in Section 1.2, mentioning for now that existing testers are non-tolerant and have an $\omega(s)$ and s^s dependence in their query and time complexities respectively.

The possibility of improved parameters. It is natural to ask if Corollary 1 can be strengthened so that the tester rejects all f's that are $\Omega(\varepsilon)$-far from size-s decision trees—or more strongly, whether Theorem 1 can be improved to provide query access to a size-s decision tree. We show that such a tester would yield an efficient algorithm for properly learning decision trees:

Theorem 2 (Testing $\Rightarrow$ Proper learning)**.** *Suppose there is an algorithm which, given query access to $f : \{\pm 1\}^n \to \{\pm 1\}$ and parameters $s \in \mathbb{N}$ and $\varepsilon \in (0, 1)$,*

- *Makes $\mathrm{poly}(s, n, 1/\varepsilon)$ queries to f, runs in $\mathrm{poly}(s, n, 1/\varepsilon)$ time, and*
- *Accepts w.h.p. if f is ε-close to a size-s decision tree;*
- *Rejects w.h.p. if f is $\Omega(\varepsilon)$-far from size-s decision trees.*

Then there is a $\mathrm{poly}(s, n, 1/\varepsilon)$-time membership query algorithm for properly learning size-s decision trees with respect to the uniform distribution.

We note that the assumption of Theorem 2 allows for testers with time and query complexities that are considerably higher than those of our algorithm in Corollary 1. The classic work of Ehrenfeucht and Haussler [EH89] gave a $\mathrm{poly}(n^{\log s}, 1/\varepsilon)$-time algorithm for properly learning decision trees. Three decades later this remains the state of the art. A $\mathrm{poly}(s, n, 1/\varepsilon)$-time algorithm would be a significant breakthrough, and indeed, it is plausible that no such algorithm exists; see Section 1.2.

As alluded to above, it has long been known [GGR98] that proper learning algorithms for any class $\mathcal{H}$ yield comparably efficient testers for $\mathcal{H}$. Theorem 2 provides an example of a converse.

Reconstruction algorithms and testers for other properties. Decision tree complexity is well known to be related to numerous other basic complexity measures of boolean functions: Fourier degree, approximate degree, randomized and quantum query complexities, certificate complexity, block sensitivity, sensitivity, etc. Our results provide a new approach to reconstructing and testing these properties. For example, we have the following reconstruction algorithm for functions with low *Fourier degree*:

Corollary 2 (Reconstruction of low Fourier degree functions)**.** *There is a randomized algorithm which, given query access to $f : \{\pm 1\}^n \to \{\pm 1\}$ and parameters $d \in \mathbb{N}$ and $\varepsilon \in (0,1)$, provides query access to a fixed function $g : \{\pm 1\}^n \to \{\pm 1\}$ where*

- *g has Fourier degree $O(d^7/\varepsilon^2)$,*
- *$\mathrm{dist}(f,g) \le O(\mathrm{opt}_d) + \varepsilon$ w.h.p., where opt_d denotes the distance of f to closest $h : \{\pm 1\}^n \to \{\pm 1\}$ of Fourier degree d.*

Every query to g is answered in $\mathrm{poly}(d, 1/\varepsilon) \cdot n \log n$ time and with $\mathrm{poly}(d, 1/\varepsilon) \cdot \log n$ queries to f.

This reconstruction algorithm in turn yields a tester that runs in $\mathrm{poly}(d, 1/\varepsilon) \cdot n \log n$ time, makes $\mathrm{poly}(d, 1/\varepsilon) \cdot \log n$ queries to a function f, accepts w.h.p. if f is ε-close to having degree d, and rejects w.h.p. if f is $\Omega(\varepsilon)$-far from having degree $O(d^7/\varepsilon^2)$. Corollary 2 gives the first reconstruction algorithm for low Fourier degree functions, though there has been extensive work on testing for low Fourier degree [DLM+07, CGSM11a, CGSM11b, BBM12, BH13, Bsh20].

Table 1 lists the measures for which we obtain new reconstruction algorithms, each of which in turn gives a new tester.

Complexity measure	*Assumption* Query access to f that is opt_d-close to h where:	*Guarantee* Query access to g that is $O(\mathrm{opt}_d + \varepsilon)$-close to f where:
Fourier degree	$\deg(h) \le d$	$\deg(g) \le O(d^7/\varepsilon^2)$
Approximate degree	$\widetilde{\deg}(h) \le d$	$\widetilde{\deg}(g) \le O(d^9/\varepsilon^2)$
Randomized query complexity	$\mathrm{R}(h) \le d$	$\mathrm{R}(g) \le O(d^7/\varepsilon^2)$
Quantum query complexity	$\mathrm{Q}(h) \le d$	$\mathrm{Q}(g) \le O(d^{10}/\varepsilon^2)$
Certificate complexity	$\mathrm{C}(h) \le d$	$\mathrm{C}(g) \le O(d^5/\varepsilon^2)$
Block sensitivity	$\mathrm{bs}(h) \le d$	$\mathrm{bs}(g) \le O(d^8/\varepsilon^2)$
Sensitivity	$\mathrm{s}(h) \le d$	$\mathrm{s}(g) \le O(d^{13}/\varepsilon^2)$

Table 1: Performance guarantees of our reconstruction algorithms for various complexity measures. In all cases, every query to g is answered in $\mathrm{poly}(d, 1/\varepsilon) \cdot n \log n$ time with $\mathrm{poly}(d, 1/\varepsilon) \cdot \log n$ queries to f.

1.2 Background and comparison with prior work

In the language of property testing, Corollary 1 gives a *tolerant* tester for decision trees in the *parameterized* setting. A tolerant tester [PRR06] distinguishes functions that are ε_1-close to having

a certain property from those that are ε_2-far from having it, a more challenging task than non-tolerant testing where $\varepsilon_1 = 0$. In the *parameterized* model of property testing [KR00], one is interested in distinguishing functions that are in, or close to, a class $\mathcal{H}$ from functions that are far from a larger class $\mathcal{H}' \supseteq \mathcal{H}$, a relaxation of the standard non-parameterized setting where $\mathcal{H}' = \mathcal{H}$.

Corollary 1 gives the first tolerant tester for decision trees: previously, there were no non-trivial algorithms that distinguished functions that are ε-close to size-s decision trees from those that are $\Omega(\varepsilon)$-far from size-S decision trees for any choice of S. Tolerant testing, like agnostic learning, is motivated by the fact in many settings, the relevant function f is merely well-approximated by a small decision tree rather than fit perfectly by a small decision tree.

On the other hand, *non-tolerant* testing of decision trees has been intensively studied in both the non-parameterized and parameterized settings:

Non-parameterized testing of decision trees. We begin by contrasting Corollary 1 with the state of the art for non-tolerant testing in the non-parameterized setting: the task of distinguishing size-s decision trees from functions that are ε-far from size-s decision trees. Recent work of Bshouty [Bsh20] gives the current best algorithm for this task, running in $\mathrm{poly}(s^s, 1/\varepsilon) \cdot n$ time and using $O((s \log s)/\varepsilon)$ queries. Prior to [Bsh20], Chakraborty, García-Soriano, and Matsliah [CGSM11a] gave an $O((s \log s)/\varepsilon^2)$-query algorithm, and before that Diakonikolas, Lee, Matulef, Onak, Rubinfeld, Servedio, and Wan [DLM+07] gave an $\tilde{O}(s^4/\varepsilon^2)$-query algorithm. Like [Bsh20]'s algorithm, the algorithms of [CGSM11a, DLM+07] also run in $\mathrm{poly}(s^s, 1/\varepsilon) \cdot n$ time. This exponential dependence on s in the runtimes is due to the common framework that all three works employ ("testing by implicit learning", introduced in [DLM+07]), which involves a brute-force search over all $s^{O(s)}$ many size-s decision trees over s variables. This drawback of the otherwise versatile and powerful framework was noted in [DLM+07, DLM+08, Ser10], and these papers raised the natural open problem of obtaining more efficient testers.

Compared to these algorithms, our algorithm in Corollary 1 solves an incomparable problem with efficiency parameters that compare rather favorably with theirs. Our time and query complexities both depend polylogarithmically on s instead of exponentially and super-linearly respectively, albeit with an additional $\log n$ dependence in both cases. (We discuss this additional $\log n$ dependence in Section 1.3.) For example, for the natural setting of $s = \mathrm{poly}(n)$, our algorithm runs in $\tilde{O}(n) \cdot \mathrm{poly}(1/\varepsilon)$ time and makes $\mathrm{polylog}(n) \cdot \mathrm{poly}(1/\varepsilon)$ queries, whereas these algorithms run in $\exp(\mathrm{poly}(n)) \cdot \mathrm{poly}(1/\varepsilon)$ time and make $\mathrm{poly}(n) \cdot \mathrm{poly}(1/\varepsilon)$ queries.

Parameterized testing of decision trees. Kearns and Ron [KR00] were the first to propose the parameterized model of property testing, and decision trees were a motivating function class that they studied in their paper: they gave a tester with time and query complexities $\mathrm{poly}(n^n, (\log s)^n)$ that distinguishes size-s decision trees over $[0,1]^n$ from functions that are $(\frac{1}{2} - n^{-\Theta(n)})$-far from size-$\mathrm{poly}(2^n, s)$ decision trees. Compared to our algorithm in Corollary 1, Kearns and Ron handle decision trees over a more expressive domain ($[0,1]^n$ rather than $\{0,1\}^n$), but the parameters of their result are such that one should think of the dimension 'n' as being a constant rather than an asymptotic parameter.

A number of properties have since been studied within [KR00]'s parameterized model of property testing, including graphs of small diameter [PR02, CGR13], sets in $\mathbb{R}^n$ with small surface area [KNOW14, Nee14], and juntas [FKR+04, RT11, BBM12, BCE+18].

Property reconstruction. Property reconstruction was introduced by Ailon, Chazelle, Comandur, and Liu [ACCL08] and Austin and Tao [AT10]. ([AT10] termed such algorithms "repair algorithms".) Reconstruction has since been studied for a number of properties, including monotone functions [ACCL08, SS10, BGJ+12], hypergraph properties [AT10], convexity [CS06], expanders [KPS13], Lipschitz functions [JR13], graph connectivity and diameter [CGR13], and error correcting codes [CFM14]. Property reconstruction falls within the *local computation algorithms* framework of Rubinfeld, Tamir, Vardi, and Xie [RTVX11].

The recent paper [BGLT20a] designs a new decision tree learning algorithm that is amenable to *learnability estimation*, a notion introduced by Kong and Valiant [KV18] and Blum and Hu [BH18]: given a training set S of *unlabeled* examples, the performance of this algorithm $\mathcal{A}$ trained on S—that is, the generalization error of the hypothesis that $\mathcal{A}$ would construct if we were to label all of S and train $\mathcal{A}$ on it—can be accurately estimated by labeling only a small number of the examples in S. From the results in [BGLT20a] one can derive a reconstruction algorithm that achieves guarantees similar to those in Theorem 1, but only for *monotone* functions f. This limitation is inherent: as noted in [BGLT20a], their algorithm is known to fail for non-monotone functions. Our work draws on and extends the techniques in [BGLT20a].

Testers for Fourier degree and other properties in Table 1. There has been extensive work on testing functions of low Fourier degree [DLM+07, CGSM11a, CGSM11b, BBM12, BH13, Bsh20]. Existing testers are non-tolerant: the current best tester distinguishes functions of degree d from those that are ε-far from having degree d in $\mathrm{poly}(2^d) \cdot n$ time and with $\mathrm{poly}(2^d, 1/\varepsilon)$ queries [Bsh20]; this should be contrasted with the guarantees of the tester that follows from Corollary 2. The comparison with the prior state of the art for all other properties listed in Table 1 is qualitatively the same, and in particular, our work gives the first tolerant testers for these properties.

