Properly Learning Decision Trees in almost Polynomial Time

GUY BLANC, Stanford, USA
JANE LANGE, MIT, USA
MINGDA QIAO and LI-YANG TAN, Stanford, USA

We give an $n^{O(\log \log n)}$-time membership query algorithm for properly and agnostically learning decision trees under the uniform distribution over $\{\pm 1\}^n$. Even in the realizable setting, the previous fastest runtime was $n^{O(\log n)}$, a consequence of a classic algorithm of Ehrenfeucht and Haussler.

Our algorithm shares similarities with practical heuristics for learning decision trees, which we augment with additional ideas to circumvent known lower bounds against these heuristics. To analyze our algorithm, we prove a new structural result for decision trees that strengthens a theorem of O'Donnell, Saks, Schramm, and Servedio. While the OSSS theorem says that every decision tree has an influential variable, we show how every decision tree can be "pruned" so that *every* variable in the resulting tree is influential.

CCS Concepts: • **Theory of computation → Boolean function learning**; *Design and analysis of algorithms;*

Additional Key Words and Phrases: Decision trees, proper learning

ACM Reference format:
Guy Blanc, Jane Lange, Mingda Qiao, and Li-Yang Tan. 2022. Properly Learning Decision Trees in almost Polynomial Time. *J. ACM* 69, 6, Article 39 (November 2022), 19 pages.
https://doi.org/10.1145/3561047

1 INTRODUCTION

Decision trees are a simple and effective way to represent boolean functions $f : \{\pm 1\}^n \to \{\pm 1\}$. Their logical, flow-chart-like structure makes them easy to understand, and they are the canonical example of an interpretable model in machine learning. They are also fast to evaluate: the complexity of evaluating a decision tree on an input scales with the depth of the tree, which is often much smaller than the dimension n of f.

The algorithmic problem of converting a function f into a decision tree representation T has therefore been extensively studied by a number of communities spanning both theory and practice. Naturally, we would like T to be as small as possible, ideally close to the optimal decision tree size of f. If we require T to compute f *exactly*, this is unfortunately likely an intractable problem, even if T is allowed to be larger than the optimal decision tree for f: finding an approximately minimal decision tree for a given function is NP-hard [1, 15, 28, 30].

Guy and Li-Yang are supported by NSF CAREER Award 1942123. Mingda is supported by DOE Award DE-SC0019205 and ONR Young Investigator Award N00014-18-1-2295. Jane is supported by NSF Award CCF-2006664.
Authors' addresses: G. Blanc, M. Qiao, and L.-Y. Tan, Gates Computer Science building, 353 Serra Mall, Stanford, CA 94305; J. Lange, Stata Center, 32 Vassar St, Cambridge, MA 02139.

0004-5411/2022/11-ART39 $15.00
https://doi.org/10.1145/3561047

We therefore allow T to err on a small fraction of inputs. Our main result is a new algorithm for this problem:

Theorem 1. *There is an algorithm which, given as input $\varepsilon > 0$, $s \in \mathbb{N}$, and query access to a function $f : \{\pm 1\}^n \to \{\pm 1\}$ that is promised to be opt_s-close to a size-s decision tree, runs in time*

$$\tilde{O}(n^2) \cdot (s/\varepsilon)^{O(\log((\log s)/\varepsilon))}$$

and outputs a size-s decision tree T that w.h.p. satisfies $\Pr_{\text{uniform } \boldsymbol{x}}[T(\boldsymbol{x}) \neq f(\boldsymbol{x})] \leq \mathrm{opt}_s + \varepsilon$.

For $s = \mathrm{poly}(n)$ and $\varepsilon \geq 1/\mathrm{polylog}(n)$, our algorithm runs in almost polynomial time, $n^{O(\log \log n)}$. Even in the realizable setting ($\mathrm{opt}_s = 0$), the previous fastest algorithms took quasipolynomial time, $n^{\Omega(\log n)}$, even for constant ε. This was the state of the art even for algorithms with access to an explicit representation of f, rather than just query access.

Another interesting setting is when the algorithm is only given uniform random examples labeled by f rather than query access. For this setting, we have the following result:

Theorem 2. *In the context of Theorem 1, if f is* monotone*, our algorithm uses only random labeled examples $(\boldsymbol{x}, f(\boldsymbol{x}))$ where $\boldsymbol{x} \sim \{\pm 1\}^n$ is uniformly random.*

1.1 Background and Context

In the language of learning theory, Theorem 1 gives a query algorithm for properly and agnostically learning decision trees under the uniform distribution. We now overview previous algorithms for this and related problems.

Ehrenfeucht and Haussler [10], in an early paper following the introduction of the PAC learning model, gave an $n^{O(\log s)}$ time algorithm for properly learning size-s decision trees. [10]'s algorithm works in the more general distribution-free setting and only uses random examples. On the other hand, [10] assumes the realizable setting, and their algorithm is not known to extend to the agnostic setting. This limitation is likely inherent: being an Occam algorithm, its analysis crucially relies on noiseless examples. Furthermore, [10]'s algorithm is *weakly* proper, in the sense that its decision tree hypothesis can be as large as $n^{\Omega(\log s)}$. A (strongly) proper algorithm returns a hypothesis that belongs to the target concept class; in this case, a size-s decision tree hypothesis for a size-s decision tree target.

Since the work of Ehrenfeucht and Haussler, a couple of alternative algorithms for properly learning decision trees have been developed in the uniform-distribution setting. These algorithms are quite different from [10]'s and from each other. Mehta and Raghavan [22] gave an $n^{O(\log s)}$ time algorithm that uses random examples, and more recently [3] gave a $\mathrm{poly}(n) \cdot s^{O(\log s)}$ time membership query algorithm. For the standard setting where $s = \mathrm{poly}(n)$, these runtimes are still $n^{\Omega(\log n)}$, just like [10]'s.

Therefore, while [10]'s $n^{O(\log n)}$ runtime for properly learning polynomial-size decision trees has been matched twice in the uniform-distribution setting, it has remained unsurpassed for over three decades. Furthermore, the analyses of all three algorithms are known to be tight: for each of them, there are targets for which the algorithm can be shown to require $n^{\tilde{\Omega}(\log n)}$ time.

Table 1 summarizes how our algorithm compares with existing ones.

Improper algorithms. While the focus of our work is on proper learning, the problem of improperly learning decision trees, where the hypothesis is not required to itself be a decision tree, is also the subject of intensive study. Kusilevitz and Mansour [20] gave a polynomial-time membership query algorithm for learning polynomial-size decision trees under the uniform distribution; this was subsequently extended to the agnostic setting by Gopalan, Kalai, and Klivans [11]. Both works

Table 1. Algorithms for Properly Learning Size-s Decision Trees

Reference	Running time	Hypothesis size	Access to target	Agnostic?
[10]	$n^{O(\log s)}$	$n^{O(\log s)}$	Random examples	×
[22]	$n^{O(\log s)}$	s	Random examples	✓
[3]	$\mathrm{poly}(n) \cdot s^{O(\log s)}$	$s^{O(\log s)}$	Queries	×
This work	$\mathrm{poly}(n) \cdot s^{O(\log\log s)}$	s	Queries	✓

[10]'s algorithm works in the more general distribution-free setting, whereas all others, including ours, work in the uniform-distribution setting.

employ Fourier-analytic techniques, and their algorithms return the sign of a Fourier polynomial as their hypothesis.

Other works on improper learning of decision trees include [4, 5, 7, 8, 12–14, 16, 18, 19, 26, 27].

On the use of membership queries. It would be preferable if our algorithm in Theorem 1 did not require membership queries and instead relied only on random examples. However, there are well-known barriers to obtaining such an improvement of our algorithm, even an improper one and even just within the realizable setting.

First, no such statistical query algorithm exists: any SQ algorithm for learning polynomial-size decision trees has to take $n^{\Omega(\log n)}$ time [5]. Second, we observe that our $\mathrm{poly}(n) \cdot s^{O(\log\log s)}$ runtime is fixed-parameter tractable in 's'. Obtaining a $\mathrm{poly}(n) \cdot \Phi(s)$ time algorithm that only uses random examples, for any growth function Φ, would give the first polynomial-time algorithm for learning $\omega_n(1)$-juntas. This would be a breakthrough on a notorious open problem [6]; current algorithms for learning k-juntas take time $n^{\Omega(k)}$ [23, 29].

2 OVERVIEW OF OUR APPROACH

The starting point of our work is [3]'s $\mathrm{poly}(n) \cdot s^{O(\log s)}$ time algorithm for the realizable setting. We begin with a brief overview of their algorithm, followed by a description of how we obtain our improved $\mathrm{poly}(n) \cdot s^{O(\log\log s)}$ time algorithm in the realizable setting. We then explain how we extend our algorithm to the agnostic setting.

[3]'s greedy algorithm. At the heart of [3]'s algorithm, as well as ours, is the notion of the *influence* of a variable on a function. For a function $f : \{\pm 1\}^n \to \{\pm 1\}$ and a variable $i \in [n]$, the influence of i on f is the quantity $\Pr[f(\boldsymbol{x}) \neq f(\boldsymbol{x}^{\sim i})]$, where $\boldsymbol{x} \sim \{\pm 1\}^n$ is uniformly random and $\boldsymbol{x}^{\sim i}$ denotes $\boldsymbol{x}$ with its i-th coordinate rerandomized.

