Reconstructing Decision Trees

Guy Blanc
Stanford University, CA, USA

Jane Lange
MIT, Cambridge, MA, USA

Li-Yang Tan
Stanford University, CA, USA

Abstract

We give the first *reconstruction algorithm* for decision trees: given queries to a function f that is opt-close to a size-s decision tree, our algorithm provides query access to a decision tree T where:

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

This yields a *tolerant tester* that distinguishes functions that are close to size-s decision trees from those that are far from size-S decision trees. The polylogarithmic dependence on s in the efficiency of our tester is exponentially smaller than that of existing testers.

Since decision tree complexity is well known to be related to numerous other boolean function properties, our results also provide a new algorithm for reconstructing and testing these properties.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Property reconstruction, property testing, tolerant testing, decision trees

Digital Object Identifier 10.4230/LIPIcs.ICALP.2022.24

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2012.08735`

Funding *Guy Blanc*: Supported by NSF CAREER Award 1942123.
Jane Lange: Supported in part by NSF Award CCF-2006664, Big George Fellowship, Akamai Presidential Fellowship and Google.
Li-Yang Tan: Supported by NSF CAREER Award 1942123.

Acknowledgements We would like to thank the anonymous reviews of ICALP 2022 for their helpful feedback.

1 Introduction

We study the problem of *reconstructing* decision trees: given queries to a function f that is close to a size-s decision tree, provide fast query access to a decision tree, ideally one of size not much larger than s, that is close to f. This can be viewed as an "on the fly" variant of the problem of properly and agnostically learning decision trees, where the goal there is to output the entire decision tree hypothesis. More broadly, reconstruction algorithms, introduced by Ailon, Chazelle, Comandur, and Liu [2], can be viewed as sublinear algorithms that restore structure – in our case, that of a decision tree – in a function that has been lost due to noise.

Decision trees have long been a popular and effective model in machine learning, and relatedly, they are among the most intensively studied concept classes in learning theory. The literature on learning decision trees is vast, spanning three decades and studying the problem in a variety of models and from a variety of perspectives [24, 41, 9, 26, 14, 8, 27, 36, 29, 39, 34, 31, 28, 22, 13, 7]. In contrast, the problem of reconstructing decision trees has thus far been surprisingly understudied.

1.1 Our contributions

We give the first reconstruction algorithm for decision trees. Our algorithm achieves a *polylogarithmic* dependence on s in its query and time complexities, exponentially smaller than the information-theoretic minimum required to learn.

▶ **Theorem 1** (Main result). *There is a randomized algorithm which, given queries to $f : \{\pm 1\}^n \to \{\pm 1\}$ and parameters $s \in \mathbb{N}$ and $\varepsilon \in (0, 1)$, provides query access to a fixed 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.*

Notably, in the standard setting where $s = \mathrm{poly}(n)$, the query and time complexities of our algorithm are $\mathrm{polylog}(n)$ and $\tilde{O}(n)$ respectively. Previously, the only known approach was to simply properly and agnostically learn f; the current fastest such algorithm has query and time complexities $n^{O(\log \log n)}$ [7].

Our reconstruction algorithm is furthermore *local* in the sense of Saks and Seshadhri [43], allowing queries to be answered in parallel assuming a shared random string. In particular, once f, s, ε and the random string are fixed, all queries are answered consistently with a single decision tree.

1.1.1 Implications of Theorem 1 and further results

By a standard reduction, Theorem 1 gives a *tolerant tester* for decision trees:

▶ **Corollary 2** (Tolerant testing of decision trees). *There is a randomized algorithm which, given queries 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.*

This adds to a long line of work on testing decision trees [33, 23, 19, 5, 15]. We give an overview of prior testers in Section 1.2, mentioning for now that they all have (at least) an exponentially larger dependence on s in their query and time complexities.

1.1.1.1 A new connection between tolerant testing and learning

It would be preferable if our tester can be improved to reject all f's that are far from size-s decision trees – or more strongly, if our reconstructor can be improved to provide query access to a size-s decision tree.