Proper learning of decision trees. Turning to our hardness result (Theorem 2), the literature on learning decision trees is vast. While numerous algorithms have been developed, most of them are *improper*, returning a hypothesis that is not itself a decision tree. Notably, Kusilevitz and Mansour [KM93] gave a $\mathrm{poly}(s, n, 1/\varepsilon)$-time algorithm for learning size-s decision trees over $\{0,1\}^n$, but their algorithm returns a hypothesis that is the sign of a low-degree polynomial.

As for proper learning algorithms, the classic work of Ehrenfeucht and Haussler [EH89] gave an algorithm that runs in $\mathrm{poly}(n^{\log s}, 1/\varepsilon)$ time. This remains the state of the art, and a $\mathrm{poly}(s, n, 1/\varepsilon)$-time proper learning algorithm would be a significant breakthrough. In fact, for the more challenging setting of learning with respect to an *arbitrary* distribution (as opposed to the uniform-distribution setting of Theorem 2), Alekhnovich, Braverman, Feldman, Klivans, and Pitassi [ABF+09] have shown that there is no such algorithm unless $\mathsf{NP} \subseteq \mathsf{DTIME}(2^{n^{o(1)}})$. It remains open whether a similar impossibility result also holds for the uniform-distribution setting [Fel16].

The work of [BGLT20b]. Our reconstruction algorithm is built on a key new structural lemma about decision trees that was recently established in the context of proper learning [BGLT20b]. The proof of this lemma relies on the *OSSS inequality* from the analysis of boolean functions [OSSS05]. [BGLT20b] in fact uses a sophisticated generalization of the OSSS inequality, the "two-function version for semi-metrics", established by [OSSS05] themselves and appearing as Theorem 3.3 of their paper. Unfortunately, Theorem 3.3 of [OSSS05] has recently been shown to be false.[1]

[1] This was shown by Mingda Qiao. We thank him and Ryan O'Donnell for discussions about this.

In this work we recover [BGLT20b]'s structural lemma (modulo minor changes in parameters) by sidestepping their use of Theorem 3.3 of [OSSS05]. Our overall proof strategy remains very much in the spirit of [BGLT20b]'s.

1.3 Future directions

Compared to the problem of learning decision trees, for which three decades of research has yielded strong algorithms and lower bounds in a variety of models and settings, there are surprisingly large gaps in our understanding of the problems of testing and reconstruction. Our results fill in some of these gaps and map out new connections to learning, but there remains much more to be done. We list a few concrete avenues for future work suggested by our results:

- *Testing via reconstruction meets testing by implicit learning?* Can our reconstruction-based approach to testing can be fruitfully combined with the testing by implicit learning framework employed by previous works [DLM+07, CGSM11a, Bsh20]? A motivating question here is whether the $\mathrm{poly}((\log s)/\varepsilon) \cdot \log n$ query complexity of our tester can be made independent of n. We believe that this may be possible via a combination of the two approaches, though potentially at the price of a linear dependence on s in both time and query complexities, as opposed to the $\mathrm{polylog}(s)$ dependence that we achieve. Relatedly, it would be interesting to prove a lower bound showing that any tester achieving a $\mathrm{polylog}(s)$ dependence necessarily has to incur a dependence on n.

- *Tighter connections between testing and learning:* Our tester rejects functions that are $\Omega(\varepsilon)$-far from $\mathrm{quasipoly}(s)$ decision trees, and Theorem 2 shows that a tester that rejects functions that are $\Omega(\varepsilon)$-far from size-s decision trees would yield a comparably efficient algorithm for properly learning decision trees. A concrete avenue for future work is to narrow this gap between $\mathrm{quasipoly}(s)$ and s, with the ultimate goal of getting them to match.

  There are also other ways in which Theorem 2 could be strengthened: Do *non-tolerant* testers for decision trees yield proper learning algorithms? Do tolerant testers yield proper learning algorithms with *agnostic* guarantees?

  More broadly, it would be interesting to exhibit other function classes for which property testers yield learning algorithms, and to explore the extent to which techniques from property testing can be leveraged to make progress in learning theory.

- *Improved reconstruction algorithms and testers for other properties:* The reconstruction algorithms that we obtain for the properties listed in Table 1 follow by combining Theorem 1 with known relationships between these measures and decision tree complexity. It would be interesting to obtain improved parameters by designing reconstruction algorithms that are tailored to each of these properties, without going through decision trees.

  The same questions can be asked of property testers, and about properties that are not known to be quantitatively related to decision tree size. Can we achieve similar exponential improvements in the time and query complexities of non-parameterized testers by relaxing to the parameterized setting? Theorem 2 suggests that for certain properties, efficient algorithms may only be possible in the parameterized setting.

Finally, we mention that there remains a large gap in the known bounds on the query complexity of non-tolerant testing of decision trees in the non-parameterized setting: the current best upper

bound is $\tilde{O}(s)$ [Bsh20, CGSM11a] whereas the current best lower bound is $\Omega(\log s)$ [DLM+07, BBM12].

Notation. All probabilities and expectations are with respect to the uniform distribution unless otherwise stated; we use boldface (e.g. $\boldsymbol{x}$) to denote random variables. For two functions $f, g : \{\pm 1\}^n \to \{\pm 1\}$, we write $\mathrm{dist}(f, g)$ to denote the quantity $\Pr[f(\boldsymbol{x}) \neq g(\boldsymbol{x})]$. We say that f and g are *ε-close* if $\Pr[f(\boldsymbol{x}) \neq g(\boldsymbol{x})] \leq \varepsilon$, and *$\varepsilon$-far* otherwise.

For a function $f : \{\pm 1\}^n \to \{\pm 1\}$, a decision tree T over the same variables as f, and a node v in T, we write f_v to denote the subfunction of f obtained by restricting f according to the root-to-v path in T. We write $|v|$ to denote the depth of v within T, and so the probability that a uniform random $\boldsymbol{x} \sim \{\pm 1\}^n$ reaches v is $2^{-|v|}$.

2 Proofs of Theorem 1 and Corollary 1

Our proof of Theorem 1 has two main components:

- A structural lemma about functions f that are opt_s-close to a size-s decision tree $T^\star$. While we have no information about the structure of this tree $T^\star$ that f is opt_s-close to, we will show that f is $O(\mathrm{opt}_s + \varepsilon)$-close to a tree $T^\diamond$ of size $S = S(s, \varepsilon)$ with a very specific structure.
- An algorithmic component that leverages this specific structure of $T^\diamond$ to show that for any input $x \in \{\pm 1\}^n$, the value of $T^\diamond(x)$ can be computed with only $\log S \cdot \log n$ queries to f.

As discussed in Section 1.2, the structural lemma is essentially due to [BGLT20b]. Their proof, however, relies on a result—Theorem 3.3 of [OSSS05], the "two-function OSSS inequality for semi-metrics"—that has recently been shown to be false by Mingda Qiao. We recover [BGLT20b]'s structural lemma (modulo minor changes in parameters) by sidestepping their use of this result; our overall proof strategy remains very much in the spirit of [BGLT20b]'s. Aside from this issue, there are also various technical differences between our setting and that of [BGLT20b]'s that necessitate minor adjustments to the proof; see Remark 1.

Section 2.1 will be devoted to the structural lemma and Section 2.2 to the algorithmic component. We prove Theorem 1 in Section 2.2.2, and we derive Corollary 1 as a simple consequence of Theorem 1 in Section 2.3.

2.1 Structural component of Theorem 1

Definition 1 (Noise sensitivity)**.** *The noise sensitivity of $f : \{\pm 1\}^n \to \{\pm 1\}$ at noise rate p is the quantity*

$$\mathrm{NS}_p(f) := \Pr[f(\boldsymbol{x}) \neq f(\boldsymbol{y})],$$

where $\boldsymbol{x} \sim \{\pm 1\}^n$ is uniform random and $\boldsymbol{y} \sim_p \boldsymbol{x}$ is a p-noisy copy of $\boldsymbol{x}$, obtained from $\boldsymbol{x}$ by independently rerandomizing each coordinate with probability p.

We assign each coordinate $i \in [n]$ of a function f a *score*, which measures the expected decrease in the noise sensitivity of f if x_i is queried:

Definition 2 (Score of a variable)**.** *Given a function $f : \{\pm 1\}^n \to \{\pm 1\}$, noise rate $p \in (0,1)$, and coordinate $i \in [n]$, the score of x_i is defined as*

$$\mathrm{Score}_i(f,p) = \mathrm{NS}_p(f) - \mathop{\mathbb{E}}_{\boldsymbol{b}\in\{\pm 1\}}\left[\mathrm{NS}_p(f_{x_i=\boldsymbol{b}})\right].$$

(Our notion of score is equivalent, up to scaling factors depending on p, to the notion of "noisy influence" as in [O'D14, BGLT20b]. We use our definition of score as it simplifies our presentation.) We are now ready to define the tree $T^\diamond$ described at the beginning of this section and state our structural lemma.

Definition 3. *For a function $f : \{\pm 1\}^n \to \{\pm 1\}$, parameters $d \in \mathbb{N}$ and $p \in (0,1)$, we write $T_f^{d,p}$ to denote the complete decision tree of depth d defined as follows:*

- *At every internal node v, query x_i where $i \in [n]$ maximizes $\mathrm{Score}_i(f_v, p)$.* [2]
- *Label every leaf ℓ with $\mathrm{sign}(\mathbb{E}[f_\ell])$.*

Lemma 1 (Structural lemma)**.** *Let $f : \{\pm 1\}^n \to \{\pm 1\}$ be opt_s-close to a size-s decision tree. Then for $d = O((\log s)^3/\varepsilon^3)$ and $p = \varepsilon/(\log s)$, we have $\mathrm{dist}(f, T_f^{d,p}) \le O(\mathrm{opt}_s) + \varepsilon$.*

Remark 1 (Technical differences between our setting and [BGLT20b]'s)**.** [BGLT20b] analyzes a tree, call it Υ, which is similar to $T_f^{d,p}$ but differs in a couple of ways. Unlike $T_f^{d,p}$, their tree Υ is not necessarily complete: it is iteratively constructed in a top-down manner, where in each iteration the size of the tree grows by one. In each iteration, the leaf ℓ in the current tree with the highest "value" is replaced with a query the variable of f_ℓ with the highest score, where the "value" of a leaf is defined to be the score of the highest-scoring variable of f_ℓ normalized by ℓ's depth in the current tree. [BGLT20b]'s definition of score differs from ours: for their intended application, it was important that the score of a variable can be efficiently estimated to high accuracy from random labeled examples $(\boldsymbol{x}, f(\boldsymbol{x}))$ where $\boldsymbol{x} \sim \{\pm 1\}^n$ is uniform random; Definition 2 does not lend itself to such an estimation procedure.

Due to these differences, Lemma 1 does not appear explicitly in [BGLT20b], but the essential ideas underlying its proof are from [BGLT20b].

Noise-sensitivity-based potential function. First, we introduce the potential function that will facilitate our proof of Lemma 1. Every decision tree T naturally induces a distribution over its leaves where each leaf ℓ receives weight $2^{-|\ell|}$. We write $\boldsymbol{\ell} \sim T$ to denote a draw of a leaf of T according to this distribution.

Definition 4 (Noise sensitivity of f with respect to a tree T)**.** *Let $f : \{\pm 1\}^n \to \{\pm 1\}$ be a function, $p \in (0,1)$, and T be a decision tree. The noise sensitivity of f at noise rate p with respect to T is the quantity*

$$\mathrm{NS}_p(f,T) \coloneqq \mathop{\mathbb{E}}_{\boldsymbol{\ell}\sim T}\left[\mathrm{NS}_p(f_{\boldsymbol{\ell}})\right].$$

Note that if T is the empty tree, then $\mathrm{NS}_p(f,T)$ is simply $\mathrm{NS}_p(f)$, the noise sensitivity of f at noise rate p. The following proposition is a bound on $\mathrm{NS}_p(f)$ that takes into account its distance from a small decision tree:

[2] Ties are arbitrarily broken; our results hold regardless of how ties are broken.