[3] analyzes a simple greedy algorithm for constructing a decision tree T for f:

(1) Using membership queries to f, identify the variable $i \in [n]$ with (approximately) the largest influence on f.
(2) Query x_i at the root of T.
(3) Build the left and right subtrees of T by recursing on $f_{x_i=-1}$ and $f_{x_i=1}$ respectively.

[3] proved that growing this tree to size $s^{O(\log s)}$ yields a high-accuracy hypothesis for f. Their algorithm, like [10]'s, is weakly proper.

A near-matching lower bound. [3] provided a near-matching lower bound showing that their analysis of their algorithm is essentially tight. They exhibited a size-s decision tree target f such

that the tree grown by their algorithm has to reach size $s^{\tilde{\Omega}(\log s)}$ before achieving any nontrivial accuracy.

2.1 Our Algorithm and Its Analysis

[3]'s algorithm formalizes the intuition, drawn from decision tree learning heuristics used in practice (e.g., ID3, CART, C4.5), that the most influential variable is a "somewhat good" root: the greedy strategy of recursively querying the most influential variable converges to a high-accuracy hypothesis at size $s^{O(\log s)}$. Their lower bound establishes the limitations of this strategy.

At a high level, we obtain our improved algorithm by showing that *there's an even better root among the* polylog(s) *most influential variables.* Rather than committing to the single most influential variable as the root of our tree, we consider the set of polylog(s) most influential variables as candidate roots. We prove the existence of a variable x_i within this set such that growing a size-s tree with x_i as the root results in a high-accuracy hypothesis for f.

2.1.1 Our Key New Tool: A Pruning Lemma for Decision Trees. The analysis of our algorithm is driven by a new structural lemma for decision trees. This lemma generalizes a result of O'Donnell, Saks, Schramm, Servedio [25]—the *OSSS inequality*—which is the crux of [3]'s analysis of their algorithm:

Theorem 3 (OSSS Inequality). *Let $f : \{\pm 1\}^n \to \{\pm 1\}$ be a size-s decision tree. Then:*

$$\max_{i \in [n]} \{\mathrm{Inf}_i(f)\} \geq \frac{\mathrm{Var}(f)}{2 \log s},$$

where $\mathrm{Var}(f)$ *denotes the variance of the random variable $f(\boldsymbol{x})$.*

In words, the OSSS inequality says that every small-size decision tree (that is not too biased) has an influential variable.

Our new structural lemma shows that every decision tree can be "pruned" so that *every* variable in the resulting tree is influential. Our notion of pruning is simple and is based on a single atomic procedure: one prunes a decision tree T by iteratively replacing any of its internal nodes by one of the node's subtrees.

Theorem 4 (Our Pruning Lemma for the Realizable Setting). *Let f be computable by a size-s decision tree T and $\tau > 0$. There is a pruning $T^\star$ of T satisfying:*

- $\Pr_{\text{uniform } \boldsymbol{x}}[f(\boldsymbol{x}) \neq T^\star(\boldsymbol{x})] \leq \tau \log s$;
- *For every node v of $T^\star$, writing $i(v)$ to denote the variable queried at v, we have that*

$$\mathrm{Inf}_{i(v)}(f_v) \geq \tau, \tag{1}$$

  where f_v denotes the restriction of f by the root-to-v path in $T^\star$.

(We show in the body of this paper that this pruning lemma implies the OSSS inequality.)

For the realizable setting, this lemma is useful because only a small number of variables can satisfy Equation (1). It is well known and easy to show that the *total* influence of a size-s decision tree, the sum of individual variable influences, is upper bounded by $\log s$. There can therefore be at most $(\log s)/\tau$ many variables with influence at least τ. Our $\mathrm{poly}(n) \cdot s^{O(\log \log s)}$ time algorithm for the realizable setting follows quite easily from Theorem 4.

The agnostic setting. In the agnostic setting, there is no longer a good bound on the number of variables of f with influence at least τ. If f is merely *close to* a size-s decision tree, say 0.1-close, the size of this set can be as large as $\Omega(n)$ as opposed to $(\log s)/\tau$ as in the realizable setting.

To overcome this, we consider the *smoothing* of f and the *noisy influence* of its variables, and rely on a generalization of our pruning lemma based on these notions. By choosing an appropriate smoothing/noise parameter δ, we show that:

- The smoothing $\tilde{f}$ is $(\delta \log s)$-close to f;
- There are at most $1/(\tau\delta)$ many variables with noisy influence at least τ on $\tilde{f}$.

A straightforward application of these ideas yields an agnostic algorithm that achieves accuracy $O(\mathrm{opt}) + \varepsilon$. A more careful analysis further improves the guarantee to $\mathrm{opt} + \varepsilon$.

The high-level idea of using smoothing and noisy influence to upgrade a non-agnostic algorithm into an agnostic one already appears in prior work on decision tree learning [2], though the details of our analyses differ.

3 PRELIMINARIES

We use **boldface** (e.g., $\boldsymbol{x} \sim \{\pm 1\}^n$) to denote random variables, and unless otherwise stated, all probabilities and expectations are with respect to the uniform distribution. A restriction π of a function $f : \{\pm 1\}^n \to \mathbb{R}$, denoted f_π, is the subfunction of f that one obtains by fixing a subset of the variables to constants (i.e., $x_i = b$ for $i \in [n]$ and $b \in \{\pm 1\}$). We write $|\pi|$ to denote the number of variables fixed by π.

The *size* of a tree is its number of leaves, its *depth* is the length of the longest root-to-leaf path, and we define its *average depth* to be the quantity:

$$\Delta(T) \coloneqq \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \{\pm 1\}^n}[\text{depth of leaf that } \boldsymbol{x} \text{ reaches}] = \sum_{\text{leaves } \ell \in T} 2^{-|\ell|} \cdot |\ell|,$$

where $|\ell|$ denotes the depth of ℓ within T. Note that if T is a size-s decision tree, then $\Delta(T) \leq \log s$.

Definition 1 (Influence of Variables). For $f : \{\pm 1\}^n \to \{\pm 1\}$ and $i \in [n]$, the *influence of x_i with respect to f* is the quantity

$$\mathrm{Inf}_i(f) \coloneqq \Pr_{\boldsymbol{x} \sim \{\pm 1\}^n}\left[f(\boldsymbol{x}) \neq f(\boldsymbol{x}^{\sim i})\right], \tag{2}$$

where $\boldsymbol{x}^{\sim i}$ denotes $\boldsymbol{x}$ with its i^{th} coordinate rerandomized (i.e., flipped with probability $\frac{1}{2}$). More generally, for $f : \{\pm 1\}^n \to Y$ where Y is a metric space equipped with a distance function ρ,

$$\mathrm{Inf}_i(f) \coloneqq \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \{\pm 1\}^n}\left[\rho(f(\boldsymbol{x}), f(\boldsymbol{x}^{\sim i}))\right]. \tag{3}$$

Remark 1 (Metric Spaces of Interest). Although the focus of our work is on learning boolean-valued functions $f : \{\pm 1\}^n \to \{\pm 1\}$, our approach involves reasoning more generally about real-valued functions $\tilde{f} : \{\pm 1\}^n \to \mathbb{R}$. Several intermediate results that we establish for real-valued functions hold even more generally for any metric space Y as the codomain (e.g., our pruning lemma), and in those cases we state and prove them in their most general form.

Throughout this paper the codomain $Y = \{\pm 1\}$ is by default equipped with the not-equals metric $\rho(x, y) = \mathbb{1}[x \neq y]$ (note that in this case Equations (2) and (3) are equivalent), and the codomain $Y = \mathbb{R}$ is by default equipped with the absolute value metric $\rho(x, y) = |x - y|$.

Definition 2 (Distance between Functions). For any metric space Y equipped with a distance function ρ, we define the distance between two functions $f, g : \{\pm 1\}^n \to Y$ to be

$$\mathrm{dist}(f, g) \coloneqq \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \{\pm 1\}^n}[\rho(f(\boldsymbol{x}), g(\boldsymbol{x}))].$$

We say that f is *ε-close to g* if $\mathrm{dist}(f, g) \leq \varepsilon$.

We note that we can express influence in terms of distance.

Fact 3.1. *For any metric space Y, function $f : \{\pm 1\} \to Y$, and $i \in [n]$,*

$$\mathrm{Inf}_i(f) = \mathrm{dist}(f, f_{x_i=1}) = \mathrm{dist}(f, f_{x_i=-1}).$$

Fourier analysis of boolean functions. We will need the very basics of the Fourier analysis of boolean functions; for an in-depth treatment, see [24]. Every function $f : \{\pm 1\}^n \to \mathbb{R}$ can be uniquely expressed as a multilinear polynomial via its *Fourier expansion*:

$$f(x) = \sum_{S \subseteq [n]} \hat{f}(S) \prod_{i \in S} x_i, \quad \text{where } \hat{f}(S) = \mathbb{E}\left[f(\boldsymbol{x}) \prod_{i \in S} \boldsymbol{x}_i\right].$$

Definition 3 (Smoothed Version of a Function). For a function $f : \{\pm 1\}^n \to \{\pm 1\}$ and noise rate $\delta \in [0, 1]$, the *δ-smoothed version of f* is the function $f_\delta : \{\pm 1\}^n \to [-1, 1]$ defined as

$$f_\delta(x) \coloneqq \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim_\delta x}[f(\tilde{\boldsymbol{x}})],$$

where $\tilde{\boldsymbol{x}} \sim_\delta x$ denotes drawing $\tilde{\boldsymbol{x}}$ such that each coordinate $\tilde{\boldsymbol{x}}_i$ is set to x_i with probability $1 - \delta$, and rerandomized with probability δ. Equivalently, each bit of x gets flipped in $\tilde{\boldsymbol{x}}$ with probability $\frac{\delta}{2}$ independently.