We show that such a tester, even one that is considerably less efficient than ours, would yield the first polynomial-time algorithm for properly learning decision trees:

▶ **Theorem 3** (Tolerant testing $\Longrightarrow$ 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.*

This would represent a breakthrough on a central open problem in learning theory. Recent work of Blanc, Lange, Qiao, and Tan [7] gives a $\mathrm{poly}(n) \cdot s^{O(\log \log s)}$ time algorithm, improving on the prior state of the art of $n^{O(\log s)}$ [24]. Neither [24]'s nor [7]'s algorithm goes through testing.

It is well known and easy to see that proper learning algorithms yield comparably efficient testers [25]. Theorem 3 provides an example of a converse; we find the existence of such a converse surprising, and are not aware of any previous examples.

1.1.1.2 Reconstructors and testers for other properties

Decision tree complexity is quantitatively related to numerous other complexity measures of boolean functions: Fourier degree, approximate degree, randomized and quantum query complexities, certificate complexity, block sensitivity, sensitivity, etc. Our results therefore immediately yield new reconstructors and tolerant testers for these properties. For example, we have the following:

▶ **Corollary 4** (Reconstruction of low Fourier degree functions)**.** *There is a randomized algorithm which, given queries 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 in turn yields a tolerant tester for Fourier degree. As in the case for decision trees, all prior testers for low Fourier degree [23, 19, 20, 5, 12, 15] have an exponential dependence on d in their query and time complexities.

Table 1 lists examples of measures for which we obtain new reconstruction algorithms, each of which in turn give new tolerant testers.

1.2 Background and comparison with prior work

As already mentioned, Theorem 1 gives the first reconstruction algorithm for decision trees. The problem of testing decision trees, on the other hand, has been intensively studied.

Testing decision trees. Recent work of Bshouty [15] gives an algorithm, running in $\mathrm{poly}(s^s, 1/\varepsilon) \cdot n$ time and using $O((s \log s)/\varepsilon)$ queries, that distinguishes between size-s decision trees from functions that are ε-far from size-s decision trees. Prior to [15], Chakraborty, García-Soriano, and Matsliah [19] gave an $O((s \log s)/\varepsilon^2)$-query algorithm, and before that Diakonikolas, Lee, Matulef, Onak, Rubinfeld, Servedio, and Wan [23] gave an $\tilde{O}(s^4/\varepsilon^2)$-query algorithm. Like [15]'s algorithm, the algorithms of [19, 23] also run in $\mathrm{poly}(s^s, 1/\varepsilon) \cdot n$ time.[1]

[1] All these testers enjoy a weak form of tolerance: they are in fact able to distinguish between functions that are $O(\mathrm{poly}(\varepsilon/s))$-close to size-$s$ decision trees from those that are ε-far from size-s decision trees. (Briefly, this is because their queries, while correlated, are each uniformly distributed.)

Table 1 Performance guarantees of our reconstruction algorithms for various complexity measures. In each row, opt_d denotes the distance from f to the closest function h such that the complexity measure of that row for h is bounded by d. 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.

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	$\mathrm{s}(h) \le d$	$\mathrm{s}(g) \le O(d^{13}/\varepsilon^2)$

Compared to these algorithms, our algorithm in Corollary 2 solves an incomparable problem with efficiency parameters that compare rather favorably with theirs. Notably, our time and query complexities both depend *polylogarithmically* on s instead of exponentially and super-linearly respectively.

Turning to the parameterized setting, Kearns and Ron [33] 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. The parameters of this result are such that one should think of the dimension "n" as being a constant rather than an asymptotic parameter.

Property reconstruction

Property reconstruction was introduced by Ailon, Chazelle, Comandur, and Liu [2]. (See also the work of Austin and Tao [3], who termed such algorithms "repair algorithms".) Reconstruction has since been studied for a number of properties, including monotone functions [2, 43, 4], hypergraph properties [3], convexity [21], expanders [32], Lipschitz functions [30], graph connectivity and diameter [17], and error correcting codes [18]. Property reconstruction falls within the *local computation algorithms* framework of Rubinfeld, Tamir, Vardi, and Xie [42].