Proposition 1 (Noise sensitivity of f)**.** *Let $f : \{\pm 1\}^n \to \{\pm 1\}$ be opt_s-close to a size-s decision tree T. For all $p \in (0,1)$, we have $\mathrm{NS}_p(f) \le p \log s + 2\,\mathrm{opt}_s$.*

Proof. Let $\boldsymbol{x} \sim \{\pm 1\}^n$ be uniform random, $\boldsymbol{y} \sim_p \boldsymbol{x}$ be a p-noisy copy of $\boldsymbol{x}$, and $\boldsymbol{x}^{\oplus i}$ denote $\boldsymbol{x}$ with its i-th coordinate flipped. We first observe that

$$\begin{aligned}\mathrm{NS}_p(f) &= \Pr[f(\boldsymbol{x}) \neq f(\boldsymbol{y})] \\ &\le \Pr[f(\boldsymbol{x}) \neq T(\boldsymbol{x})] + \Pr[T(\boldsymbol{x}) \neq T(\boldsymbol{y})] + \Pr[T(\boldsymbol{y}) \neq f(\boldsymbol{y})] \\ &= \mathrm{NS}_p(T) + 2\,\mathrm{opt}_s.\end{aligned}$$

To bound $\mathrm{NS}_p(T)$, we use the inequality $\mathrm{NS}_p(T) \le p \cdot \mathrm{Inf}(T)$ where $\mathrm{Inf}(T) \coloneqq \sum_{i=1}^n \Pr[f(\boldsymbol{x}) \neq f(\boldsymbol{x}^{\oplus i})]$ is the total influence of T [O'D14, Exercise 2.42], along with the bound $\mathrm{Inf}(T) \le \log s$ (see e.g. [OS07]). □

We prove Lemma 1 by quantifying the difference between $\mathrm{NS}_p(f, T_f^{j+1,p})$ and $\mathrm{NS}_p(f, T_f^{j,p})$: we show that for every $j \in \mathbb{N}$, either $\mathrm{dist}(f, T_f^{j,p}) \le O(\mathrm{opt}_s + \varepsilon)$ or it must be the case that $\mathrm{NS}_p(f, T_f^{j+1,p})$ is significantly smaller than $\mathrm{NS}_p(f, T_f^{j,p})$. Since $\mathrm{NS}_p(f, T) \ge 0$ for all trees T, the second case can only happen so many times before we fall into the first case.

We will need the two-function, real-valued generalization of the O'Donnell, Saks, Schramm, and Servedio inequality [OSSS05]:

Theorem 3 (Theorem 3.2 of [OSSS05])**.** *Let $T : \{\pm 1\}^n \to \{\pm 1\}$ be a decision tree. For all functions $g : \{\pm 1\}^n \to \mathbb{R}$, writing $\boldsymbol{x}, \boldsymbol{x}' \sim \{\pm 1\}^n$ to denote uniform random and independent inputs and $\boldsymbol{x}^{\sim i}$ to denote $\boldsymbol{x}$ with its i-th coordinate rerandomized,*

$$\mathrm{CoVr}(T, g) \le \sum_{i=1}^n \lambda_i(T) \cdot \mathop{\mathbb{E}}_{\boldsymbol{x}} \left[|g(\boldsymbol{x}) - g(\boldsymbol{x}^{\sim i})| \right],$$

where

$$\begin{aligned}\mathrm{CoVr}(T, f) &\coloneqq \mathop{\mathbb{E}}_{\boldsymbol{x}, \boldsymbol{x}'} \left[|T(\boldsymbol{x}) - g(\boldsymbol{x}')| \right] - \mathop{\mathbb{E}}_{\boldsymbol{x}} \left[|T(\boldsymbol{x}) - g(\boldsymbol{x})| \right], \\ \lambda_i(T) &\coloneqq \Pr[\, T \text{ queries } \boldsymbol{x}_i \,].\end{aligned}$$

By applying Theorem 3 to a suitably smoothened version of f, we are able to derive a lower bound on the score of the highest-scoring variable of f.

Definition 5 (p-smoothed version of f)**.** *For $f : \{\pm 1\}^n \to \{\pm 1\}$ and $p \in (0,1)$, the p-smoothed version of f is the function $\tilde{f}^{(p)} : \{\pm 1\}^n \to [-1, 1]$,*

$$\tilde{f}^{(p)}(x) = \mathop{\mathbb{E}}_{\boldsymbol{y} \sim_p x} [f(\boldsymbol{y})] = \sum_{S \subseteq [n]} (1-p)^{|S|} \widehat{f}(S) \prod_{i \in S} x_i,$$

where the $\widehat{f}(S)$ is the S-th Fourier coefficients of f. When p is clear from context, we write $\tilde{f}$.

Lemma 2 (Score of the highest-scoring variable)**.** *Let $f : \{\pm 1\}^n \to \{\pm 1\}$ be a function, $p \in (0,1)$, and $\tilde{f} = \tilde{f}^{(p)}$ be its p-smoothed version. For all size-s decision trees $T : \{\pm 1\}^n \to \{\pm 1\}$,*

$$\max_{i \in [n]} \left\{ \sqrt{\mathrm{Score}_i(f, p)} \right\} \ge \frac{\sqrt{p}}{\log s} \cdot \left(\tfrac{1}{2} \mathrm{Var}(\tilde{f}) - \mathbb{E} \left[|T(\boldsymbol{x}) - \tilde{f}(\boldsymbol{x})| \right] \right).$$

Proof. Applying Theorem 3 with 'g' being the p-smoothed version $\tilde{f}$ of f, we have

$$\mathrm{CoVr}(T, \tilde{f}) \le \sum_{i=1}^{n} \lambda_i(T) \cdot \mathbb{E}[|\tilde{f}(\boldsymbol{x}) - \tilde{f}(\boldsymbol{x}^{\sim i})|]. \tag{1}$$

We first lowerbound the LHS of Equation (1). For $\boldsymbol{x}, \boldsymbol{x}' \sim \{\pm 1\}^n$ uniform and independent,

$$\begin{aligned}
\mathrm{CoVr}(T, \tilde{f}) &= \mathbb{E}\left[|T(\boldsymbol{x}) - \tilde{f}(\boldsymbol{x}')|\right] - \mathbb{E}\left[|T(\boldsymbol{x}) - \tilde{f}(\boldsymbol{x})|\right] && \text{(Definition of CoVr)} \\
&\ge \mathbb{E}\left[|\tilde{f}(\boldsymbol{x}) - \tilde{f}(\boldsymbol{x}')|\right] - 2\,\mathbb{E}\left[|T(\boldsymbol{x}) - \tilde{f}(\boldsymbol{x})|\right] && \text{(Triangle inequality)} \\
&\ge \tfrac{1}{2}\,\mathbb{E}\left[(\tilde{f}(\boldsymbol{x}) - \tilde{f}(\boldsymbol{x}'))^2\right] - 2\,\mathbb{E}\left[|T(\boldsymbol{x}) - \tilde{f}(\boldsymbol{x})|\right] && (\tilde{f} \text{ is } [-1,1]\text{-valued}) \\
&\ge \mathrm{Var}(\tilde{f}) - 2\,\mathbb{E}\left[|T(\boldsymbol{x}) - \tilde{f}(\boldsymbol{x})|\right]. && (2)
\end{aligned}$$

For a function $g : \{\pm 1\}^n \to \mathbb{R}$, its *$i$-th discrete derivative* is the function

$$(D_i g)(x) := \tfrac{1}{2}\big(g(x^{i=1}) - g(x^{i=-1})\big) = \sum_{S \ni i} \widehat{g}(S) \prod_{j \in S \setminus \{i\}} x_j,$$

where $x^{i=b}$ denotes x with its i-th coordinate set to b. With this definition in hand, we now analyze the expectation on the RHS of Equation (1). By Jensen's inequality,

$$\mathbb{E}[|\tilde{f}(\boldsymbol{x}) - \tilde{f}(\boldsymbol{x}^{\sim i})|]^2 \le \mathop{\mathbb{E}}_{\boldsymbol{x}}\left[(\tilde{f}(\boldsymbol{x}) - \tilde{f}(\boldsymbol{x}^{\sim i}))^2\right] = \tfrac{1}{2}\mathop{\mathbb{E}}_{\boldsymbol{x}}\left[(\tilde{f}(\boldsymbol{x}^{i=1}) - \tilde{f}(\boldsymbol{x}^{i=-1}))^2\right] = 2\mathop{\mathbb{E}}_{\boldsymbol{x}}\left[D_i\tilde{f}(\boldsymbol{x})^2\right].$$

Applying Plancherel's identity twice,

$$\mathop{\mathbb{E}}_{\boldsymbol{x}}\left[D_i\tilde{f}(\boldsymbol{x})^2\right] = \sum_{S \ni i}(1-p)^{2|S|}\widehat{f}(S)^2 \le \sum_{S \ni i}(1-p)^{|S|}\widehat{f}(S)^2 = \mathop{\mathbb{E}}_{\boldsymbol{x},\boldsymbol{y}}\left[D_i f(\boldsymbol{x}) D_i f(\boldsymbol{y})\right]$$

where $\boldsymbol{y} \sim_p \boldsymbol{x}$ is a p-noisy copy of $\boldsymbol{x}$. It follows from a straightforward calculation [BGLT20b, Lemma 3.2] that

$$\mathop{\mathbb{E}}_{\boldsymbol{x},\boldsymbol{y}}\left[D_i f(\boldsymbol{x}) D_i f(\boldsymbol{y})\right] = \frac{2 \cdot \mathrm{Score}_i(f,p)}{p}.$$

Therefore, combining the three equations above we have shown that

$$\mathop{\mathbb{E}}_{\boldsymbol{x}}\left[|\tilde{f}(\boldsymbol{x}) - \tilde{f}(\boldsymbol{x}^{\sim i})|\right] \le \sqrt{\frac{4 \cdot \mathrm{Score}_i(f,p)}{p}}.$$

Plugging this inequality into the RHS of Equation (1),

$$\begin{aligned}
\sum_{i=1}^{n} \lambda_i(T) \cdot \mathop{\mathbb{E}}_{\boldsymbol{x}}\left[|\tilde{f}(\boldsymbol{x}) - \tilde{f}(\boldsymbol{x}^{\sim i})|\right] &= \sum_{i=1}^{n} \lambda_i(T) \cdot \sqrt{\frac{4 \cdot \mathrm{Score}_i(f,p)}{p}} \\
&\le \max_{i \in [n]}\left\{\sqrt{\mathrm{Score}_i(f,p)}\right\} \cdot \frac{2}{\sqrt{p}} \cdot \sum_{i=1}^{n} \lambda_i(T) \\
&\le \max_{i \in [n]}\left\{\sqrt{\mathrm{Score}_i(f,p)}\right\} \cdot \frac{2\log s}{\sqrt{p}}, && (3)
\end{aligned}$$

where the final inequality holds because

$$\sum_{i=1}^{n} \lambda_i(T) = \sum_{i=1}^{n} \Pr[\,T \text{ queries } \boldsymbol{x}_i\,] = \mathop{\mathbb{E}}_{\boldsymbol{\ell} \sim T}\left[|\boldsymbol{\ell}|\right] \le \log s.$$

The lemma follows by combining Equations (1) to (3). □

2.1.1 Proof of Lemma 1

Let $T^\star$ be the size-s decision tree that f is opt_s-close to. Fix $j \in \mathbb{N}$ and consider the tree $T_f^{j,p}$. We have that:

$$\begin{aligned}
\mathrm{NS}_p(f, T_f^{j+1,p}) &\le \mathrm{NS}_p(f, T_f^{j,p}) - \mathop{\mathbb{E}}_{\boldsymbol{\ell} \sim T_f^{j,p}} \left[\max_{i \in [n]} \left\{ \mathrm{Score}_i(f_{\boldsymbol{\ell}}, p) \right\} \right] && \text{(Definition 2)} \\
&\le \mathrm{NS}_p(f, T_f^{j,p}) - \left(\mathop{\mathbb{E}}_{\boldsymbol{\ell} \sim T_f^{j,p}} \left[\max_{i \in [n]} \left\{ \sqrt{\mathrm{Score}_i(f_{\boldsymbol{\ell}}, p)} \right\} \right] \right)^2. && \text{(Jensen's inequality)}
\end{aligned}$$

Recall that we write $\boldsymbol{\ell} \sim T$ to denote a draw of a leaf of T where each leaf ℓ receives weight $2^{-|\ell|}$. We consider two cases:

Case 1: $\mathbb{E}_{\boldsymbol{\ell} \sim T_f^{j,p}}[\mathrm{Var}(\widetilde{f}_{\boldsymbol{\ell}})] \ge 2\left(\mathbb{E}_{\boldsymbol{\ell}, \boldsymbol{x}}\left[|T^\star(\boldsymbol{x}) - \widetilde{f}_{\boldsymbol{\ell}}(\boldsymbol{x})|\right] + \varepsilon\right)$.