We remark that f_δ is sometimes also denoted $\mathrm{T}_{1-\delta} f$, with $\mathrm{T}_{1-\delta}$ being called the noise operator with parameter $1 - \delta$.

4 OUR PRUNING LEMMA

In this section we prove our key new structural result, the decision tree pruning lemma. The actual result that we establish, Theorem 5, generalizes the pruning lemma as stated in the introduction (Theorem 4) in two ways:

(1) It holds for functions mapping into an arbitrary metric space rather than just boolean-valued functions;
(2) The decision tree T need not compute f.

Both aspects will be needed for the application to agnostic learning.

Definition 4 (Everywhere τ-influential). For any function $f : \{\pm 1\}^n \to Y$, threshold $\tau > 0$, and decision tree $T : \{\pm 1\}^n \to Y$, we say that T is *everywhere τ-influential with respect to f* if, for every internal node v of T, writing $i(v)$ to denote the variable queried at v, we have

$$\mathrm{Inf}_{i(v)}(f_v) \geq \tau,$$

where f_v denotes the restriction of f by the root-to-v path in T.

The proof of our pruning lemma is constructive—we give an efficient algorithm (Figure 1) showing how to prune T so that the resulting tree is everywhere τ-influential with respect to f—though in our applications to learning we do not need it to be constructive.

The remainder of this subsection will be devoted to proving the following generalization of Theorem 4.

Theorem 5 (Properties of Prune). *For any metric space Y, function $f : \{\pm 1\} \to Y$, decision tree $T : \{\pm 1\} \to Y$, and threshold $\tau > 0$, let $T^\star = \textsc{Prune}(f, T, \tau)$. Then,*

(1) Size and depth do not increase: The size and depth of $T^\star$ are at most the size and depth of T.
(2) Everywhere τ-influential: $T^\star$ is everywhere τ-influential with respect to f.
(3) Small increase in distance: For $\Delta(T)$ the average depth of T,

$$\mathrm{dist}(T^\star, f) \leq \mathrm{dist}(T, f) + \Delta(T) \cdot \tau.$$

PRUNE(f, T, τ):

Input: Query access to a function $f : \{\pm 1\}^n \to Y$, a decision tree $T : \{\pm 1\}^n \to Y$, and a threshold $\tau > 0$.
Output: A decision tree that is everywhere τ-influential w.r.t. f.

(1) If T has depth 0, return T.
(2) Let x_i be the variable queried at root of T and T_{-1} and T_1 be its left and right subtree, respectively.
(3) If $\mathrm{Inf}_i(f) > \tau$, return the tree that queries x_i as its root and has PRUNE$(f_{x_i=-1}, T_{-1}, \tau)$ and PRUNE$(f_{x_i=1}, T_1, \tau)$ as its left and right subtree, respectively.
(4) If $\mathrm{Inf}_i(f) \le \tau$, return whichever of PRUNE(f, T_{-1}, τ) or PRUNE(f, T_1, τ) have less distance w.r.t. f.

Fig. 1. A procedure for pruning a tree T to ensure it is everywhere τ-influential with respect to a function f.

Theorem 4 is a special case of Theorem 5 where $Y = \{\pm 1\}$ with the not-equals metric and $f \equiv T$. (Recall also that $\Delta(T) \le \log s$.) We prove each guarantee of Theorem 5 separately.

Proof of the First Guarantee of Theorem 5. By induction on the depth of T. If T has depth 0, then the size and depth of PRUNE(f, T, τ) are the same as T. For x_i the variable queried at root of T and T_{-1}, T_1 its left and right subtrees, respectively, if $\mathrm{Inf}_i(f) > \tau$,

$$\begin{aligned}\mathrm{size}(\textsc{Prune}(f, T, \tau)) &= \mathrm{size}(\textsc{Prune}(f_{x_i=-1}, T_{-1}, \tau)) + \mathrm{size}(\textsc{Prune}(f_{x_i=1}, T_1, \tau)) \\ &\le \mathrm{size}(T_{-1}) + \mathrm{size}(T_1) \\ &= \mathrm{size}(T).\end{aligned}$$

where the second step is the inductive hypothesis. Similarly, for depth

$$\begin{aligned}\mathrm{depth}(\textsc{Prune}(f, T, \tau)) &= 1 + \max\left(\mathrm{depth}(\textsc{Prune}(f_{x_i=-1}, T_{-1}, \tau)), \mathrm{depth}(\textsc{Prune}(f_{x_i=1}, T_1, \tau))\right) \\ &\le 1 + \max\left(\mathrm{depth}(T_{-1},), \mathrm{depth}(T_1)\right) \\ &= \mathrm{depth}(T).\end{aligned}$$

Finally, if $\mathrm{Inf}_i(f) \le \tau$, then PRUNE$(f, T, \tau)$ is equal to either PRUNE(f, T_{-1}, τ) or PRUNE(f, T_1, τ). Since T_{-1} and T_1 each have size and depth less than those of T, the desired result holds by the inductive hypothesis. □

Proof of the Second Guarantee of Theorem 5. By induction on the depth of T. If T has depth 0, then it has no internal nodes, so vacuously is everywhere τ-influential. Otherwise, let x_i be the variable queried at root of T and T_{-1}, T_1 be its left and right subtrees, respectively. If $\mathrm{Inf}_i(f) \le \tau$, then PRUNE$(f, T, \tau)$ is either PRUNE(f, T_{-1}, τ) or PRUNE(f, T_1, τ), which is everywhere τ-influential w.r.t. f by the inductive hypothesis.

If we did not fall in either of the above cases, then $\mathrm{Inf}_i(f) > \tau$. Let v be some internal node of PRUNE(f, T, τ). If v is the root of PRUNE(f, T, τ), then

$$\mathrm{Inf}_{i(v)}(f_v) = \mathrm{Inf}_i(f) > \tau.$$

Otherwise, let α be the restriction corresponding to the root-to-v path. Since x_i is the root of PRUNE(f, T, τ), we have that α must fix $x_i = b$ for $b \in \{\pm 1\}$ and v is an internal node for PRUNE$(f_{x_i=b}, T_b, \tau)$. Applying the inductive hypothesis to PRUNE$(f_{x_i=b}, T_b, \tau)$, we have that $\mathrm{Inf}_{i(v)}(f_v) > \tau$. □

Before we prove the third and final guarantee of Theorem 5, we state two easy facts about the subtrees of a decision tree.

Fact 4.1 (Subtrees of a Tree). *Let $T : \{\pm 1\}^n \to Y$ be some decision tree and T_{-1}, T_1 be its left and right subtrees, respectively. Then,*

$$\tfrac{1}{2} \cdot (\Delta(T_{-1}) + \Delta(T_1)) = \Delta(T) - 1. \tag{4}$$

Furthermore, any function $f : \{\pm 1\}^n \to Y$ and x_i being the root of T,

$$\tfrac{1}{2} \cdot (\mathrm{dist}(T_{-1}, f_{x_i=-1}) + \mathrm{dist}(T_1, f_{x_i=1})) = \mathrm{dist}(T, f). \tag{5}$$

Proof of the Third Guarantee of Theroem 5. By induction on the depth of T. If T has depth 0 then the claim easily holds with equality. Otherwise, let x_i be the variable queried at root of T and T_{-1}, T_1 be its left and right subtrees, respectively. We note that the depth of T_{-1} and T_1 are strictly less than the depth of T, so we can apply our inductive hypothesis to them. We consider two cases.

Case 1: $\mathrm{Inf}_i(f) > \tau$.

$$\begin{aligned}
\mathrm{dist}(\textsc{Prune}(f, T, \tau), f) &= \frac{1}{2} \cdot \left(\sum_{b \in \{\pm 1\}} \mathrm{dist}(\textsc{Prune}(f_{x_i=b}, T_b, \tau), f_{x_i=b}) \right) && \text{(Equation (5))} \\
&\le \frac{1}{2} \cdot \left(\sum_{b \in \{\pm 1\}} \mathrm{dist}(T_b, f_{x_i=b}) + \Delta(T_b) \cdot \tau \right) && \text{(Inductive hypothesis)} \\
&= \mathrm{dist}(T, f) + (\Delta(T) - 1) \cdot \tau && \text{(Equations (4) and (5))} \\
&\le \mathrm{dist}(T, f) + \Delta(T) \cdot \tau.
\end{aligned}$$

Case 2: $\mathrm{Inf}_i(f) \le \tau$.

$$\begin{aligned}
\mathrm{dist}(\textsc{Prune}(f, T, \tau), f) &= \min_{b \in \{\pm 1\}} \{\mathrm{dist}(\textsc{Prune}(f, T_b, \tau), f)\} \\
&\le \frac{1}{2} \cdot \left(\sum_{b \in \{\pm 1\}} \mathrm{dist}(\textsc{Prune}(f, T_b, \tau), f) \right) && \text{(min} \le \text{average)} \\
&\le \frac{1}{2} \cdot \left(\sum_{b \in \{\pm 1\}} \mathrm{dist}(T_b, f) + \Delta(T_b) \cdot \tau \right) && \text{(Inductive hypothesis)} \\
&\le \frac{1}{2} \cdot \left(\sum_{b \in \{\pm 1\}} \mathrm{dist}(T_b, f_{x_i=b}) + \mathrm{dist}(f, f_{x_i=b}) + \Delta(T_b) \cdot \tau \right) \\
& && \text{(Triangle inequality)} \\
&= \frac{1}{2} \cdot \left(\sum_{b \in \{\pm 1\}} \mathrm{dist}(T_b, f_{x_i=b}) + \mathrm{Inf}_i(f) + \Delta(T_b) \cdot \tau \right) && \text{(Fact 3.1)} \\
&= \mathrm{dist}(T, f) + (\Delta(T) - 1) \cdot \tau + \mathrm{Inf}_i(f) && \text{(Equations (4) and (5))} \\
&\le \mathrm{dist}(T, f) + \Delta(T) \cdot \tau. && (\mathrm{Inf}_i(f) \le \tau)
\end{aligned}$$