The paper of Blanc, Gupta, Lange, and Tan [6] designs a decision tree learning algorithm that is amenable to *learnability estimation* [35, 10]: 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. Their techniques can be used to 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 [6], their algorithm is fails for non-monotone functions.

1.2.1 The work of Bhsouty and Haddad-Zaknoon

Subsequent to the posting of our work to the ArXiv, Bshouty and Haddad-Zaknoon [16] have given a tester that is closely related, but incomparable, to Corollary 2. Their tester:

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

Comparing [16]'s tester to ours, their query complexity is independent of n (whereas ours has a $\log n$ dependence), and the size of decision trees in their reject condition is only $(s/\varepsilon)^{O(\log(s/\varepsilon))}$ (whereas we require $s^{O((\log s)^2/\varepsilon^3)}$).

On the other hand, our tester is tolerant and has query complexity that achieves a polylogarithmic instead of polynomial dependence on s. Furthermore, [16] does not give a reconstruction algorithm, while that is the main contribution of our work.

1.3 Future directions

We list a few concrete avenues for future work suggested by our results:

- *Tighter connections between testing and learning:* Our tester rejects functions that are $\Omega(\varepsilon)$-far from $\mathrm{quasipoly}(s)$ decision trees, and Theorem 3 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 3 could be strengthened: Do *non-tolerant* testers for decision trees yield proper learning algorithms? Do tolerant testers yield proper learning algorithms with *agnostic* guarantees?
- *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 3 could be viewed as suggesting 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 $\tilde{O}(s)$ [15, 19] whereas the current best lower bound is $\Omega(\log s)$ [23, 5]. It would be interesting to explore whether our techniques could be useful in closing this exponential gap.

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 $f, g : \{\pm1\}^n \to \{\pm1\}$, 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 $f : \{\pm1\}^n \to \{\pm1\}$, 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 $\boldsymbol{x} \sim \{\pm1\}^n$ reaches v is $2^{-|v|}$.

2 Proofs of Theorem 1 and Theorem 2

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 $x \in \{\pm1\}^n$, the value of $T^\diamond(x)$ can be computed with only $\log S \cdot \log n$ queries to f.

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 2 as a simple consequence of Theorem 1 in Section 2.3.

2.1 Structural component of Theorem 1

▶ **Definition 5** (Noise sensitivity). *The noise sensitivity of $f : \{\pm1\}^n \to \{\pm1\}$ at noise rate p is the quantity*

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

where $\boldsymbol{x} \sim \{\pm1\}^n$ 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 6** (Score of a variable). *Given a function $f : \{\pm1\}^n \to \{\pm1\}$, noise rate $p \in (0, 1)$, and coordinate $i \in [n]$, the score of x_i is defined as*

$$\mathrm{Score}_i(f, p) = \mathrm{NS}_p(f) - \mathop{\mathbb{E}}_{\boldsymbol{b} \in \{\pm1\}}\left[\mathrm{NS}_p(f_{x_i = \boldsymbol{b}})\right].$$

(Our notion of score is equivalent, up to scaling factors depending on p, to the notion of "noisy influence" as in defined in O'Donnell's monograph [38]. 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 7.** *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 8** (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 $\mathrm{dist}(f, T_f^{d,p}) \le O(\mathrm{opt}_s) + \varepsilon$.*

While Lemma 8 covers the main essence of our structural result, we'll need a slightly more robust version.

▶ **Lemma 9** (Robust version of Lemma 8). *Let $f : \{\pm 1\}^n \to \{\pm 1\}$ 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 $\mathrm{dist}(f, T) \le O(\mathrm{opt}_s + \varepsilon)$.

The proofs of Lemmas 8 and 9 are in the full version of this paper.

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 10** (Score estimator). *There is an algorithm which, given query access to a function $f : \{\pm 1\}^n \to \{\pm 1\}$, 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

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

We prove Theorem 10 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 10 takes the mean of multiple runs of that unbiased estimator, with its guarantees following from a simple concentration bound.