In this case we apply Lemma 2 to each leaf ℓ of $T_f^{j,p}$ to get that

$$\begin{aligned}
\mathop{\mathbb{E}}_{\boldsymbol{\ell} \sim T_f^{j,p}} \left[\max_{i \in [n]} \left\{ \sqrt{\mathrm{Score}_i(f_{\boldsymbol{\ell}}, p)} \right\} \right] &\ge \frac{\sqrt{p}}{\log s} \cdot \mathop{\mathbb{E}}_{\boldsymbol{\ell} \sim T_f^{j,p}} \left[\tfrac{1}{2} \mathrm{Var}(\tilde{f}_{\boldsymbol{\ell}}) - \mathbb{E}\left[|T^\star(\boldsymbol{x}) - \tilde{f}_{\boldsymbol{\ell}}(\boldsymbol{x})| \right] \right] \\
&\ge \frac{\varepsilon \sqrt{p}}{\log s},
\end{aligned}$$

and hence

$$\begin{aligned}
\mathrm{NS}_p(f, T_f^{j+1,p}) &\le \mathrm{NS}_p(f, T_f^{j,p}) - \frac{\varepsilon^2 p}{(\log s)^2}. \\
&= \mathrm{NS}_p(f, T_f^{j,p}) - \frac{\varepsilon^3}{(\log s)^3}. && \text{(Our choice of } p = \varepsilon/(\log s)\text{)}
\end{aligned}$$

Case 2: $\mathbb{E}_{\boldsymbol{\ell} \sim T_f^{j,p}}[\mathrm{Var}(\widetilde{f}_{\boldsymbol{\ell}})] < 2\left(\mathbb{E}_{\boldsymbol{\ell}, \boldsymbol{x}}\left[|T^\star(\boldsymbol{x}) - \widetilde{f}_{\boldsymbol{\ell}}(\boldsymbol{x})|\right] + \varepsilon\right)$**.**

In this case we claim that $\mathrm{dist}(f, T_f^{j,p}) \le O(\mathrm{opt}_s + \varepsilon)$. We will need a couple of simple propositions:

Proposition 2. $\mathbb{E}_{\boldsymbol{\ell}, \boldsymbol{x}}[(\widetilde{f}_{\boldsymbol{\ell}}(\boldsymbol{x}) - f_\ell(\boldsymbol{x}))^2] \le 4\,\mathrm{NS}_p(f)$.

Proof. Since f_ℓ and $\widetilde{f}_\ell$ are $[-1, 1]$-valued, we have that

$$\begin{aligned}
\mathop{\mathbb{E}}_{\boldsymbol{\ell}, \boldsymbol{x}}\left[(\widetilde{f}_{\boldsymbol{\ell}}(\boldsymbol{x}) - f_{\boldsymbol{\ell}}(\boldsymbol{x}))^2\right] &\le 2 \mathop{\mathbb{E}}_{\boldsymbol{\ell}, \boldsymbol{x}}\left[|\widetilde{f}_{\boldsymbol{\ell}}(\boldsymbol{x}) - f_{\boldsymbol{\ell}}(\boldsymbol{x})|\right] \\
&= 2 \mathop{\mathbb{E}}_{\boldsymbol{\ell}} \left[\mathop{\mathbb{E}}_{\substack{\boldsymbol{x} \\ \boldsymbol{y} \sim_p \boldsymbol{x}}} \left[|(f_{\boldsymbol{\ell}})(\boldsymbol{y}) - f_{\boldsymbol{\ell}}(\boldsymbol{x})| \right] \right] \\
&= 2 \mathop{\mathbb{E}}_{\boldsymbol{\ell}} \left[2 \Pr_{\substack{\boldsymbol{x} \\ \boldsymbol{y} \sim_p \boldsymbol{x}}} \left[f_{\boldsymbol{\ell}}(\boldsymbol{y}) \ne f_{\boldsymbol{\ell}}(\boldsymbol{x}) \right] \right] \\
&= 4\,\mathbb{E}_{\boldsymbol{\ell}}\left[\mathrm{NS}_p(f_{\boldsymbol{\ell}})\right] \\
&= 4\,\mathrm{NS}_p(f, T_f^{j,p}) \le 4\,\mathrm{NS}_p(f),
\end{aligned}$$

where the final inequality is a consequence of the fact that score is a nonnegative quantity (Section 2.1). □

Proposition 3. *For any function $g : \{\pm 1\}^n \to \{\pm 1\}$ and constant $c \in \mathbb{R}$,*

$$\mathbb{E}\left[(g(\boldsymbol{x}) - \operatorname{sign}(\mathbb{E}[g])^2\right] \leq 2\,\mathbb{E}[(g(\boldsymbol{x}) - c)^2].$$

Proof. Let $a \coloneqq \Pr[g(\boldsymbol{x}) = 1]$ and assume without loss of generality that $a \geq \frac{1}{2}$. On one hand, we have that $\mathbb{E}\left[(g(\boldsymbol{x}) - \operatorname{sign}(\mathbb{E}[g])^2\right] = \mathbb{E}\left[(g(\boldsymbol{x}) - 1)^2\right] = 4(1-a)$. On the other hand, since

$$\mathbb{E}[(g(\boldsymbol{x}) - c)^2] = a(1-c)^2 + (1-a)(1+c)^2$$

this quantity is minimized for $c = 2a - 1$ and attains value $4a(1-a)$ at this minimum. Therefore indeed

$$\min_{c \in \mathbb{R}} \left\{ \mathbb{E}[(g(\boldsymbol{x}) - c)^2] \right\} = 4a(1-a) \geq 2(1-a) = \tfrac{1}{2}\,\mathbb{E}\left[(g(\boldsymbol{x}) - \operatorname{sign}(\mathbb{E}[g])^2\right]$$

and the proposition follows. □

With Propositions 2 and 3 in hand, we now bound $\operatorname{dist}(f, T_f^{j,p})$:

$$\begin{aligned}
\operatorname{dist}(f, T_f^{j,p}) &= \mathop{\mathbb{E}}_{\boldsymbol{\ell} \sim T_f^{j,p}}\left[\operatorname{dist}(f_{\boldsymbol{\ell}}, \operatorname{sign}(\mathbb{E}[f_{\boldsymbol{\ell}}]))\right] \\
&= \tfrac{1}{4} \mathop{\mathbb{E}}_{\boldsymbol{\ell}, \boldsymbol{x}}\left[(f_{\boldsymbol{\ell}}(\boldsymbol{x}) - \operatorname{sign}(\mathbb{E}[f_{\boldsymbol{\ell}}]))^2\right] \\
&\leq \tfrac{1}{2} \mathop{\mathbb{E}}_{\boldsymbol{\ell}, \boldsymbol{x}}\left[(f_{\boldsymbol{\ell}}(\boldsymbol{x}) - \mathbb{E}[\widetilde{f}_{\boldsymbol{\ell}}])^2\right] && \text{(Proposition 3)} \\
&\leq \mathop{\mathbb{E}}_{\boldsymbol{\ell}, \boldsymbol{x}}\left[(f_{\boldsymbol{\ell}}(\boldsymbol{x}) - \widetilde{f}_{\boldsymbol{\ell}}(\boldsymbol{x}))^2\right] + \mathop{\mathbb{E}}_{\boldsymbol{\ell}, \boldsymbol{x}}\left[(\widetilde{f}_{\boldsymbol{\ell}}(\boldsymbol{x}) - \mathbb{E}[\widetilde{f}_{\boldsymbol{\ell}}])^2\right] && \text{(“almost-triangle” inequality)} \\
&\leq 4\,\mathrm{NS}_p(f) + \mathop{\mathbb{E}}_{\boldsymbol{\ell}}[\operatorname{Var}(\widetilde{f}_{\boldsymbol{\ell}})]. && \text{(Proposition 2)}
\end{aligned}$$

By the assumption that we are in Case 2,

$$\begin{aligned}
\mathop{\mathbb{E}}_{\boldsymbol{\ell}}[\operatorname{Var}(\widetilde{f}_{\boldsymbol{\ell}})] &< 2 \mathop{\mathbb{E}}_{\boldsymbol{\ell}, \boldsymbol{x}}\left[|T^\star(\boldsymbol{x}) - \widetilde{f}_{\boldsymbol{\ell}}(\boldsymbol{x})|\right] + 2\,\varepsilon \\
&\leq 2\Big(\mathop{\mathbb{E}}_{\boldsymbol{\ell}, \boldsymbol{x}}\left[|T^\star(\boldsymbol{x}) - f_{\boldsymbol{\ell}}(\boldsymbol{x})|\right] + \mathop{\mathbb{E}}_{\boldsymbol{\ell}, \boldsymbol{x}}\left[|f_{\boldsymbol{\ell}}(\boldsymbol{x}) - \widetilde{f}_{\boldsymbol{\ell}}(\boldsymbol{x})|\right] \Big) + 2\,\varepsilon && \text{(Triangle inequality)} \\
&\leq O\big(\mathop{\mathbb{E}}_{\boldsymbol{\ell}}[\operatorname{dist}(f_{\boldsymbol{\ell}}, T^\star)] + \mathrm{NS}_p(f) \big) + 2\,\varepsilon && \text{(Proposition 2)} \\
&\leq O\big(\mathrm{opt}_s + \varepsilon + \mathrm{NS}_p(f)\big) && (\operatorname{dist}(f, T^\star) = \mathrm{opt}_s) \\
&\leq O(\mathrm{opt}_s + p \log s + \varepsilon) && \text{(Proposition 1)} \\
&= O(\mathrm{opt}_s + \varepsilon). && \text{(Our choice of } p = \varepsilon / \log s)
\end{aligned}$$

Summarizing what we have shown through Cases 1 and 2, for all $j \in \mathbb{N}$, we either have

$$\mathrm{NS}_p(f, T_f^{j+1,p}) \leq \mathrm{NS}_p(f, T_f^{j,p}) - \frac{\varepsilon^3}{(\log s)^3}$$

or it must be the case that $\operatorname{dist}(f, T_f^{j,p}) \leq O(\mathrm{opt}_s + \varepsilon)$. Since $\mathrm{NS}_p(f, T) \in [0, 1]$ for all decision trees T, we must fall into the latter case for some $j \leq O((\log s)^3/\varepsilon^3)$. Finally, since $\operatorname{dist}(f, T_f^{j+1,p}) \leq \operatorname{dist}(f, T_f^{j,p})$ for all $j \in \mathbb{N}$, we conclude that $\operatorname{dist}(f, T_f^{d,p}) \leq O(\mathrm{opt}_s + \varepsilon)$ for our choice of $d = O((\log s)^3/\varepsilon^3)$, and Lemma 1 follows.