This completes the proof. □

4.1 Our Pruning Lemma Implies the OSSS Inequality

Several variants of the OSSS inequality (Theorem 3) have been proved over the years [9, 17, 21, 24]. We show that our pruning lemma implies the following strengthening of the OSSS inequality:

Theorem 6 ([17]). *For any function $f : \{\pm 1\}^n \to \{\pm 1\}$ and decision tree $T : \{\pm 1\}^n \to \{\pm 1\}$,*

$$\max_{i \in [n]} \left\{ \mathrm{Inf}_i(f) \right\} \geq \frac{\mathrm{bias}(f) - \mathrm{dist}(T, f)}{\Delta(T)}$$

where the bias of f is defined as

$$\mathrm{bias}(f) = \min_{b \in \{\pm 1\}} \left\{ \Pr_{\boldsymbol{x} \sim \{\pm 1\}^n} [f(\boldsymbol{x}) \neq b] \right\}.$$

The OSSS inequality follows from Theorem 6 by taking $T = f$, and because $2 \cdot \mathrm{bias}(f) \leq \mathrm{Var}(f)$. We now show that Theorem 6 is a special case of Theorem 5:

Proof of Theorem 5 $\implies$ Theorem 6. Set $Y = \{\pm 1\}$ and $\tau = \max_{i \in [n]} \{\mathrm{Inf}_i(f)\}$. There are no variables with influence more than τ on f so the only decision trees that are everywhere τ-influential w.r.t. f are the trivial ones that make no queries. In other words, $\textsc{Prune}(T, f, \tau)$ is either the constant $+1$ function or constant -1 function. Therefore,

$$\mathrm{dist}(f, \textsc{Prune}(f, T, \tau)) \geq \mathrm{bias}(f).$$

By the third guarantee of Theorem 5,

$$\begin{aligned} \mathrm{bias}(f) \leq \mathrm{dist}(f, \textsc{Prune}(f, T, \tau)) &\leq \mathrm{dist}(T, f) + \Delta(T) \cdot \tau \\ &= \mathrm{dist}(T, f) + \Delta(T) \cdot \max_{i \in [n]} \left\{ \mathrm{Inf}_i(f) \right\}. \end{aligned}$$

Rearranging completes the proof. □

5 LEARNING IN THE REALIZABLE SETTING

We first present and analyze our algorithm in the simpler realizable setting where f is exactly a size-s decision tree (i.e., $\mathrm{opt}_s = 0$).

Theorem 7 (Special Case of Theorem 1: The Realizable Setting). *There is an algorithm which, given as input $\varepsilon > 0$, $s \in \mathbb{N}$, and query access to a size-s decision tree $f : \{\pm 1\}^n \to \{\pm 1\}$, runs in time*

$$\tilde{O}(n^2) \cdot (s/\varepsilon)^{O(\log((\log s)/\varepsilon))}$$

and outputs a size-s decision tree hypothesis T that w.h.p. satisfies $\mathrm{dist}(T, f) \leq \varepsilon$.

For clarity, we describe our algorithm, BuildDT in Figure 2, under the assumption that variable influences of f and its subfunctions (i.e., the quantities $\mathrm{Inf}_i(f_\pi)$ for all i and π) can be computed exactly in unit time. In actuality one can only obtain high-accuracy estimates of these quantities via random sampling. When we prove Theorem 7 we will show how this assumption can be removed via standard arguments.

Claim 5.1 (Correctness). *During the execution of BuildDT, for any $f : \{\pm 1\}^n \to [-1, 1]$, restriction π, and $d, s \in \mathbb{N}$, if $M[\pi, s]$ is nonempty, it contains a tree T that minimizes $\mathrm{dist}(f_\pi, T)$ among all depth-$(d - |\pi|)$, size-s, everywhere τ-influential trees.*

Proof. We proceed by induction on $d - |\pi|$. When $d = |\pi|$, BuildDT populates $M[\pi, s]$ with the singleton leaf $b \in \{\pm 1\}$ that minimizes $\mathrm{dist}(f_\pi, b)$, which is indeed $\mathrm{sign}(\mathbb{E}[f_\pi])$. For the inductive step, note that each $T_{i,k}$ satisfies

$$\mathrm{dist}(f_\pi, T_{i,k}) = \tfrac{1}{2}\Big(\mathrm{dist}(f_{\pi \cup \{x_i = -1\}}, M[\pi \cup \{x_i = -1\}, k]) + \mathrm{dist}(f_{\pi \cup \{x_i = 1\}}, M[\pi \cup \{x_i = 1\}, s - k])\Big).$$

It follows from the inductive hypothesis that $T_{i,k}$ minimizes distance among all everywhere τ-influential, depth-$(d - |\pi|)$, size-s trees with x_i as the root, and whose left and right subtrees have

BuildDT$_M(f, \pi, s, d, \tau)$:

Input: Query access to a function $f : \{\pm 1\}^n \to [-1, 1]$, restriction π, size parameter s, depth parameter d, influence parameter τ. It maintains a map $M : \{\text{restrictions}\} \times [s] \to \{\text{decision trees}\}$.

Output: A decision tree T that minimizes $\text{dist}(T, f_\pi)$ among all depth-$(d - |\pi|)$, size-s, everywhere τ-influential trees.

(1) If $\pi = \varnothing$, initialize M to the empty map.
(2) If $M[\pi, s]$ is nonempty, return $M[\pi, s]$.
(3) If $|\pi| = d$ or $s = 1$, return the singleton leaf labeled $\text{sign}(\mathbb{E}[f_\pi])$.
(4) Otherwise:
   (a) Let $S \subseteq [n]$ be the set of variables i such that $\text{Inf}_i(f_\pi) \geq \tau$.
   (b) For each $i \in S$ and $k \in [s-1]$, let $T_{i,k}$ be the tree such that

$$\text{root}(T_{i,k}) = x_i$$
$$\text{left-subtree}(T_{i,k}) = \text{BuildDT}_M(f, \pi \cup \{x_i = -1\}, k, d, \tau)$$
$$\text{right-subtree}(T_{i,k}) = \text{BuildDT}_M(f, \pi \cup \{x_i = 1\}, s-k, d, \tau)$$

   (c) Set $M[\pi, s]$ to be the tree among the $T_{i,k}$'s defined above with minimal distance to f_π.
   (d) Return $M[\pi, s]$.

Fig. 2. BuildDT uses dynamic programming to find the size-s, depth-d, everywhere τ-influential tree of minimal distance to f.

sizes k and $s - k$, respectively. Since $M[\pi, s]$ is chosen to minimize distance among all such $T_{i,k}$, its distance is minimal among all size-s, depth-$(d - |\pi|)$, everywhere τ-influential trees. □

Claim 5.2 (Runtime). *Let $d, s \in \mathbb{N}$. Let $f : \{\pm 1\}^n \to \{\pm 1\}$ be a size-s decision tree, and assume that variable influences of f and its subfunctions can be computed exactly in unit time. The algorithm BuildDT$_M(f, \varnothing, s, d, \tau)$ runs in time $n \cdot s^2 \cdot ((\log s)/\tau)^{O(d)}$.*

Proof. For all π, the size of the set S defined on Step 4(a) is at most

$$|S| \leq \frac{1}{\tau} \sum_{i=1}^{n} \text{Inf}_i(f_\pi) \leq \frac{\log s}{\tau}, \tag{6}$$

where the second inequality uses the fact that for any size-s decision tree $T : \{\pm 1\}^n \to \{\pm 1\}$,

$$\sum_{i=1}^{n} \text{Inf}_i(T) \leq \sum_{i=1}^{n} \Pr[T \text{ queries } \boldsymbol{x}_i] = \Delta(T) \leq \log s.$$

Since BuildDT terminates once $|\pi| = d$ (Step 3), and a restriction π is extended by $\{x_i = b\}$ for some $b \in \{\pm 1\}$ only if $\text{Inf}_i(f_\pi) \geq \tau$ (Step 4), the number of different restrictions that can be constructed throughout the execution of the algorithm is at most

$$\sum_{k=1}^{d} \left(\frac{\log s}{\tau}\right)^k = \left(\frac{\log s}{\tau}\right)^{O(d)}.$$

Since BuildDT returns at Step 2 if $M[\pi, s]$ is nonempty, this ensures that Step 4, the recursive part of BuildDT, is reached at most once for each restriction π and size s. The total number of recursive calls is therefore upper bounded by

$$s \cdot \left(\frac{\log s}{\tau}\right)^{O(d)}. \tag{7}$$

Outside of the recursive calls, the runtime of BuildDT is

$$O(n + s \cdot |S|) \leq O(n) + s \cdot \left(\frac{\log s}{\tau}\right). \tag{8}$$

The factor of n comes from computing and comparing influences of variables (Line 4(a)), and the factor of $s \cdot |S|$ comes from Line 4(b), the number of different (candidate root, size split) pairs. The overall runtime is therefore at most the product of the bounds in Equation (7) and (8), and the proof is complete. □

Proof of Theorem 7. Let $d := \log(s/\varepsilon)$ and $\tau := \varepsilon/\log s$. We first establish correctness: we claim that $\text{BuildDT}_M(f, \varnothing, s, d, \tau)$ returns a size-s tree T satisfying $\text{dist}(T, f) \le 2\varepsilon$. Since $\Delta(f) \le \log s$, our pruning lemma, Theorem 5, tells us that there is a pruning $T^\star$ of f that is everywhere τ-influential and satisfies $\text{dist}(f, T^\star) \le \Delta(f) \cdot \tau \le \varepsilon$.