▶ **Lemma 11** (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*

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

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

```
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

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

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

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

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

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,

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

We now prove Equation (1):

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

Lemma 11 then holds by linearity of expectation. ◀

We now prove Theorem 10.

Proof of Theorem 10. 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 10, 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 Theorem 1. We build off a simple observation that also underlies [6]: 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 [6]'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 $T^\circ_{\text{internal}}(\alpha) \in [n] \cup \{\varnothing\}$ 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, $T^\circ_{\text{leaf}}(\alpha) \in \{-1, 1, \varnothing\}$ 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 12** (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 $z^{(1)}, \dots, z^{(m)} \in \{\pm1\}^n$, the outputs of RECONSTRUCTOR are consistent with some decision tree T where*

- *T has size $s^{O((\log s)^2/\varepsilon^3)}$,*
- *$\mathrm{dist}(T, f) \le O(\mathrm{opt}_s) + \varepsilon$ 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.

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 9 and therefore $\mathrm{dist}(T, f) \le O(\mathrm{opt}_s) + \varepsilon$. 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 13** (Efficiency of RECONSTRUCTOR). *For any $f : \{\pm1\}^n \to \{\pm1\}$, $s \in \mathbb{N}$, $\varepsilon \in (0, \frac{1}{2})$, $\delta \in (0,1)$, particular input $z \in \{\pm1\}^n$, 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.

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

Input: Query access to a function $f : \{\pm 1\}^n \to \{\pm 1\}$, size parameter $s$, error parameter $\varepsilon$, and failure probability $\delta$.
Output: Query access to a decision tree $T$ that satisfies $\mathrm{dist}(f, T) \le O(\mathrm{opt}_s) + \varepsilon$ with probability at least $1 - \delta$.

1. Set parameters $d, p$, and $\tau$ as in Lemma 9.
2. Initialize $T^\circ$ to be the empty partial tree.
3. Upon receiving an input $z \in \{\pm 1\}^n$:
   a. Initialize $\alpha$ to be the root of $T^\circ$.
   b. Repeat $d$ times.
      i. If $T^\circ_{\text{internal}}(\alpha)$ is $\varnothing$ use the estimator from Theorem 10 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 $\overline{z}_i$ is 1, set $\alpha$ to its right child. Otherwise, set $\alpha$ 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.

Proof. On each input, the estimator from Theorem 10 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 14 (Local reconstruction). We remark that our reconstruction algorithm can be made local in the sense of [43]. 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 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 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 12, 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 2

In this section we derive Corollary 2 as a simple consequence of Theorem 1. The connection between reconstruction and tolerant testing has been noted in other works (see e.g. [17, 11]); we provide a proof here for completeness.

▶ **Theorem 15** (Corollary 2 restated). *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 $\text{poly}(\log s, 1/\varepsilon) \cdot n \log n \cdot \log(1/\delta)$ time, makes $\text{poly}(\log s, 1/\varepsilon) \cdot \log n \cdot \log(1/\delta)$ 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, $\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)} \sim \{\pm 1\}^n$ where $m = O(\log(1/\delta)/\varepsilon^2)$. Let $\boldsymbol{b}^{(1)}, \dots, \boldsymbol{b}^{(m)} \in \{\pm 1\}$ 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. $\text{opt}_s \leq \varepsilon$). By Lemma 12, with probability at least $1 - \delta$ the outputs of $\textsc{Reconstructor}$ are consistent with a tree, T, satisfying $\text{dist}(T, f) \leq O(\varepsilon)$. By Hoeffding's inequality,

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

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 12 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 13 and our choice of $m = O(\log(1/\delta)/\varepsilon^2)$. ◀

3 Proof of Corollary 4

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

▶ **Theorem 16** (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 $O(d^3/\varepsilon^2)$,*
- *$\mathrm{dist}(T, f) \le O(\mathrm{opt}_d) + \varepsilon$ 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 16, we use the fact that every depth-d decision tree has size $\le 2^d$, 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 17** (Decision tree depth vs. Fourier degree [37, 44])**.** *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 $\deg(g) \le D(g)$ and $D(g) \le \deg(g)^3$.*

We first observe Theorem 16 and Fact 17 already gives a quantitatively weaker version of Corollary 4 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 17, $D(h) \le \deg(h)^3$, and so the algorithm of Theorem 16 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 17 again, we conclude that $\deg(T) \le D(T) \le O(\deg(h)^9/\varepsilon^2)$.

To obtain the sharper bound of $O(\deg(h)^7/\varepsilon^2)$, we observe that the proof of Lemma 8 in fact bounds the depth of T by $O(D(h)^2\,\mathrm{Inf}(h)/\varepsilon^2)$. Influence and degree of boolean functions are related via the following basic fact (see e.g. [38, Theorem 37]):

▶ **Fact 18.** *For all $h : \{\pm 1\}^n \to \{\pm 1\}$, we have $\mathrm{Inf}(h) \le \deg(h)$.*

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

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 [1].

4 Proof of Theorem 3

In this section, we prove the following theorem:

▶ **Theorem 19** (Tolerant testing of DTs ⇒ 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 $s \le s'$ and $\varepsilon \ge \mathrm{poly}(1/s', \varepsilon')$, 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 3 follows as a special case of Theorem 19.

We prove Theorem 19 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 [40] 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 20** (Tolerant testing ⇒ distance estimation [40]). *Let c and $\mathcal{A}$ be as in Theorem 19. 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 $\varepsilon = \frac{\gamma}{c}, \frac{2\gamma}{c}, \frac{3\gamma}{c}, \dots, 1$, 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 19).

▶ **Lemma 21** (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 BUILDDT(f, s, d, γ) outputs a decision tree T satisfying*

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