Remark 2. Lemma 1 concerns the tree $T_f^{d,p}$ as defined in Definition 3, where each internal node v of $T_f^{d,p}$ is a query the variable x_i that maximizes $\mathrm{Score}_i(f_v, p)$. For the algorithmic component of Theorem 1, we will need a robust version of Lemma 1. An inspection of its proof shows that the same statement holds for any tree where each internal node v is a query to a variable of *approximately* maximal score, within $\tau := O(\varepsilon^3/(\log s)^3)$ of $\max_{j\in[n]} \mathrm{Score}_j(f_v, p)$. Indeed, the only change to the proof will be that for all $j \in \mathbb{N}$, we either have that

$$\mathrm{NS}_p(f, T_f^{j+1,p}) \le \mathrm{NS}_p(f, T_f^{j,p}) - \frac{\varepsilon^3}{(\log s)^3} + \tau$$

or it must be the case that $\mathrm{dist}(f, T_f^{j,p}) \le O(\mathrm{opt}_s + \varepsilon)$. Therefore, as long as $\tau \le O(\varepsilon^3/(\log s)^3)$ the conclusion is unaffected. Similarly, instead of labeling every leaf ℓ with $\mathrm{sign}(\mathbb{E}[f_\ell])$, the same conclusion holds if we only require this for leaves ℓ such that $|\mathbb{E}[f_\ell]| > \varepsilon$.

Lemma 3 (Robust version of Lemma 1)**.** *Let $f : \{\pm1\}^n \to \{\pm1\}$ be opt_s-close to a size-s decision tree. For $d = O((\log s)^3/\varepsilon^3)$, $p = \varepsilon/(\log s)$, and $\tau = O(\varepsilon^3/(\log s)^3)$, let T be any complete decision tree of depth d satisfying:*

- *At every internal node v, the variable x_i that is queried at this node satisfies:*

$$\mathrm{Score}_i(f_v, p) \ge \max_{j\in[n]} \{\mathrm{Score}_j(f_v, p)\} - \tau.$$

- *Every leaf ℓ such that $|\mathbb{E}[f_\ell]| > \varepsilon$ is labeled $\mathrm{sign}(\mathbb{E}[f_\ell])$.*

Then $\mathrm{dist}(f, T) \le O(\mathrm{opt}_s + \varepsilon)$.

2.2 Algorithmic component of Theorem 1

2.2.1 Query-efficient simultaneous score estimation

We begin by designing a query-efficient subroutine that simultaneously estimates the scores of all n variables of a function f. The fact that we are able to do so with $O(\log n)$ queries, as opposed to $\Omega(n)$ as would be required by a naive approach, will be a key component in the query efficiency of our reconstructor.

Theorem 4 (Score estimator)**.** *There is an algorithm which, given query access to a function $f : \{\pm1\}^n \to \{\pm1\}$, noise rate $p \in (0, 1)$, accuracy parameter $\tau \in (0, 1)$, and confidence parameter $\delta \in (0, 1)$, for*

$$q = O\left(\frac{\log n + \log(1/\delta)}{\tau^2}\right)$$

makes $O(q)$ queries, runs in $O(qn)$ time, and returns estimates $\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_n$ such that, with probability at least $1 - \delta$, satisfies

$$\left|\boldsymbol{\eta}_i - \mathrm{Score}_i(f, p)\right| < \tau \quad \text{for all } i \in [n].$$

We prove Theorem 4 by first giving a 2-query algorithm, UnbiasedEstimator (Figure 1), that runs in $O(n)$ time and outputs unbiased estimates of all n scores. The algorithm of Theorem 4 takes the mean of multiple runs of that unbiased estimator, with its guarantees following from a simple concentration bound.

UnbiasedEstimator(f, p):

Input: Query access to a function $f : \{\pm 1\}^n \to \{\pm 1\}$ and a noise rate $p \in (0, 1)$.

Output: Unbiased estimates of $\mathrm{Score}_i(f, p)$ for all $i \in [n]$.

1. Choose $\boldsymbol{x} \in \{\pm 1\}^n$ uniformly at random and generate a p-noisy copy $\boldsymbol{y}$ of $\boldsymbol{x}$.
2. For each $i \in [n]$, return the estimate

$$\boldsymbol{\eta}_i = \mathbb{1}\left[f(\boldsymbol{x}) \neq f(\boldsymbol{y})\right] \cdot \left(1 - \frac{1}{1 - \frac{p}{2}} \cdot \mathbb{1}[\boldsymbol{x}_i = \boldsymbol{y}_i]\right).$$

Figure 1: UnbiasedEstimator computes unbiased estimates of the scores of all variables of a function f.

Lemma 4 (Analysis of UnbiasedEstimator)**.** *For any $f : \{\pm 1\}^n \to \{\pm 1\}$ and $p \in (0, 1)$, let $\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_n$ be the outputs of* UnbiasedEstimator(f, p)*. Then*

$$\mathop{\mathbb{E}}_{\boldsymbol{x},\boldsymbol{y}}[\boldsymbol{\eta}_i] = \mathrm{Score}_i(f, p) \quad \textit{for all } i \in [n].$$

Proof. We first note that $\Pr[f(\boldsymbol{x}) \neq f(\boldsymbol{y})]$ is $\mathrm{NS}_p(f)$ by definition. Therefore, it is enough for us to prove that

$$\mathop{\mathbb{E}}_{\boldsymbol{b}\in\{\pm 1\}}\left[\mathrm{NS}_p(f_{x_i=\boldsymbol{b}})\right] = \frac{1}{1 - \frac{p}{2}} \cdot \Pr_{\boldsymbol{x},\boldsymbol{y}}\left[f(\boldsymbol{x}) \neq f(\boldsymbol{y}) \text{ and } \boldsymbol{x}_i = \boldsymbol{y}_i\right]. \tag{4}$$

Given the above equation, the desired result holds by linearity of expectation and the definition of score. Consider the distribution over $(\boldsymbol{x}, \boldsymbol{y})$ conditioned on the event that $b = \boldsymbol{x}_i = \boldsymbol{y}_i$. That distribution is equivalent to if we picked $\boldsymbol{x}$ randomly from the domain of $f_{x_i=b}$ and selected $\boldsymbol{y}$ by rerandomizing each coordinate in that domain with probability p. Therefore,

$$\begin{aligned}
\mathrm{NS}_p(f_{x_i=b}) &= \Pr_{\boldsymbol{x},\boldsymbol{y}}[f(\boldsymbol{x}) \neq f(\boldsymbol{y}) \,|\, b = \boldsymbol{x}_i = \boldsymbol{y}_i] \\
&= \frac{1}{\Pr_{\boldsymbol{x},\boldsymbol{y}}[b = \boldsymbol{x}_i = \boldsymbol{y}_i]}] \cdot \Pr_{\boldsymbol{x},\boldsymbol{y}}[f(\boldsymbol{x}) \neq f(\boldsymbol{y}) \text{ and } b = \boldsymbol{x}_i = \boldsymbol{y}_i].
\end{aligned}$$

We now prove Equation (4):

$$\begin{aligned}
\mathop{\mathbb{E}}_{\boldsymbol{b}\in\{\pm 1\}}\left[\mathrm{NS}_p(f_{x_i=\boldsymbol{b}})\right] &= \mathop{\mathbb{E}}_{\boldsymbol{b}\in\{\pm 1\}}\left[\frac{1}{\Pr_{\boldsymbol{x},\boldsymbol{y}}[\boldsymbol{b} = \boldsymbol{x}_i = \boldsymbol{y}_i]}] \cdot \Pr_{\boldsymbol{x},\boldsymbol{y}}[f(\boldsymbol{x}) \neq f(\boldsymbol{y}) \text{ and } \boldsymbol{b} = \boldsymbol{x}_i = \boldsymbol{y}_i]\right] \\
&= \frac{1}{\frac{1}{2} \cdot \Pr_{\boldsymbol{x},\boldsymbol{y}}[\boldsymbol{x}_i = \boldsymbol{y}_i]} \mathop{\mathbb{E}}_{\boldsymbol{b}\in\{\pm 1\}}\left[\Pr_{\boldsymbol{x},\boldsymbol{y}}[f(\boldsymbol{x}) \neq f(\boldsymbol{y}) \text{ and } \boldsymbol{b} = \boldsymbol{x}_i = \boldsymbol{y}_i]\right] \\
&= \frac{1}{\frac{1}{2} \cdot (1 - \frac{p}{2})} \cdot \tfrac{1}{2} \cdot \Pr_{\boldsymbol{x},\boldsymbol{y}}\left[f(\boldsymbol{x}) \neq f(\boldsymbol{y}) \text{ and } \boldsymbol{x}_i = \boldsymbol{y}_i\right] \\
&= \frac{1}{1 - \frac{p}{2}} \cdot \Pr_{\boldsymbol{x},\boldsymbol{y}}\left[f(\boldsymbol{x}) \neq f(\boldsymbol{y}) \text{ and } \boldsymbol{x}_i = \boldsymbol{y}_i\right].
\end{aligned}$$

Lemma 4 then holds by linearity of expectation. □

We now prove Theorem 4.

Proof of Theorem 4. The algorithm runs UnbiasedEstimator(f, p) q times and then outputs the means of each returned estimates. Each estimate from UnbiasedEstimator is bounded between -1 and 1. By Hoeffding's inequality, for any $i \in [n]$,

$$\Pr\left[\left|\boldsymbol{\eta}_i - \mathrm{Score}_i(f,p)\right| \geq \tau\right] \leq -\exp_e\left(-\frac{q\cdot\tau^2}{2}\right).$$

For q as in Theorem 4, the above probability is at most δ/n. By union bound, all estimates are accurate within $\pm\tau$ with probability at least $1-\delta$.

Finally, this algorithm uses only $2q = O(q)$ queries. Each run of UnbiasedEstimator estimator takes $O(n)$ time to construct the query and compute all the estimates, so the entire algorithm takes $O(qn)$ time. □

2.2.2 Proof of Theorem 1

We prove Theorem 1 by providing an algorithm, Reconstructor (Figure 2), which assumes query access to a function $f : \{\pm1\}^n \to \{\pm1\}$ and provides fast query access to a tree T meeting the criteria of Lemma 3. We build off a simple observation that also underlies [BGLT20a]: to determine the output of a decision tree T on a particular input z, it suffices to build the root-to-leaf path corresponding to z, which can be exponentially faster than building the entire tree. Our algorithm is different from [BGLT20a]'s; as mentioned in the introduction their algorithm is tailored to monotone functions, and is known to fail for non-monotone ones. We on the other hand leverage the specific structure of T established in Section 2.1 together with the query-efficient score estimator from Section 2.2.1 in our design and analysis of Reconstructor.

Reconstructor maintains a partial tree T° containing all the root-to-leaf paths in T corresponding to queries received so far. In the pseudocode for Reconstructor, we use the notation $T^\circ_{\mathrm{internal}}(\alpha) \in [n] \cup \{\varnothing\}$ to indicate the variable queried in $[n]$ at internal node α of the partial tree T°, or $\varnothing$ if that node has not yet been built. Similarly, $T^\circ_{\mathrm{leaf}}(\alpha) \in \{-1, 1, \varnothing\}$ indicates the value at leaf α in T°, or $\varnothing$ if that value has not yet been decided.

Theorem 1 follows from the following two lemmas, showing the correctness and efficiency of Reconstructor respectively.

Lemma 5 (Correctness of Reconstructor)**.** *For any $f : \{\pm1\}^n \to \{\pm1\}$, $s \in \mathbb{N}$, $\varepsilon \in (0, \frac{1}{2})$, $\delta \in (0,1)$, and sequence of inputs $z^{(1)}, \ldots, z^{(m)} \in \{\pm1\}^n$, the outputs of* Reconstructor *are consistent with some decision tree T where*

- *T has size $s^{O((\log s)^2/\varepsilon^3)}$,*
- *$\mathrm{dist}(T, f) \leq O(\mathrm{opt}_s) + \varepsilon$ with probability at least $1-\delta$.*

Proof. The outputs of Reconstructor are always consistent with T° and the depth of T° is always capped at d. Let T be the tree that T° would be if every $x \in \{\pm1\}^n$ were given as an input to Reconstructor. Then, T has size at most $2^d = s^{O((\log s)^2/\varepsilon^3)}$, and every output is consistent with T.