Let $T^\star_{\text{trunc}}$ be $T^\star$ truncated to depth d (where the new leaves introduced by truncated paths are labeled with arbitrary leaf values, say 1). This tree, $T^\star_{\text{trunc}}$, is a depth-d, size-s, everywhere τ-influential tree. Furthermore,

$$\begin{aligned}\text{dist}\left(T^\star, T^\star_{\text{trunc}}\right) &\le \Pr_{\boldsymbol{x}\sim\{\pm1\}^n}[\text{depth of leaf in } T^\star \text{ that } \boldsymbol{x} \text{ reaches} > d] \\ &\le 2^{-d} \cdot s \qquad \text{(Union bound over leaves of } T^\star) \\ &= \varepsilon.\end{aligned}$$

Therefore, $\text{dist}(f, T^\star_{\text{trunc}}) \le \text{dist}(f, T^\star) + \varepsilon \le 2\varepsilon$ and , by Claim 5.1 BuildDT returns a tree T that also satisfies $\text{dist}(f, T) \le 2\varepsilon$.

As for runtime, in Claim 5.2 we assumed that variable influences can be computed exactly in unit time, whereas in actuality, we can only obtain estimates of these quantities via random sampling. By inspection of our proofs, it is straightforward to verify that it suffices for these estimates to be accurate to $\pm\frac{\tau}{2}$. Query access to f provides us with query access to f_π for any π, and hence by the Chernoff bound, we can estimate $\text{Inf}_i(f_\pi)$ to accuracy $\pm\frac{\tau}{2}$ and with confidence $1-\delta$ using $O(\log(1/\delta)/\tau^2)$ queries and in $n \cdot O(\log(1/\delta)/\tau^2)$ time. As shown in Claim 5.2, the number of times variables influences are computed throughout the execution of the algorithm is at most $n \cdot ((\log s)/\tau)^{O(d)}$, and so by setting $\delta < 1/(n \cdot ((\log s)/\tau)^{O(d)})$, we ensure that w.h.p. all our estimates are indeed accurate to within $\pm\frac{\tau}{2}$. Combining this with Claim 5.2, the overall runtime of our algorithm is

$$n \cdot s^2 \cdot \left(\frac{\log s}{\tau}\right)^{O(d)} \cdot \frac{n}{\tau^2}\Big(\log n + d\log\left((\log s)/\tau\right)\Big) \le \tilde{O}(n^2) \cdot (s/\varepsilon)^{O(\log((\log s)/\varepsilon))},$$

and this completes the proof. □

6 LEARNING MONOTONE TARGET FUNCTIONS IN THE AGNOSTIC SETTING

In the remainder of this paper we extend our analysis from the realizable to the agnostic setting. As alluded to in the introduction, the main challenge that arises when if f is merely *close to* a small-size decision tree, instead of being *exactly* a small-size decision tree, is that we no longer have a good bound on the number of its variables with influence at least τ. In the realizable setting we were able to bound this number by $(\log s)/\tau$ (Equation (6) in the proof of Theorem 7) but this crucially relied on the assumption that f and and its subfunctions are size-s decision trees, and hence have total influence at most $\log s$.

The way we handle this in the case of general target functions is somewhat involved; we give the full analysis in the next section. In this section we consider the special case of *monotone* target functions and prove Theorem 2. For monotone functions f, we show that we can easily bound the number of variables of influence at least τ by $1/\tau^2$, even if f is not a small-size decision tree. Furthermore, we also show that for monotone targets f our algorithm does not need membership queries to f and can instead rely only on uniform random labeled examples.

We recall two basic facts from the Fourier analysis of boolean functions:

Fact 6.1 (Parseval's Identity). *For all boolean functions $f : \{\pm 1\}^n \to \{\pm 1\}$,*

$$\sum_{S \subseteq [n]} \hat{f}(S)^2 = \mathbb{E}[f(\boldsymbol{x})^2] = 1.$$

Fact 6.2 (Influence = linear Fourier Coefficient for Monotone f). *For all monotone boolean functions $f : \{\pm 1\}^n \to \{\pm 1\}$ and all $i \in [n]$,*

$$\mathrm{Inf}_i(f) = \tfrac{1}{2}\,\mathbb{E}[f(\boldsymbol{x})\boldsymbol{x}_i] = \tfrac{1}{2}\hat{f}(\{i\}).$$

Combining these facts, we also have the following, which is needed for our runtime bound.

Corollary 6.3. *For all monotone boolean functions $f : \{\pm 1\}^n \to \{\pm 1\}$ and all $\tau \in [0, 1]$,*

$$|\{i \mid \mathrm{Inf}_i(f) \geq \tau\}| \leq \frac{1}{4\tau^2}.$$

Proof. The sum of squares of linear Fourier coefficients is at most the sum of squares of all Fourier coefficients, so by Parseval's identity, it is at most 1:

$$\sum_{i \in [n]} \mathrm{Inf}_i(f)^2 = \frac{1}{4}\sum_{i=1}^{n} \hat{f}(i)^2 \leq \frac{1}{4}\sum_{S \subseteq [n]} \hat{f}(S)^2 = \frac{1}{4}.$$

The corollary follows since $\mathrm{Inf}_i(f) \geq \tau$ iff $\mathrm{Inf}_i(f)^2 \geq \tau^2$. □

6.1 Proof of Theorem 2

Theorem 8 (Theorem 2 Restated). *Let $f : \{\pm 1\}^n \to \{\pm 1\}$ be a monotone boolean function that is opt_s-close to a size-s decision tree. Then for $d := \log(s/\varepsilon)$ and $\tau := \varepsilon/\log s$, the algorithm $\textsc{BuildDT}_M(f, \varnothing, s, d, \tau)$ runs in time*

$$\tilde{O}(n^2) \cdot (s/\varepsilon)^{O(\log((\log s)/\varepsilon))},$$

uses $\mathrm{poly}(s/\varepsilon) \cdot \log n$ uniform random examples labeled by f, and outputs a size-s decision tree hypothesis T that satisfies $\mathrm{dist}(f, T) \leq \mathrm{opt}_s + \varepsilon$.

The proof is very similar to that of Theorem 7 and we point out the essential differences.

Correctness. Claim 5.1 does not use the assumption that f is exactly a size-s decision tree, so correctness essentially follows from Claim 5.1 exactly as in the proof of Theorem 7. Let T_{opt} be the size-s decision tree that is opt_s-close to f. Our pruning lemma, Theorem 5, tells us that there is a pruning $T^\star$ of T_{opt} that is everywhere τ-influential and satisfies $\mathrm{dist}(f, T^\star) \leq \mathrm{opt}_s + \varepsilon$. Then, letting $T^\star_{\mathrm{trunc}}$ be $T^\star$ truncated to depth d, we have that $T^\star_{\mathrm{trunc}}$ is a depth-d, size-s, everywhere τ-influential tree that satisfies $\mathrm{dist}(f, T^\star_{\mathrm{trunc}}) \leq \mathrm{opt}_s + O(\varepsilon)$. Therefore, by Claim 5.1 BuildDT returns a tree T that also satisfies $\mathrm{dist}(f, T) \leq \mathrm{opt}_s + O(\varepsilon)$.

Runtime. We have the following analogue of Claim 5.2:

Claim 6.4 (Runtime in the Monotone Case). *Assume that variable influences of f and its subfunctions can be computed exactly in unit time. For all $d, s \in \mathbb{N}$ and $\tau > 0$, the algorithm $\textsc{BuildDT}_M(f, \varnothing, s, d, \tau)$ runs in time $n \cdot s^2 \cdot (1/\tau)^{O(d)}$.*

Proof. By Fact 6.3, we have that for any restriction f_π, the set S defined on Step 4(a) of the algorithm has size at most $|S| \leq \frac{1}{4\tau^2}$. The rest of the proof proceeds exactly as in Claim 5.2, where $\frac{\log s}{\tau}$ is replaced by $\frac{1}{4\tau^2}$. This gives a bound of

$$s \cdot \left(\frac{1}{\tau}\right)^{O(d)}$$

for the number of recursive calls, and a bound of

$$n \cdot s^2 \cdot \left(\frac{1}{\tau}\right)^{O(d)}$$

for the total running time. □

Finally, we remove the assumption that the variable influences of f and its restrictions can be computed in unit time. We claim that they can be efficiently estimated to sufficiently high accuracy using only uniform random labeled examples $(\boldsymbol{x}, f(\boldsymbol{x}))$. As in the proof of Theorem 7, it suffices to ensure that all the estimates that our algorithm makes are accurate to within $\pm\frac{\tau}{2}$.