```
BuildDT$(f, s, d, \gamma)$:

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 $\mathrm{sign}(\mathbb{E}[f])$.
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 20 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 $f : \{\pm 1\}^n \to \{\pm 1\}$.

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, $\mathrm{dist}(T, f) \leq \mathrm{opt}_s(f)$, satisfying Equation (2). If $d = 0$ and $s \geq 2$, then $\frac{s}{2^{d+2}} \geq \frac{2}{4} = \frac{1}{2}$. Furthermore, in Step 1, BuildDT always outputs a tree with error at most $\frac{1}{2}$. Therefore,

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

Finally, we consider the case where $d \geq 1$ and $s \geq 2$. 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. $\mathrm{error}(i_{\mathrm{opt}}, s_{0,\mathrm{opt}}, s_{1,\mathrm{opt}}) \leq \mathrm{opt}_s(f)$), and since $i^\star, s_0^\star, s_1^\star$ are chosen in Step 3 to minimize the estimated error, we have

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

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

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

$$= c^d \cdot \mathrm{error}(i^\star, s_0^\star, s_1^\star) + \gamma \cdot \frac{c^{d-1}(c-1) + c^{d-1} - 1}{c-1} + \frac{s}{2^{d+2}}$$
$$\le c^d \cdot \mathrm{opt}_s(f) + \gamma \cdot \left(\frac{c^d - 1}{c-1}\right) + \frac{s}{2^{d+2}}.$$

The desired result holds by induction. ◀

For readability, Lemma 21 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 19:

Proof of Theorem 19. 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 BuildDT(g, s', d, γ), 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 21, for T the tree BuildDT outputs,

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

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 20, those calls to $\mathcal{E}$ each make c/ε calls to $\mathcal{A}$. Hence, the total number of calls to $\mathcal{A}$ is

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

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 $\varepsilon = \frac{\varepsilon'}{2} \cdot \left(\frac{\varepsilon'}{s'}\right)^{\max(1, \log c)} \ge \mathrm{poly}(1/s', \varepsilon')$. ◀

References

1 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. `arXiv:2010.12629`.

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

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

4 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.