RECONSTRUCTOR$(f, s, \varepsilon, \delta)$:

Input: Query access to a function $f : \{\pm 1\}^n \to \{\pm 1\}$, size parameter s, error parameter ε, and failure probability δ.

Output: Query access to a decision tree T that satisfies $\mathrm{dist}(f, T) \leq O(\mathrm{opt}_s) + \varepsilon$ with probability at least $1 - \delta$.

1. Set parameters d, p, and τ as in Lemma 3.
2. Initialize T° to be the empty partial tree.
3. Upon receiving an input $z \in \{\pm 1\}^n$:
   (a) Initialize α to be the root of T°.
   (b) Repeat d times.
      i. If $T^\circ_{\text{internal}}(\alpha)$ is $\varnothing$ use the estimator from Theorem 4 to compute estimates of $\mathrm{Score}_i(f_\alpha, p)$ with additive accuracy $\pm\frac{\tau}{2}$ and failure probability $O(\frac{\delta}{2^d})$ for all $i \in [n]$ and set $T^\circ_{\text{internal}}(\alpha)$ to the variable with highest estimated score.
      ii. For $i = T^\circ_{\text{internal}}(\alpha)$, If $\overline{x}_i$ is 1, set α to its right child. Otherwise, set α to its left child.
   (c) If $T^\circ_{\text{leaf}}(\alpha)$ is $\varnothing$, use random samples to estimate $\mathbb{E}[f_\ell]$ to additive accuracy $\pm\frac{\varepsilon}{4}$ with failure probability $O(\frac{\delta}{2^d})$ and set $T^\circ_{\text{leaf}}(\alpha)$ to whichever of $\{\pm 1\}$ that estimate is closer to.
   (d) Output $T^\circ_{\text{leaf}}(\alpha)$.

Figure 2: RECONSTRUCTOR gives efficient query access to a decision tree is close to f with high probability.

If all score estimates in Step 3(b)i are accurate to $\pm\frac{\tau}{2}$ and expectation estimates is Step 3c are accurate to $\pm\frac{\varepsilon}{4}$, then T meets the criteria of Lemma 3 and therefore $\mathrm{dist}(T, f) \leq O(\mathrm{opt}_s) + \varepsilon$. The number of time scores are estimated in Step 3(b)i is at most the number of internal nodes of T, which is $2^d - 1$. Similarly, the number of expectation estimates in Step 3(b)i is at most the number of leaves of T, which is 2^d. By union bound over the possible failures, we see that the failure probability is at most δ. □

Lemma 6 (Efficiency of RECONSTRUCTOR)**.** *For any $f : \{\pm 1\}^n \to \{\pm 1\}$, $s \in \mathbb{N}$, $\varepsilon \in (0, \frac{1}{2})$, $\delta \in (0, 1)$, particular input $z \in \{\pm 1\}^n$, and*

$$q = O\left(\frac{(\log s)^9 \cdot (\log n) \cdot \log(1/\delta)}{\varepsilon^9}\right),$$

upon receiving z as input, RECONSTRUCTOR$(f, s, \varepsilon, \delta)$ *uses $O(q)$ queries and $O(qn)$ time to return an output.*

Proof. On each input, the estimator from Theorem 4 is used up to d times. Each uses

$$q_{\text{inner}} \coloneqq O\left(\frac{\log n + \log(2^d/\delta)}{\tau^2}\right) = O\left(\frac{\log n + d + \log(1/\delta)}{\tau^2}\right)$$

queries and $O(q_{\text{inner}} n)$ time. By Hoeffding's inequality, it is sufficient to take

$$q_{\text{leaf}} \coloneqq O\left(\frac{\log(2^d/\delta)}{\varepsilon^2}\right) = O\left(\frac{d + \log(1/\delta)}{\varepsilon^2}\right)$$

random samples in Step 3c. Therefore, the total number of queries used is

$$\begin{aligned} q &= q_{\text{inner}} + q_{\text{leaf}} \\ &= O\left(\frac{\log n + d + \log(1/\delta)}{\tau^2}\right) + O\left(\frac{d + \log(1/\delta)}{\varepsilon^2}\right) \\ &= O\left(\frac{\log n + ((\log s)^3/\varepsilon^3) + \log(1/\delta)}{\varepsilon^6/(\log s)^6} + \frac{((\log s)^3/\varepsilon^2) + \log(1/\delta)}{\varepsilon^2}\right) \\ &= O\left(\frac{(\log s)^9 \cdot (\log n) \cdot \log(1/\delta)}{\varepsilon^9}\right). \end{aligned}$$

The time to prepare all queries is $O(qn)$, and all other computation is asymptotically faster. □

Remark 3 (Local reconstruction)**.** We remark that our reconstruction algorithm can be made local in the sense of [SS10]. They define a reconstruction algorithm, $\mathcal{A}$, to be local, if the output of $\mathcal{A}$ on some input z is a *deterministic* and easy to compute function of z and some small random string ρ. This allows queries to the reconstructor to be answered in parallel, as long as the random string ρ is shared. To make our reconstructor local, we note that the only place randomness is used is in generating samples consistent with some restriction α. We can set ρ to be n bits per a sample the constructor might wish to generate. Since the total number of samples the reconstructor needs per input is $\text{poly}(\log s, 1/\varepsilon, \log(1/\delta)) \cdot \log n$, we have

$$|\rho| = \text{poly}(\log s, 1/\varepsilon, \log(1/\delta)) \cdot n \log n$$

On a particular input, the local reconstructor starts with T° being the empty tree. Whenever it wishes to produce a random sample consistent with α, it sets the $\boldsymbol{x} \in \{\pm 1\}^n$ to be next n bits of ρ and then uses $\boldsymbol{x}_\alpha$ for the sample. It's easy to see that this algorithm will keep T° consistent between different runs because it will always compute the same variable as having the highest score given some restriction. Furthermore, the analysis goes through without issue. The only difference difference between this analysis and one where fresh random bits are used to for each sample is that the queries of different paths may be correlated. In our proof of Lemma 5, we use a union bound to ensure all estimates obtained through sampling are accurate, and that union bound holds regardless of whether those estimates are independent.

2.3 Proof of Corollary 1

In this section we derive Corollary 1 as a simple consequence of Theorem 1. The connection between reconstruction and tolerant testing has been noted in other works (see e.g. [CGR13, Bra08]); we provide a proof here for completeness.

Corollary 3. *There is an algorithm which, given query access to $f : \{\pm 1\}^n \to \{\pm 1\}$ and parameters $s \in \mathbb{N}$ and $\varepsilon, \delta \in (0,1)$, runs in $\mathrm{poly}(\log s, 1/\varepsilon) \cdot n \log n \cdot \log(1/\delta)$ time, makes $\mathrm{poly}(\log s, 1/\varepsilon) \cdot \log n \cdot \log(1/\delta)$ queries to f, and*

- *Accepts w.p. at least $1-\delta$ if f is ε-close to a size-s decision tree;*
- *Rejects w.p. at least $1-\delta$ if f is $\Omega(\varepsilon)$-far from size-$s^{O((\log s)^2/\varepsilon^3)}$ decision trees.*

Proof. The algorithm chooses m uniform random inputs, $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)} \sim \{\pm 1\}^n$ where $m = O(\log(1/\delta)/\varepsilon^2)$. Let $\boldsymbol{b}^{(1)}, \ldots, \boldsymbol{b}^{(m)} \in \{\pm 1\}$ be the output of $\textsc{Reconstructor}(f, s, \varepsilon, \delta)$. The tester rejects if $\mathbb{E}_{\boldsymbol{i} \in [m]}[f(\boldsymbol{x}^{(\boldsymbol{i})}) \neq \boldsymbol{b}^{(\boldsymbol{i})}] > \Omega(\varepsilon)$ and accepts otherwise.

First, we consider the case where f is ε-close to a size-s decision tree (i.e. $\mathrm{opt}_s \leq \varepsilon$). By Lemma 5, with probability at least $1-\delta$ the outputs of $\textsc{Reconstructor}$ are consistent with a tree, T, satisfying $\mathrm{dist}(T, f) \leq O(\varepsilon)$. By Hoeffding's inequality,

$$\Pr_{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}} \left[\mathop{\mathbb{E}}_{\boldsymbol{i} \in [m]} \left[f(\boldsymbol{x}^{(\boldsymbol{i})}) \neq \boldsymbol{b}^{(\boldsymbol{i})} \right] > \Omega(\varepsilon) \right] \leq \exp(-2m\varepsilon^2) \leq \delta.$$

By a union bound, the tester rejects with probability at most $\delta + \delta = 2\delta$.

We next consider the case where f is $\Omega(\varepsilon)$-far from size-$s^{O((\log s)^2/\varepsilon^3)}$ decision trees. By Lemma 5 it is guaranteed to be consistent. A similar argument to the first case shows that the probability of acceptance is at most $\delta + \exp(-2m\varepsilon^2) = 2\delta$. Finally, the efficiency of this tester is a consequence of Lemma 6 and our choice of $m = O(\log(1/\delta)/\varepsilon^2)$. □

3 Proof of Corollary 2

We first restate Theorem 1 with decision tree *depth* instead of *size* as the complexity measure:

Theorem 5 (Theorem 1 in terms of decision tree depth)**.** *There is a randomized algorithm which, given query access to $f : \{\pm 1\}^n \to \{\pm 1\}$ and parameters $d \in \mathbb{N}$ and $\varepsilon \in (0,1)$, provides query access to a fixed decision tree T where*

- *T has depth $O(d^3/\varepsilon^2)$,*
- *$\mathrm{dist}(T, f) \leq O(\mathrm{opt}_d) + \varepsilon$ w.h.p., where opt_d denotes the distance of f to the closest depth-d decision tree.*

Every query to T is answered in $\mathrm{poly}(d, 1/\varepsilon) \cdot n \log n$ time and with $\mathrm{poly}(d, 1/\varepsilon) \cdot \log n$ queries to f.

To see that our proof of Theorem 1 also establishes Theorem 5, we use the fact that every depth-d decision tree has size $\leq 2^d$, and recall that the tree T that the algorithm of Theorem 1 provides query access to is a complete tree and hence has depth logarithmic in its size.

Decision tree depth and Fourier degree of boolean functions are known to be polynomially related:

Fact 1 (Decision tree depth vs. Fourier degree [Mid04, Tal13])**.** *For $g : \{\pm 1\}^n \to \{\pm 1\}$ let $\deg(g)$ denote g's Fourier degree and $D(g)$ denote the depth of the shallowest decision tree that computes g. Then $\deg(g) \leq D(g)$ and $D(g) \leq \deg(g)^3$.*

We first observe Theorem 5 and Fact 1 already gives a quantitatively weaker version of Corollary 2 where g has degree $O(d^9/\varepsilon^2)$. To see this $f : \{\pm 1\}^n \to \{\pm 1\}$ be opt_d-close to a degree-d function $h : \{\pm 1\}^n \to \{\pm 1\}$. By Fact 1, $D(h) \leq \deg(h)^3$, and so the algorithm of Theorem 5 provides query access to a decision tree $T : \{\pm 1\}^n \to \{\pm 1\}$ that is $(O(\mathrm{opt}_d) + \varepsilon)$-close to f and where the depth of T is $O(D(h)^3/\varepsilon^2) = O(\deg(h)^9/\varepsilon^2)$. Applying Fact 1 again, we conclude that $\deg(T) \leq D(T) \leq O(\deg(h)^9/\varepsilon^2)$.