Using Fact 6.2, we have for any i and restriction π,

$$\mathrm{Inf}_i(f_\pi) = \tfrac{1}{2}\,\mathbb{E}[f_\pi(\boldsymbol{x})\boldsymbol{x}_i] = \tfrac{1}{2}\,\mathbb{E}[f(\boldsymbol{x})\boldsymbol{x}_i \mid \boldsymbol{x} \text{ consistent with } \pi].$$

The right hand side is equivalent to

$$\frac{\frac{1}{2}\,\mathbb{E}\left[f(\boldsymbol{x})\boldsymbol{x}_i \cdot \mathbb{1}[\boldsymbol{x} \text{ consistent with } \pi]\right]}{\Pr[\boldsymbol{x} \text{ consistent with } \pi]} = 2^{|\pi|-1} \cdot \mathbb{E}\left[f(\boldsymbol{x})\boldsymbol{x}_i \cdot \mathbb{1}[\boldsymbol{x} \text{ consistent with } \pi]\right].$$

To estimate $\mathrm{Inf}_i(f_\pi)$ to accuracy $\pm\frac{\tau}{2}$, it then suffices to estimate $\mathbb{E}[f(\boldsymbol{x})\boldsymbol{x}_i \cdot \mathbb{1}[\boldsymbol{x} \text{ consistent with } \pi]]$ to accuracy $\pm\tau \cdot 2^{-|\pi|}$. By Chernoff bounds, this can be estimated with confidence $\geq 1-\delta$ with

$$O\left(\frac{1}{\tau^2} \cdot 2^{2d} \cdot \log(1/\delta)\right)$$

uniform random examples $(\boldsymbol{x}, f(\boldsymbol{x}))$ labeled by f, where we have used the fact that $|\pi| \leq d$. Each estimate takes time

$$n \cdot O\left(\frac{1}{\tau^2} \cdot 2^{2d} \cdot \log(1/\delta)\right).$$

The number of times variable influences are computed during the execution of BuildDT is at most $n \cdot (\frac{1}{\tau})^{O(d)}$, so by setting $\delta < 1/(n \cdot (\frac{1}{\tau})^{O(d)})$ we ensure that w.h.p. all our estimates are indeed accurate to within $\pm\frac{\tau}{2}$. The sample complexity of our algorithm is

$$O\left(\frac{1}{\tau^2} \cdot 2^{2d} \cdot \left(\log n + d \log \tfrac{1}{\tau}\right)\right) = \mathrm{poly}(s/\varepsilon) \cdot \log n,$$

and by Fact 6.4, the overall runtime of our algorithm is

$$n \cdot s^2 \cdot \left(\frac{1}{\tau}\right)^{O(d)} \cdot \frac{n \cdot 2^{2d}}{\tau^2} \cdot (\log n + d \log(1/\tau)) \leq \tilde{O}(n^2) \cdot (s/\varepsilon)^{O(\log((\log s)/\varepsilon))}.$$

This completes the proof of Theorem 2.

7 LEARNING GENERAL TARGET FUNCTIONS IN THE AGNOSTIC SETTING

In this section we prove Theorem 1. The algorithm for the agnostic setting calls the same procedure BuildDT as in the realizable setting, but on the smoothed version f_δ of function f (recall Definition 3).

Correctness. We'll prove that the output of BuildDT on f_δ is close to f. For that, we'll need some facts about the noise operator.

Fact 7.1 (Noise Sensitivity of Decision Trees). *For any $\delta \in (0,1)$ and decision tree $T : \{\pm1\}^n \to \{\pm1\}$,*

$$\mathrm{dist}(T_\delta, T) \leq \Delta(T) \cdot \delta.$$

Proof. We expand the distance between T_δ and T,

$$\begin{aligned}\operatorname{dist}(T_\delta, T) &= \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\{\pm1\}^n}\left[|T_\delta(\boldsymbol{x}) - T(\boldsymbol{x})|\right] \\ &= \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\{\pm1\}^n}\left[\left|\mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}\sim_\delta \boldsymbol{x}}[T(\tilde{\boldsymbol{x}})] - T(\boldsymbol{x})\right|\right] \\ &= 2\cdot \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\{\pm1\}^n}\left[\Pr_{\tilde{\boldsymbol{x}}\sim_\delta \boldsymbol{x}}[T(\tilde{\boldsymbol{x}}) \neq T(\boldsymbol{x})\right].\end{aligned}$$

For any $\boldsymbol{x} \in \{\pm1\}^n$, let $d(\boldsymbol{x})$ be the depth of the leaf in T that $\boldsymbol{x}$ reaches. In order for $T(\tilde{\boldsymbol{x}}) \neq T(\boldsymbol{x})$, $T(\tilde{\boldsymbol{x}})$ must reach a different leaf in T than $\boldsymbol{x}$ does. For that to happen, one of the $d(\boldsymbol{x})$ coordinates T queries for $\boldsymbol{x}$ must flip. By union bound, this occurs with probability at most $\frac{\delta\cdot d(\boldsymbol{x})}{2}$. Therefore,

$$\operatorname{dist}(T_\delta, T) \leq 2\cdot \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\{\pm1\}^n}\left[\frac{\delta\cdot d(\boldsymbol{x})}{2}\right] = \Delta(T)\cdot\delta. \qquad \square$$

Fact 7.2 (Noise Operator is Self-adjoint, Also in [24]). *For any functions $f, g : \{\pm1\}^n \to \{\pm1\}$,*

$$\operatorname{dist}(f_\delta, g) = \operatorname{dist}(f, g_\delta).$$

Proof. Drawing $\boldsymbol{x} \sim \{\pm1\}^n$ uniformly and then $\tilde{\boldsymbol{x}} \sim_\delta \boldsymbol{x}$ gives the same joint distribution over $(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ as first drawing $\tilde{\boldsymbol{x}} \sim_\delta \{\pm1\}^n$ uniformly and then $\boldsymbol{x} \sim_\delta \tilde{\boldsymbol{x}}$. That fact is used between the third and fourth line of the following series of algebraic manipulations.

$$\begin{aligned}\operatorname{dist}(f_\delta, g) &= \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\{\pm1\}^n}\left[\left|\mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}\sim_\delta \boldsymbol{x}}[f_\delta(\tilde{\boldsymbol{x}})] - g(\boldsymbol{x})\right|\right] \\ &= 2\cdot \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\{\pm1\}^n}\left[\Pr_{\tilde{\boldsymbol{x}}\sim_\delta \boldsymbol{x}}[f(\tilde{\boldsymbol{x}}) \neq g(\boldsymbol{x})]\right] \\ &= 2\cdot \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\{\pm1\}^n, \tilde{\boldsymbol{x}}\sim_\delta \boldsymbol{x}}\left[\mathbb{1}[f(\tilde{\boldsymbol{x}}) \neq g(\boldsymbol{x})]\right] \\ &= 2\cdot \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}\sim\{\pm1\}^n, \boldsymbol{x}\sim_\delta \tilde{\boldsymbol{x}}}\left[\mathbb{1}[f(\tilde{\boldsymbol{x}}) \neq g(\boldsymbol{x})]\right] \\ &= 2\cdot \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}\sim\{\pm1\}^n}\left[\Pr_{\boldsymbol{x}\sim_\delta \tilde{\boldsymbol{x}}}[f(\tilde{\boldsymbol{x}}) \neq g(\boldsymbol{x})]\right] \\ &= \operatorname{dist}(f, g_\delta). \qquad \square\end{aligned}$$

Given the above two facts, we are able to prove that our algorithm has the desired error on guarantee.

Lemma 7.3. *For any size s and $\varepsilon \in (0,1)$, set $d := \log(\frac{s}{\varepsilon})$ and $\tau := \frac{\varepsilon}{\log s}$. Then, for any $\delta \leq \frac{\varepsilon}{\log s}$, BuildDT$_M(f_\delta, \varnothing, s, d, \tau)$ returns a decision tree T satisfying*

$$\operatorname{dist}(T, f) \leq \operatorname{opt}_s + 4\varepsilon.$$

Proof. Let $T^\star$ be the size-s decision tree that f is opt_s-close to. First, we show $T^\star$ is also close to f_δ.

$$\begin{aligned}\operatorname{dist}(T^\star, f_\delta) &= \operatorname{dist}((T^\star)_\delta, f) && \text{(Fact 7.2)} \\ &\leq \operatorname{dist}((T^\star)_\delta, T^\star) + \operatorname{dist}(T^\star, f) && \text{(Triangle inequality)} \\ &\leq \delta\cdot\Delta(T^\star) + \operatorname{opt}_s && \text{(Fact 7.1)} \\ &\leq \varepsilon + \operatorname{opt}_s. && (\Delta(T^\star) \leq \log(s), \delta \leq \tfrac{\varepsilon}{\log s})\end{aligned}$$

By Theorem 5 (applied with the metric space $Y = [-1, 1]$), we know that there is some $T^\star_{\text{pruned}}$ that is everywhere τ-influential with respect to f_δ satisfying,

$$\text{dist}\left(T^\star_{\text{pruned}}, f_\delta\right) \le \left(\text{opt}_s + \varepsilon\right) + \varepsilon = \text{opt}_s + 2\varepsilon.$$

As in the proof of Theorem 7, let $T^\star_{\text{trunc}}$ be $T^\star_{\text{pruned}}$ truncated to depth d (where the new leaves introduced by truncated paths are labeled with arbitrary leaf values, say 1). This tree $T^\star_{\text{trunc}}$ is a depth-d, size-s, everywhere τ-influential tree that satisfies

$$\text{dist}\left(f_\delta, T^\star_{\text{trunc}}\right) \le \text{dist}\left(f_\delta, T^\star_{\text{pruned}}\right) + \varepsilon \le \text{opt}_s + 3\varepsilon.$$