5 Eric Blais, Joshua Brody, and Kevin Matulef. Property testing lower bounds via communication complexity. *computational complexity*, 21(2):311–358, 2012.

6 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.

7 Guy Blanc, Jane Lange, Mingda Qiao, and Li-Yang Tan. Properly learning decision trees in almost polynomial time. In *Proceedings of the 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, 2021.

8 Avirm Blum, Merrick Furst, Jeffrey Jackson, Michael Kearns, Yishay Mansour, and Steven Rudich. Weakly learning DNF and characterizing statistical query learning using Fourier analysis. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC)*, pages 253–262, 1994.

9 Avrim Blum. Rank-r decision trees are a subclass of r-decision lists. *Inform. Process. Lett.*, 42(4):183–185, 1992. doi:10.1016/0020-0190(92)90237-P.

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

11 Zvika Brakerski. Local property restoring, 2008. Manuscript.

12 Joshua Brody and Pooya Hatami. Distance-sensitive property testing lower bounds. *CoRR*, abs/1304.6685, 2013. arXiv:1304.6685.

13 Alon Brutzkus, Amit Daniely, and Eran Malach. ID3 learns juntas for smoothed product distributions. In *Proceedings of the 33rd Annual Conference on Learning Theory (COLT)*, pages 902–915, 2020.

14 Nader Bshouty. Exact learning via the monotone theory. In *Proceedings of 34th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 302–311, 1993.

15 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.

16 Nader H. Bshouty and Catherine A. Haddad-Zaknoon. On learning and testing decision tree. *ArXiv preprint*, abs/2108.04587, 2021. arXiv:2108.04587.

17 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.

18 Sourav Chakraborty, Eldar Fischer, and Arie Matsliah. Query complexity lower bounds for reconstruction of codes. *Theory of Computing*, 10(19):515–533, 2014. doi:10.4086/toc.2014.v010a019.

19 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.

20 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.

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

22 Sitan Chen and Ankur Moitra. Beyond the low-degree algorithm: mixtures of subcubes and their applications. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC)*, pages 869–880, 2019.

23 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.

24 Andrzej Ehrenfeucht and David Haussler. Learning decision trees from random examples. *Information and Computation*, 82(3):231–246, 1989.

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

26 Thomas Hancock. Learning $k\mu$ decision trees on the uniform distribution. In *Proceedings of the 6th Annual Conference on Computational Learning Theory (COT)*, pages 352–360, 1993.

27 Thomas Hancock, Tao Jiang, Ming Li, and John Tromp. Lower bounds on learning decision lists and trees. *Information and Computation*, 126(2):114–122, 1996.

28 Elad Hazan, Adam Klivans, and Yang Yuan. Hyperparameter optimization: A spectral approach. *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

29 Jeffrey C. Jackson and Rocco A. Servedio. On learning random dnf formulas under the uniform distribution. *Theory of Computing*, 2(8):147–172, 2006. doi:10.4086/toc.2006.v002a008.

30 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.

31 Adam Kalai, Alex Samorodnitsky, and Shang-Hua Teng. Learning and smoothed analysis. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 395–404, 2009.

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

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

34 Adam Klivans and Rocco Servedio. Toward attribute efficient learning of decision lists and parities. *Journal of Machine Learning Research*, 7(Apr):587–602, 2006.

35 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.

36 Dinesh Mehta and Vijay Raghavan. Decision tree approximations of boolean functions. *Theoretical Computer Science*, 270(1-2):609–623, 2002.

37 Gatis Midrijānis. Exact quantum query complexity for total boolean functions. *arXiv preprint*, quant-ph/0403168, 2004. arXiv:quant-ph/0403168.

38 Ryan O'Donnell. *Analysis of Boolean Functions.* Cambridge University Press, 2014.

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

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

41 Ronald Rivest. Learning decision lists. *Machine learning*, 2(3):229–246, 1987.

42 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.

43 Michael Saks and Comandur Seshadhri. Local monotonicity reconstruction. *SIAM Journal on Computing*, 39(7):2897–2926, 2010.

44 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.