To obtain the sharper bound of $O(\deg(h)^7/\varepsilon^2)$, we observe that the proof of Lemma 1 in fact bounds the depth of T by $O(D(h)^2 \operatorname{Inf}(h)/\varepsilon^2)$. (Specifically, the proof of Proposition 1 shows that $\mathrm{NS}_p(f) \leq p \cdot \operatorname{Inf}(h) + \operatorname{dist}(f, h)$ for all $f, h : \{\pm 1\}^n \to \{\pm 1\}$ and $p \in (0, 1)$.) Influence and degree of boolean functions are related via the following basic fact (see e.g. [O'D14, Theorem 37]):

Fact 2. *For all $h : \{\pm 1\}^n \to \{\pm 1\}$, we have $\operatorname{Inf}(h) \leq \deg(h)$.*

Therefore, we can bound the degree of T by $O(D(h)^2 \operatorname{Inf}(h)/\varepsilon^2) \leq O(\deg(h)^7/\varepsilon^2)$.

Guarantees for the other measures listed in Table 1 follow from similar calculations and known quantitative relationships between these measures and decision tree complexity; the current best bounds are summarized in Table 1 of [ABK+20].

4 Proof of Theorem 2

In this section, we prove the following theorem:

Theorem 6 (Tolerant testing of DTs $\Rightarrow$ Proper learning of DTs)**.** *Let $c > 0$ be an absolute constant and $\mathcal{A}$ be an algorithm with the following guarantee. Given query access to $f : \{\pm 1\}^n \to \{\pm 1\}$ and parameters $s \in \mathbb{N}$ and $\varepsilon \in (0, 1)$, the algorithm $\mathcal{A}$:*

- *Accepts w.h.p. if f is ε-close to a size-s decision tree;*
- *Rejects w.h.p. if f is $(c\varepsilon)$-far from all size-s decision trees.*

Then there is an algorithm $\mathcal{B}$ with the following guarantee. Given parameters $s' \in \mathbb{N}$ and $\varepsilon' \in (0, 1)$, and query access to a function $g : \{\pm 1\}^n \to \{\pm 1\}$ that is computed by a size-s' decision tree, $\mathcal{B}$ makes $\mathrm{poly}(s', n, 1/\varepsilon')$ calls to $\mathcal{A}$, each with parameters $s \leq s'$ and $\varepsilon \geq \mathrm{poly}(1/s', \varepsilon')$, and produces a decision tree which is ε'-close to g with high probability. Furthermore, the auxiliary computation that g does takes time $\mathrm{poly}(n, s', 1/\varepsilon')$.

Theorem 2 follows as a special case of Theorem 6. In fact, Theorem 6 further proves that any tolerant tester for decision trees running in $\mathrm{poly}(n^{o(\log s)}, 1/\varepsilon)$ time yields a proper learner with the same runtime, which would already improve upon the longstanding state of the art of $\mathrm{poly}(n^{\log s}, 1/\varepsilon)$ [EH89].

We prove Theorem 6 in two steps:

1. A tolerant tester implies an algorithm for estimating the distance of any function to the class of size-s decision trees. This is well known [PRR06] and applies to any function class, not just decision trees.
2. An algorithm for estimating distance to decision trees implies a proper learner for decision trees. Here, we take advantage of the structure of decision trees.

For a function $g : \{\pm 1\}^n \to \{\pm 1\}$ and $s \in \mathbb{N}$, we write $\mathrm{opt}_s(g)$ to denote the distance of g to the closest size-s decision tree.

Lemma 7 (Tolerant testing $\Rightarrow$ distance estimation [PRR06])**.** *Let c and $\mathcal{A}$ be as in Theorem 6. There exists an estimator $\mathcal{E}$ with the following guarantee. Given query access to $g : \{\pm 1\}^n \to \{\pm 1\}$ and parameters $s' \in \mathbb{N}$ and $\gamma \in (0, 1)$, the estimator $\mathcal{E}$ makes c/γ calls to $\mathcal{A}$ and returns an $\boldsymbol{\eta}$ that with high probability satisfies*

$$\boldsymbol{\eta} \le \mathrm{opt}_s(g) \le c \cdot \boldsymbol{\eta} + \gamma.$$

Furthermore, the auxiliary computation of g takes time $O(c/\gamma)$.

Proof. The algorithm $\mathcal{E}$ runs $\mathcal{A}$ with $\varepsilon = \frac{\gamma}{c}, \frac{2\gamma}{c}, \frac{3\gamma}{c}, \ldots, 1$, and sets $\boldsymbol{\eta}$ to be the largest ε for which $\mathcal{A}(g, s, \varepsilon)$ rejects. Since $\mathcal{A}(g, s, \boldsymbol{\eta})$ rejected,

$$\boldsymbol{\eta} < \mathrm{opt}_s(g)$$

with high probability. Furthermore, since $\mathcal{A}(g, s, \boldsymbol{\eta} + \frac{\gamma}{c})$ accepted,

$$\begin{aligned} \mathrm{opt}_s(g) &< c \cdot \left(\boldsymbol{\eta} + \tfrac{\gamma}{c}\right) \\ &= c \cdot \boldsymbol{\eta} + \gamma \end{aligned}$$

with high probability. Finally, we note that $\mathcal{E}$ indeed makes c/γ calls to $\mathcal{A}$, and aside from those calls, it only needs to make a single pass over the output of those calls and return the largest ε that led to a rejection, which takes time $O(c/\gamma)$. □

We are now ready to state our algorithm, BuildDT (Figure 3), for properly learning size-s' decision trees. BuildDT will additionally take in a depth parameter d that will facilitate our analysis of it (looking ahead, d will be chosen to be $O(\log(s'/\varepsilon'))$ in our proof of Theorem 6).

Lemma 8 (Error of BuildDT)**.** *For all functions $f : \{\pm 1\}^n \to \{\pm 1\}$ and parameters $s, d \in \mathbb{N}$ and $\gamma \in (0, 1)$, the algorithm $\textsc{BuildDT}(f, s, d, \gamma)$ outputs a decision tree T satisfying*

$$\mathrm{dist}(T, f) \le c^d \cdot \mathrm{opt}_s(f) + \gamma \cdot \frac{c^d - 1}{c - 1} + \frac{s}{2^{d+2}}. \tag{5}$$

Proof. We proceed by induction on s and d. If $s = 1$, then in Step 1, BuildDT outputs the best decision tree of size 1. Therefore, $\mathrm{dist}(T, f) \le \mathrm{opt}_s(f)$, satisfying Equation (5). If $d = 0$ and $s \ge 2$, then $\frac{s}{2^{d+2}} \ge \frac{2}{4} = \frac{1}{2}$. Furthermore, in Step 1, BuildDT always outputs a tree with error at most $\frac{1}{2}$. Therefore,

$$\mathrm{dist}(T, f) \le \frac{s}{2^{d+2}} \le c^d \cdot \mathrm{opt}_s(f) + \gamma \cdot \frac{c^d - 1}{c - 1} + \frac{s}{2^{d+2}}.$$

Finally, we consider the case where $d \ge 1$ and $s \ge 2$. Let T_{opt} be the size-s decision tree that is $\mathrm{opt}_s(f)$ close to f. Let $x_{i_{\mathrm{opt}}}$ the root of T_{opt}, and $s_{0,\mathrm{opt}}$, $s_{1,\mathrm{opt}}$ the sizes of the left and right subtrees of T_{opt} respectively. Since the estimates computed in Step 2a are underestimates of or equal to the true error (i.e. $\mathrm{error}(i_{\mathrm{opt}}, s_{0,\mathrm{opt}}, s_{1,\mathrm{opt}}) \le \mathrm{opt}_s(f)$), and since $i^\star, s_0^\star, s_1^\star$ are chosen in Step 3 to minimize the estimated error, we have

$$\mathrm{error}(i^\star, s_0^\star, s_1^\star) \le \mathrm{error}(i_{\mathrm{opt}}, s_{0,\mathrm{opt}}, s_{1,\mathrm{opt}}) \le \mathrm{opt}_s(f).$$

BuildDT(f, s, d, γ):

Input: Query access to $f : \{\pm 1\}^n \to \{\pm 1\}$, parameters $s, d \in \mathbb{N}$ and $\gamma \in (0, 1)$.

Output: A size-s depth-d decision tree T.

1. If $s = 1$ or $d = 0$, return round$(\mathbb{E}[f])$.
2. For each $i \in [n]$ and integers $s_0, s_1 \geq 1$ satisfying $s_0 + s_1 = s$:
   (a) Use $\mathcal{E}$ from Lemma 7 to obtain estimates $\boldsymbol{\eta}(x_i = 0, s_0)$ and $\boldsymbol{\eta}(x_i = 1, s_1)$ that satisfy:
   $$\boldsymbol{\eta}(x_i = 0, s_0) \leq \mathrm{opt}_{s_0}(f_{x_i=0}) \leq c \cdot \boldsymbol{\eta}(x_i = 0, s_0) + \gamma;$$
   $$\boldsymbol{\eta}(x_i = 1, s_1) \leq \mathrm{opt}_{s_1}(f_{x_i=1}) \leq c \cdot \boldsymbol{\eta}(x_i = 1, s_1) + \gamma.$$
   (b) Store $\mathrm{error}(i, s_1, s_2) \leftarrow \frac{1}{2}\big(\boldsymbol{\eta}(x_i = 0, s_0) + \boldsymbol{\eta}(x_i = 1, s_1)\big)$.
3. Let $(i^\star, s_0^\star, s_1^\star)$ be the tuple that minimizes $\mathrm{error}(i, s_0, s_1)$. Output the tree with $x_{i^\star}$ as its root, BuildDT$(f_{x_{i^\star}=0}, s_0^\star, d-1, \gamma)$ as its left subtree, and BuildDT$(f_{x_{i^\star}=1}, s_1^\star, d-1, \gamma)$ as its right subtree.

Figure 3: BuildDT computes a size-s depth-d decision tree that approximates a target function $f : \{\pm 1\}^n \to \{\pm 1\}$.

Finally, we bound $\mathrm{dist}(T, f)$. Let T_0 and T_1 be the left and right subtrees of T. Then,

$$\begin{aligned}
\mathrm{dist}(T, f) &= \tfrac{1}{2}\big(\mathrm{dist}(T_0, f_{x_{i^\star}=0}) + \mathrm{dist}(T_1, f_{x_{i^\star}=1})\big) \\
&\leq \tfrac{1}{2}\Big(c^{d-1} \cdot \mathrm{opt}_{s_0^\star}(f_{x_{i^\star}=0}) + \gamma \cdot \frac{c^{d-1}-1}{c-1} + \frac{s_0^\star}{2^{d+1}} \\
&\quad + c^{d-1} \cdot \mathrm{opt}_{s_1^\star}(f_{x_{i^\star}=1}) + \gamma \cdot \frac{c^{d-1}-1}{c-1} + \frac{s_1^\star}{2^{d+1}}\Big) && \text{(Inductive hypothesis)} \\
&= c^{d-1} \cdot \tfrac{1}{2}\left(\mathrm{opt}_{s_0^\star}(f_{x_{i^\star}=0}) + \mathrm{opt}_{s_1^\star}(f_{x_{i^\star}=1})\right) + \gamma \cdot \frac{c^{d-1}-1}{c-1} + \frac{s_0^\star + s_1^\star}{2^{d+2}} \\
&\leq c^{d-1} \cdot \tfrac{1}{2}\big((c \cdot \boldsymbol{\eta}(x_{i^\star} = 0, s_0^\star) + \gamma) + (c \cdot \boldsymbol{\eta}(x_{i^\star} = 1, s_1^\star) + \gamma)\big) + \gamma \cdot \frac{c^{d-1}-1}{c-1} + \frac{s}{2^{d+2}} \\
&= c^d \cdot \tfrac{1}{2}\left(\boldsymbol{\eta}(x_{i^\star} = 0, s_0^\star) + \boldsymbol{\eta}(x_{i^\star} = 1, s_1^\star)\right) + c^{d-1} \cdot \gamma + \gamma \cdot \frac{c^{d-1}-1}{c-1} + \frac{s}{2^{d+2}} \\
&= c^d \cdot \mathrm{error}(i^\star, s_0^\star, s_1^\star) + \gamma \cdot \frac{c^{d-1}(c-1) + c^{d-1} - 1}{c-1} + \frac{s}{2^{d+2}} \\
&\leq c^d \cdot \mathrm{opt}_s(f) + \gamma \cdot \left(\frac{c^d - 1}{c-1}\right) + \frac{s}{2^{d+2}}.
\end{aligned}$$

The desired result holds by induction. □

For readability, Lemma 8 assumes that BuildDT is able to compute round$(\mathbb{E}[f])$ in Step 1. To

make BuildDT efficient, we would only estimate $\mathbb{E}[f]$ by querying f on uniform random inputs $\boldsymbol{x} \in \{\pm 1\}^n$. If those estimates are computed to accuracy ε', then each leaf of our tree can have up to ε' additional error. This is not an issue since it increases the total error of T, which is simply the average of the error at each leaf, by only ε'.