Therefore, by Claim 5.1 BuildDT returns a tree T that also satisfies $\text{dist}(f_\delta, T) \le \text{opt}_s + 3\varepsilon$. Finally, we bound the distance between f and T.

$$\begin{aligned}
\text{dist}(f, T) &\le \text{dist}(f, T_\delta) + \text{dist}(T, T_\delta) && \text{(Triangle inequality)}\\
&\le \text{dist}(f_\delta, T) + \text{dist}(T, T_\delta) && \text{(Fact 7.2)}\\
&\le (\text{opt}_s + 3\varepsilon) + \delta \cdot \Delta(T) && \text{(Fact 7.1)}\\
&\le (\text{opt}_s + 3\varepsilon) + \varepsilon = \text{opt}_s + 4\varepsilon. && \left(\Delta(T) \le \log s, \delta \le \tfrac{\varepsilon}{\log s}\right)
\end{aligned}$$

□

Efficiency. Now we analyze the runtime of the procedure BuildDT on the smoothed function f_δ. As in the proof of Claim 5.2 for the realizable setting, we need to upper bound the number of different recursive calls to the procedure. The key step is to control the size of S, the set of variables that is sufficient influential (w.r.t. function $(f_\delta)_\pi$ and threshold τ).

We start with a well-known fact stating that the total influence of any δ-smoothed function is at most $O(1/\delta)$. Here, we use a slightly different version of influence that is defined as the expected squared difference between the functions values at $\boldsymbol{x}$ and $\boldsymbol{x}^{\sim i}$. This squared influence does not fit into Definition 1 since the squared difference is not a metric, but the advantage is that it can be easily expressed in terms of the Fourier coefficients of the function.

Fact 7.4 (Total Influence of Smoothed Functions). *For any $f : \{\pm 1\}^n \to \{\pm 1\}$ and $\delta \in (0, 1]$,*

$$\sum_{i=1}^{n} \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \{\pm 1\}^n} \left[(f_\delta(\boldsymbol{x}) - f_\delta(\boldsymbol{x}^{\sim i}))^2\right] \le \frac{1}{e\delta}.$$

Proof. Suppose that the Fourier expansion of f is $f(x) = \sum_{S \subseteq [n]} \widehat{f}(S) \prod_{i \in S} x_i$. The Fourier coefficients of f_δ are given by $\widehat{f_\delta}(S) = (1-\delta)^{|S|}\widehat{f}(S)$. Then, using the Fourier formula for the total squared influence,

$$\begin{aligned}
\sum_{i=1}^{n} \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \{\pm 1\}^n} \left[(f_\delta(\boldsymbol{x}) - f_\delta(\boldsymbol{x}^{\sim i}))^2\right] &= 2 \sum_{S \subseteq [n]} |S| \cdot \left[\widehat{f_\delta}(S)\right]^2\\
&= \sum_{S \subseteq [n]} 2|S| \cdot (1-\delta)^{2|S|} \cdot \left[\widehat{f}(S)\right]^2\\
&\le \frac{1}{e\delta} \sum_{S \subseteq [n]} \left[\widehat{f}(S)\right]^2\\
&= \frac{1}{e\delta}.
\end{aligned}$$

The third step applies $\max_{x \ge 0} x(1-\delta)^x \le \max_{x \ge 0} xe^{-\delta x} = 1/(e\delta)$, and the last step applies Parseval's identity (Fact 6.1). □

For any restriction π, applying Fact 7.4 to f_π allows us to control the number of variables that have large influences w.r.t. $(f_\pi)_\delta$. To upper bound the runtime of BuildDT, however, we need a similar guarantee for the function $(f_\delta)_\pi$, which is different from $(f_\pi)_\delta$ in general. Fortunately, the following fact states that for small δ, the two functions are pointwise close, and thus allows us to relate the influences of each variable x_i w.r.t. the two functions.

Fact 7.5. *For any $f : \{\pm1\}^n \to \{\pm1\}$ and restriction π, it holds for every $x \in \{\pm1\}^n$ that*

$$|(f_\pi)_\delta(x) - (f_\delta)_\pi(x)| \le \delta|\pi|.$$

Furthermore, for every $i \in [n]$,

$$|\mathrm{Inf}_i((f_\pi)_\delta) - \mathrm{Inf}_i((f_\delta)_\pi)| \le 2\delta|\pi|.$$

Proof. Fix $x \in \{\pm1\}^n$ and consider the following procedure for calculating $(f_\pi)_\delta(x)$:

(1) Set $\boldsymbol{y} \leftarrow x$ and draw $s_1, s_2, \ldots, s_n$ independently from Berolulli $(\delta/2)$.
(2) For each $i \in [n]$, negate $\boldsymbol{y}_i$ if $s_i = 1$.
(3) For each constraint "$x_i = b$" in π, set the i-th bit of $\boldsymbol{y}$ to b.

We can easily verify that $(f_\pi)_\delta(x) = \mathbb{E}[f(\boldsymbol{y})]$, where the expectation is over the randomness in s.

Furthermore, $(f_\delta)_\pi(x)$ can be defined by an almost identical procedure, with Steps 2 and 3 performed in reverse order: We start with $\boldsymbol{z} = x$ and draw $s \in \{\pm1\}^n$ randomly. We set $\boldsymbol{z}_i$ to b for each constraint "$x_i = b$" in π, and then negate $\boldsymbol{z}$ according to the non-zero entries in s. Similarly, we have $(f_\delta)_\pi(x) = \mathbb{E}[f(\boldsymbol{z})]$.

We can couple the two procedures by sharing the random bits s_1 through s_n. Note that if $s_i = 0$ holds for every index i that appears in π, we would end up with $\boldsymbol{y} = \boldsymbol{z}$. In other words, $\boldsymbol{y}$ and $\boldsymbol{z}$ may differ only when $s_i = 1$ for some index i that appears in π, which, by a union bound, happens with probability $\le |\pi| \cdot (\delta/2)$. Since f has codomain $\{\pm1\}$, we have

$$|(f_\pi)_\delta(x) - (f_\delta)_\pi(x)| = |\mathbb{E}[f(\boldsymbol{y})] - \mathbb{E}[f(\boldsymbol{z})]| \le \mathbb{E}\left[|f(\boldsymbol{y}) - f(\boldsymbol{z})|\right] \le 2\Pr[\boldsymbol{y} \ne \boldsymbol{z}] \le \delta|\pi|,$$

where the probability and expectations are over the coupling of $(\boldsymbol{y}, \boldsymbol{z})$ defined earlier.

The second part of the fact follows immediately: the first part implies

$$|(f_\pi)_\delta(x) - (f_\pi)_\delta(y)| - |(f_\delta)_\pi(x) - (f_\delta)_\pi(y)| \in [-2\delta|\pi|, 2\delta|\pi|]$$

for every $x, y \in \{\pm1\}^n$. Therefore, the difference between the influences,

$$\mathrm{Inf}_i((f_\pi)_\delta) - \mathrm{Inf}_i((f_\delta)_\pi) = \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \{\pm1\}^n}\left[|(f_\pi)_\delta(\boldsymbol{x}) - (f_\pi)_\delta(\boldsymbol{x}^{\sim i})| - |(f_\delta)_\pi(\boldsymbol{x}) - (f_\delta)_\pi(\boldsymbol{x}^{\sim i})|\right],$$

is also in $[-2\delta|\pi|, 2\delta|\pi|]$. □

Claim 7.6 (Runtime). *For all $d, s \in \mathbb{N}$ and $\tau, \delta > 0$ that satisfy $\tau > 2\delta d$, assuming that variable influences of f_δ and its subfunctions can be computed exactly in unit time, the algorithm $\textsc{BuildDT}_M(f_\delta, \varnothing, s, d, \tau)$ runs in time*

$$n \cdot \mathrm{poly}(s) \cdot \left(\frac{1}{e\delta(\tau - 2\delta d)^2}\right)^{O(d)}.$$

In particular, for $\delta = \tau/(4d)$, the runtime is $n \cdot \mathrm{poly}(s) \cdot (d/\tau)^{O(d)}$.

Proof. As in the proof of Claim 5.2, it suffices to show that when invoking $\textsc{BuildDT}_M(f_\delta, \varnothing, s, d, \tau)$, at most $s \cdot (\frac{1}{e\delta(\tau-2\delta d)^2})^{O(d)}$ different parameter tuples are passed to the recursive calls. It is, in turn, sufficient to prove that $|S| \le \frac{1}{e\delta(\tau-2\delta d)^2}$ holds for every recursive call

BuildDT$_M(f_\delta, \pi, s', d, \tau)$, where S is the set of indices i that satisfy $\mathrm{Inf}_i((f_\delta)_\pi) \geq \tau$. We note that

$$\begin{aligned}
i \in S &\iff \mathrm{Inf}_i((f_\delta)_\pi) \geq \tau \\
&\implies \mathrm{Inf}_i((f_\pi)_\delta) \geq \tau - 2\delta d && \text{(Fact 7.5 and } |\pi| \leq d) \\
&\iff \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\{\pm1\}^n}\left[\left|(f_\pi)_\delta(\boldsymbol{x}) - (f_\pi)_\delta(\boldsymbol{x}^{\sim i})\right|\right] \geq \tau - 2\delta d && \text{(definition of influence)} \\
&\implies \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\{\pm1\}^n}\left[\left((f_\pi)_\delta(\boldsymbol{x}) - (f_\pi)_\delta(\boldsymbol{x}^{\sim i})\right)^2\right] \geq (\tau - 2\delta d)^2. && \text{(Jensen's inequality and } \tau > 2\delta d)
\end{aligned}$$

Applying Fact 7.4 to f_π shows that the above can hold for at most $\frac{1/(e\delta)}{(\tau-2\delta d)^2}$ different indices i. This proves $|S| \leq \frac{1}{e\delta(\tau-2\delta d)^2}$ and finishes the proof. □

Now we put everything together to prove our main theorem.

Proof of Theorem 1. Let $d := \log(s/\varepsilon)$, $\tau := \frac{\varepsilon}{\log s}$ and $\delta := \frac{\tau}{4d}$. By Fact 7.3, BuildDT$_M$ $(f_\delta, \varnothing, s, d, \tau)$ returns a decision tree T that satisfies $\mathrm{dist}(T, f) \leq \mathrm{opt}_s + 4\varepsilon$.

For the runtime, in Claim 7.6 we again assumed that the influences of f_δ and its restrictions can be computed exactly in unit time. As in the proof of Theorem 7, estimating these influences up to an $O(\tau)$ additive error would suffice. Given query access to f, $(f_\delta)_\pi(x)$ can be estimated up to $O(\tau)$ error with probability $1 - \delta$ using $O(\log(1/\delta)/\tau^2)$ queries for any restriction π and input x. Then, by randomly sampling $O(\log(1/\delta)/\tau^2)$ copies of $\boldsymbol{x} \sim \{\pm1\}$ and estimating both $(f_\delta)_\pi(\boldsymbol{x})$ and $(f_\delta)_\pi(\boldsymbol{x}^{\oplus i})$, we can estimate $\mathrm{Inf}_i((f_\delta)_\pi)$ up to $O(\tau)$ error with probability $1 - O(\log(1/\delta)/\tau^2) \cdot \delta$.

By Claim 7.6, the number of variable influences that need to be computed is at most $n \cdot (d/\tau)^{O(d)}$. By setting $\delta < 1/(n^2 \cdot (d/\tau)^{O(d)})$, we can ensure that

$$n \cdot (d/\tau)^{O(d)} \cdot O(\log(1/\delta)/\tau^2) \cdot \delta \ll 1,$$

so that w.h.p. all the influence estimates are accurate up to $O(\tau)$ error. Note that estimating each influence takes $[O(\log(1/\delta)/\tau^2)]^2$ queries and thus runs in time

$$n \cdot [O(\log(1/\delta)/\tau^2)]^2 = \tilde{O}(n) \cdot \mathrm{poly}(d/\tau).$$

Together with Claim 7.6, this upper bounds the overall runtime of the algorithm by

$$n \cdot \mathrm{poly}(s) \cdot \left(\frac{d}{\tau}\right)^{O(d)} \cdot \tilde{O}(n) \cdot \mathrm{poly}(d/\tau) \leq \tilde{O}(n^2) \cdot (s/\varepsilon)^{O(\log((\log s)/\varepsilon))}.$$ □

8 CONCLUSION

We have given an $n^{O(\log\log n)}$-time membership query algorithm for properly learning decision trees under the uniform distribution, improving on the previous fastest runtime of $n^{O(\log n)}$. The obvious open problem is to obtain a polynomial-time algorithm, which would bring the state of the art for proper learning of decision trees into alignment with that of improper learning [11, 20].

Improved learning algorithms for decision trees often go hand in hand with an improved understanding of their structure. Ehrenfeucht and Haussler's algorithm [10] is based on the observation that one of the subtrees of the root of a size-s decision tree has size $\leq s/2$; [3] uses the OSSS inequality to show that influence is a good proxy for quality as a root; our algorithm is built on our decision tree pruning lemma, which strengthens the OSSS inequality and the connection between influence and root quality. A natural next step is to formulate and develop new structural results that will facilitate a polynomial-time algorithm.

Concluding on a speculative note, we remark that [3]'s algorithm is modeled after practical heuristics, such as ID3, CART, and C4.5, for learning decision trees. These are some of the earliest and most basic algorithms in machine learning, and they continue to be widely used to this day. Our

algorithm extends [3]'s and circumvents lower bounds that [3] had established for their algorithm. It would be interesting to explore possible practical implications of our work.

ACKNOWLEDGMENTS

We are grateful to the anonymous reviewers of FOCS 2021 and the Journal of the ACM, whose comments and suggestions have helped improve this article.

REFERENCES

[1] Micah Adler and Brent Heeringa. 2012. Approximating optimal binary decision trees. *Algorithmica* 62, 3-4 (2012), 1112–1121.

[2] Guy Blanc, Neha Gupta, Jane Lange, and Li-Yang Tan. 2020. Universal guarantees for decision tree induction via a higher-order splitting criterion. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS'20)*.

[3] Guy Blanc, Jane Lange, and Li-Yang Tan. 2020. Top-down induction of decision trees: Rigorous guarantees and inherent limitations. In *Proceedings of the 11th Innovations in Theoretical Computer Science Conference (ITCS'20)*, Vol. 151. 1–44.

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

[5] Avirm Blum, Merrick Furst, Jeffrey Jackson, Michael Kearns, Yishay Mansour, and Steven Rudich. 1994. Weakly learning DNF and characterizing statistical query learning using Fourier analysis. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC'94)*. 253–262.

[6] Avrim Blum and Pat Langley. 1997. Selection of relevant features and examples in machine learning. *Artificial Intelligence* 97, 1-2 (1997), 245–271.

[7] Nader Bshouty. 1993. Exact learning via the monotone theory. In *Proceedings of 34th Annual Symposium on Foundations of Computer Science (FOCS'93)*. 302–311.

[8] Sitan Chen and Ankur Moitra. 2019. Beyond the low-degree algorithm: Mixtures of subcubes and their applications. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC'19)*. 869–880.

[9] Hugo Duminil-Copin, Aran Raoufi, and Vincent Tassion. 2019. Sharp phase transition for the random-cluster and Potts models via decision trees. *Annals of Mathematics* 189, 1 (2019), 75–99.

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

[11] Parikshit Gopalan, Adam Kalai, and Adam Klivans. 2008. Agnostically learning decision trees. In *Proceedings of the 40th ACM Symposium on Theory of Computing (STOC'08)*. 527–536.

[12] Thomas Hancock. 1993. Learning $k\mu$ decision trees on the uniform distribution. In *Proceedings of the 6th Annual Conference on Computational Learning Theory (COT'93)*. 352–360.

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

[14] Elad Hazan, Adam Klivans, and Yang Yuan. 2018. Hyperparameter optimization: A spectral approach. *Proceedings of the 6th International Conference on Learning Representations (ICLR'18)* (2018).

[15] Laurent Hyafil and Ronald Rivest. 1976. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters* 5, 1 (1976), 15–17.

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

[17] Rahul Jain and Shengyu Zhang. 2011. The influence lower bound via query elimination. *Theory of Computing* 7, 1 (2011), 147–153.

[18] Adam Kalai, Alex Samorodnitsky, and Shang-Hua Teng. 2009. Learning and smoothed analysis. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS'09)*. 395–404.

[19] Adam Klivans and Rocco Servedio. 2006. Toward attribute efficient learning of decision lists and parities. *Journal of Machine Learning Research* 7, April (2006), 587–602.

[20] Eyal Kushilevitz and Yishay Mansour. 1993. Learning decision trees using the Fourier spectrum. *SIAM J. Comput.* 22, 6 (Dec. 1993), 1331–1348.

[21] Homin K. Lee. 2010. Decision trees and influence: An inductive proof of the OSSS inequality. *Theory of Computing* 6, 4 (2010), 81–84. https://doi.org/10.4086/toc.2010.v006a004

[22] Dinesh Mehta and Vijay Raghavan. 2002. Decision tree approximations of Boolean functions. *Theoretical Computer Science* 270, 1-2 (2002), 609–623.

[23] Elchanan Mossel, Ryan O'Donnell, and Rocco A. Servedio. 2004. Learning functions of k relevant variables. *J. Comput. System Sci.* 69, 3 (2004), 421–434.
[24] Ryan O'Donnell. 2014. *Analysis of Boolean Functions*. Cambridge University Press.
[25] Ryan O'Donnell, Michael Saks, Oded Schramm, and Rocco Servedio. 2005. Every decision tree has an influential variable. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*. 31–39.
[26] Ryan O'Donnell and Rocco Servedio. 2007. Learning monotone decision trees in polynomial time. *SIAM J. Comput.* 37, 3 (2007), 827–844.
[27] Ronald Rivest. 1987. Learning decision lists. *Machine Learning* 2, 3 (1987), 229–246.
[28] Detlef Sieling. 2008. Minimization of decision trees is hard to approximate. *J. Comput. System Sci.* 74, 3 (2008), 394–403.
[29] Gregory Valiant. 2015. Finding correlations in subquadratic time, with applications to learning parities and the closest pair problem. *Journal of the ACM (JACM'15)* 62, 2 (2015), 1–45.
[30] Hans Zantema and Hans Bodlaender. 2000. Finding small equivalent decision trees is hard. *International Journal of Foundations of Computer Science* 11, 2 (2000), 343–354.

Received 21 October 2021; accepted 9 July 2022