Finally, we prove Theorem 6:

Proof of Theorem 6. Our goal is to properly learn a size-s' decision tree $g : \{\pm 1\}^n \to \{\pm 1\}$ to accuracy ε'. To do so, we run $\textsc{BuildDT}(g, s', d, \gamma)$, with d set to

$$d = \log(s'/\varepsilon') - 1,$$

and γ set to

$$\gamma = \frac{\varepsilon'}{2 \cdot \max(2,c)^d} = \frac{\varepsilon'}{2} \cdot (2^{-d})^{\max(1,\log c)} = \frac{\varepsilon'}{2} \cdot \left(\frac{\varepsilon'}{s'}\right)^{\max(1,\log c)}.$$

By Lemma 8, for T the tree BuildDT outputs,

$$\begin{aligned}
\operatorname{dist}(T,g) &\le c^d \cdot \operatorname{opt}_{s'}(g) + \gamma \cdot \frac{c^d - 1}{c-1} + \frac{s'}{2^{d+2}} \\
&\le 0 + \frac{\varepsilon'}{2 \cdot \max(2,c)^d} \cdot \max(2,c)^d + \frac{s'}{2^{\log(s'/\varepsilon')+1}} \\
&\le \frac{\varepsilon'}{2} + \frac{\varepsilon'}{2} = \varepsilon'.
\end{aligned}$$

Hence, BuildDT produces the desired output. We next argue that it is efficient. During the recursion, BuildDT is called at most s' times in total. Each such call makes $O(ns')$ calls to $\mathcal{E}$. By Lemma 7, those calls to $\mathcal{E}$ each make c/ε calls to $\mathcal{A}$. Hence, the total number of calls to $\mathcal{A}$ is

$$O\left(\frac{n(s')^2}{\gamma}\right) = O\left(\frac{n(s')^2}{\varepsilon'} \cdot \left(\frac{s'}{\varepsilon'}\right)^{\max(1,\log c)}\right) = \operatorname{poly}(n, s', 1/\varepsilon').$$

The total auxiliary computation of BuildDT is bounded by the same quantity. Finally, each call to $\mathcal{A}$ is made with parameters s and ε where $s \le s'$ and $\varepsilon = \frac{\varepsilon'}{2} \cdot \left(\frac{\varepsilon'}{s'}\right)^{\max(1,\log c)} \ge \operatorname{poly}(1/s', \varepsilon')$. □

Acknowledgements

We thank Eric Blais, Nader Bshouty, Ryan O'Donnell, Mingda Qiao, Mert Sağlam, Rocco Servedio, and Erik Waingarten for helpful discussions.

References

[ABF+09] Misha Alekhnovich, Mark Braverman, Vitaly Feldman, Adam Klivans, and Toniann Pitassi. The complexity of properly learning simple concept classes. *Journal of Computer & System Sciences*, 74(1):16–34, 2009. 1.2

[ABK+20] Scott Aaronson, Shalev Ben-David, Robin Kothari, Shravas Rao, and Avishay Tal. Degree vs. approximate degree and quantum implications of huang's sensitivity theorem. *arXiv preprint*, abs/2010.12629, 2020. 3

[ACCL08] Nir Ailon, Bernard Chazelle, Seshadhri Comandur, and Ding Liu. Property-preserving data reconstruction. *Algorithmica*, 51(2):160–182, 2008. 1, 1.2

[AT10] Tim Austin and Terence Tao. Testability and repair of hereditary hypergraph properties. *Random Structures & Algorithms*, 36(4):373–463, 2010. 1, 1.2

[BBM12] Eric Blais, Joshua Brody, and Kevin Matulef. Property testing lower bounds via communication complexity. *computational complexity*, 21(2):311–358, 2012. 1.1, 1.1, 1.2, 1.2, 1.3

[BCE+18] Eric Blais, Clément Canonne, Talya Eden, Amit Levi, and Dana Ron. Tolerant junta testing and the connection to submodular optimization and function isomorphism. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2113–2132, 2018. 1.2

[BGJ+12] Arnab Bhattacharyya, Elena Grigorescu, Madhav Jha, Kyomin Jung, Sofya Raskhodnikova, and David Woodruff. Lower bounds for local monotonicity reconstruction from transitive-closure spanners. *SIAM Journal on Discrete Mathematics*, 26(2):618–646, 2012. 1.2

[BGLT20a] Guy Blanc, Neha Gupta, Jane Lange, and Li-Yang Tan. Estimating decision tree learnability with polylogarithmic sample complexity. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 1.2, 2.2.2

[BGLT20b] Guy Blanc, Neha Gupta, Jane Lange, and Li-Yang Tan. Universal guarantees for decision tree induction via a higher-order splitting criterion. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 1.2, 2, 2.1, 1, 2.1

[BH13] Joshua Brody and Pooya Hatami. Distance-sensitive property testing lower bounds. *CoRR*, abs/1304.6685, 2013. 1.1, 1.2

[BH18] Avrim Blum and Lunjia Hu. Active tolerant testing. In *Proceedings of the 31st Conference On Learning Theory (COLT)*, volume 75, pages 474–497, 2018. 1.2

[Bra08] Zvika Brakerski. Local property restoring, 2008. Manuscript. 2.3

[Bsh20] Nader Bshouty. Almost optimal testers for concise representations. In *Proceedings of the 24th International Conference on Randomization and Computation (RANDOM)*, pages 5:1–5:20, 2020. 1.1, 1.1, 1.2, 1.2, 1.3

[CFM14] Sourav Chakraborty, Eldar Fischer, and Arie Matsliah. Query complexity lower bounds for reconstruction of codes. *Theory of Computing*, 10(19):515–533, 2014. 1.2

[CGR13] Andrea Campagna, Alan Guo, and Ronitt Rubinfeld. Local reconstructors and tolerant testers for connectivity and diameter. In *Proceedings of the 17th International Workshop on Randomization and Computation (RANDOM)*, pages 411–424, 2013. 1.2, 1.2, 2.3

[CGSM11a] Sourav Chakraborty, David García-Soriano, and Arie Matsliah. Efficient sample extractors for juntas with applications. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 545–556, 2011. 1.1, 1.1, 1.2, 1.2, 1.3

[CGSM11b] Sourav Chakraborty, David García-Soriano, and Arie Matsliah. Nearly tight bounds for testing function isomorphism. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1683–1702, 2011. 1.1, 1.2

[CS06] Bernard Chazelle and C Seshadhri. Online geometric reconstruction. In *Proceedings of the 22nd Annual Symposium on Computational Geometry (SoCG)*, pages 386–394, 2006. 1.2

[DLM+07] Ilias Diakonikolas, Homin Lee, Kevin Matulef, Krzysztof Onak, Ronitt Rubinfeld, Rocco Servedio, and Andrew Wan. Testing for concise representations. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 549–558, 2007. 1.1, 1.1, 1.2, 1.2, 1.3

[DLM+08] Ilias Diakonikolas, Homin Lee, Kevin Matulef, Rocco Servedio, and Andrew Wan. Efficiently testing sparse $GF(2)$ polynomials. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 502–514, 2008. 1.2

[EH89] Andrzej Ehrenfeucht and David Haussler. Learning decision trees from random examples. *Information and Computation*, 82(3):231–246, 1989. 1.1, 1.2, 4

[Fel16] Vitaly Feldman. Hardness of proper learning. In *Encyclopedia of Algorithms*, 2016. 1.2

[FKR+04] Eldar Fischer, Guy Kindler, Dana Ron, Shmuel Safra, and Alex Samorodnitsky. Testing juntas. *Journal of Computer and System Sciences*, 68(4):753–787, 2004. 1.2

[GGR98] Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45:653–750, 1998. 1, 1.1

[JR13] Madhav Jha and Sofya Raskhodnikova. Testing and reconstruction of lipschitz functions with applications to data privacy. *SIAM Journal on Computing*, 42(2):700–731, 2013. 1.2

[KM93] Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, December 1993. 1.2

[KNOW14] Pravesh Kothari, Amir Nayyeri, Ryan O'Donnell, and Chenggang Wu. Testing surface area. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1204–1214, 2014. 1.2

[KPS13] Satyen Kale, Yuval Peres, and Comandur Seshadhri. Noise tolerance of expanders and sublinear expansion reconstruction. *SIAM Journal on Computing*, 42(1):305–323, 2013. 1.2

[KR00] Michael Kearns and Dana Ron. Testing problems with sublearning sample complexity. *Journal of Computer and System Sciences*, 61(3):428–456, 2000. 1.1, 1.2, 1.2

[KV18] Weihao Kong and Gregory Valiant. Estimating learnability in the sublinear data regime. In *Proceedings of the 31st Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 5460–5469, 2018. 1.2

[Mid04] Gatis Midrijānis. Exact quantum query complexity for total boolean functions. *arXiv preprint*, quant-ph/0403168, 2004. 1

[Nee14] Joe Neeman. Testing surface area with arbitrary accuracy. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 393–397, 2014. 1.2

[O'D14] Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014. 2.1, 2.1, 3

[OS07] Ryan O'Donnell and Rocco Servedio. Learning monotone decision trees in polynomial time. *SIAM Journal on Computing*, 37(3):827–844, 2007. 2.1

[OSSS05] Ryan O'Donnell, Michael Saks, Oded Schramm, and Rocco Servedio. Every decision tree has an influential variable. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 31–39, 2005. 1.2, 2, 2.1, 3

[PR02] Michal Parnas and Dana Ron. Testing the diameter of graphs. *Random Struct. Algorithms*, 20(2):165–183, 2002. 1.2

[PRR06] Michal Parnas, Dana Ron, and Ronitt Rubinfeld. Tolerant property testing and distance approximation. *Journal of Computer and System Sciences*, 72(6):1012–1042, 2006. 1.2, 1, 7

[RS96] Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996. 1

[RT11] Dana Ron and Gilad Tsur. On approximating the number of relevant variables in a function. In *Proceedings of the 15th International Workshop on Randomization and Computation (RANDOM)*, pages 676–687, 2011. 1.2

[RTVX11] Ronitt Rubinfeld, Gil Tamir, Shai Vardi, and Ning Xie. Fast local computation algorithms. In *Proceedings of the 2nd Symposium on Innovations in Computer Science (ICS)*, pages 223–238, 2011. 1.2

[Ser10] Rocco A. Servedio. Testing by implicit learning: A brief survey. In *Property Testing - Current Research and Surveys*, pages 197–210. Springer, 2010. 1.2

[SS10] Michael Saks and Comandur Seshadhri. Local monotonicity reconstruction. *SIAM Journal on Computing*, 39(7):2897–2926, 2010. 1, 1.1, 1.2, 3

[Tal13] Avishay Tal. Properties and applications of boolean function composition. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 441–454, 2013. 1