Superconstant Inapproximability of Decision Tree Learning

Caleb Koch CKOCH@STANFORD.EDU
Carmen Strassle STRASSLE@STANFORD.EDU
Li-Yang Tan LIYANG@CS.STANFORD.EDU
Stanford University

Editors: Shipra Agrawal and Aaron Roth

Abstract

We consider the task of properly PAC learning decision trees with queries. Recent work of Koch, Strassle, and Tan showed that the strictest version of this task, where the hypothesis tree T is required to be *optimally* small, is NP-hard. Their work leaves open the question of whether the task remains intractable if T is only required to be close to optimal, say within a factor of 2, rather than exactly optimal.

We answer this affirmatively and show that the task indeed remains NP-hard even if T is allowed to be within *any* constant factor of optimal. More generally, our result allows for a smooth tradeoff between the hardness assumption and inapproximability factor. As Koch et al.'s techniques do not appear to be amenable to such a strengthening, we first recover their result with a new and simpler proof, which we couple with a new XOR lemma for decision trees. While there is a large body of work on XOR lemmas for decision trees, our setting necessitates parameters that are extremely sharp and are not known to be attainable by existing such lemmas. Our work also carries new implications for the related problem of DECISION TREE MINIMIZATION.

Keywords: Decision trees, hardness of approximation, learning with queries

1. Introduction

Decision trees are a basic and popular way to represent data. Their simple logical structure makes them the prime example of an interpretable model. They are also the base model at the heart of powerful ensemble methods, such as XGBoost and random forests, that achieve state-of-the-art performance in numerous settings. Owing in part to their practical importance, the task of efficiently constructing decision tree representations of data has been intensively studied in the theory community for decades, in a variety of models and from both algorithmic and hardness perspectives. Indeed, on the heels of Cook and Karp's papers on the theory of NP-hardness, Hyafil and Rivest (1976) proved that a certain formulation of decision tree learning is NP-hard. Quoting their introduction, "While the proof to be given is relatively simple, the importance of this result can be measured in terms of the large amount of effort that has been put into finding efficient algorithms for constructing optimal binary decision trees." This effort has only compounded over the years, with a recent surge of interest coming from the interpretable machine learning community; the 2022 survey (Rudin et al., 2022) lists decision tree learning as the very first of the field's "10 grand challenges".

We consider the problem within the model of PAC learning with queries (Valiant, 1984; Angluin, 1988). In this model, the learner is given query access to a function f and i.i.d. draws from a distribution $\mathcal{D}$, along with the promise that f is computable by a size-s decision tree. Its task is to output a size-s' decision tree that achieves high accuracy with respect to f under $\mathcal{D}$, where

s' is as close to s as possible. Motivation for studying query learners for this problem is twofold. First, it models the task of converting an *existing* trained model f, for which one has query access to, into its decision tree representation—a common post-processing step for interpretability reasons. The second, more intrinsic, motivation comes from the fact that computational lower bounds against query learners, for *any* learning task, have generally been elusive. We are aware of only one such result outside of decision tree learning, on the NP-hardness of learning DNF formulas with queries (Feldman, 2006)—this resolved a longstanding open problem of Valiant (1984, 1985).

For decision tree learning, recent work of Koch et al. (2023a) showed that the strictest version of the problem, where $s' = s$, is NP-hard. This resolved an open problem that had been raised repeatedly over the years (Bshouty, 1993; Guijarro et al., 1999; Mehta and Raghavan, 2002; Feldman, 2016), but still left open the possibility of efficient algorithms achieving s' that is slightly larger than s. Koch et al. (2023a) listed this as an natural avenue for further research, while also pointing to challenges in extending their techniques to even rule out $s' = 2s$.

This work. We show that the problem remains NP-hard even for $s' = Cs$ where C is an arbitrarily large constant:

Theorem 1 *For every constant $C > 1$, there is a constant $\varepsilon > 0$ such that the following holds. If there is an algorithm running in time $t(n)$ that, given queries to an n-variable function f computable by a decision tree of size $s = O(n)$ and random examples $(\boldsymbol{x}, f(\boldsymbol{x}))$ drawn according to a distribution $\mathcal{D}$, outputs w.h.p. a decision tree of size Cs that is ε-close to f under $\mathcal{D}$, then* SAT *can be solved in randomized time $O(n^2) \cdot t(\mathrm{poly}(n))$.*

Consequently, assuming NP $\neq$ RP, any algorithm for the problem has to either be inefficient with respect to time (i.e. take superpolynomial time), or inefficient with respect to representation size (i.e. output a hypothesis of size much larger than actually necessary).

Theorem 1 is a special case of a more general result that allows for a smooth tradeoff between the strength of the hardness assumption on one hand and the inapproximability factor on the other hand:

Theorem 2 *Suppose for some $r \geq 1$ there is a time $t(s, 1/\varepsilon)$ algorithm which given queries to an n-variable function f computable by a decision tree of size s and random examples $(\boldsymbol{x}, f(\boldsymbol{x}))$ drawn according to a distribution $\mathcal{D}$, outputs w.h.p. a decision tree of size $2^{O(r)} \cdot s$ that is ε-close to f under $\mathcal{D}$. Then* SAT *can be solved in randomized time $\tilde{O}(rn^2) \cdot t(n^{O(r)}, 2^{O(r)})$.*

By taking r to be superconstant in Theorem 2, we obtain superconstant inapproximability ratios at the price of stronger yet still widely-accepted hardness assumptions. For example, assuming SAT cannot be solved in randomized *quasipolynomial* time, we get a near-polynomial inapproximability ratio of $2^{(\log s)^\gamma}$ for any constant $\gamma < 1$.

Our work also carries new implications for the related problem of DECISION TREE MINIMIZATION: Given a decision tree T, construct an equivalent decision tree T' of minimal size. This problem was first shown to be NP-hard by Zantema and Bodlaender (2000), and subsequently Sieling (2008) showed that it is NP-hard even to approximate. We recover Sieling (2008)'s inapproximability result, and in fact strengthen it to hold even if T' is only required to *mostly agree* with T on a given *subset of inputs* (rather than fully agree with T on all inputs as in Sieling (2008)). See Appendix G for details.

2. Background and Context

2.1. Algorithms for properly learning decision trees

In the language of learning theory, we are interested in the task of properly PAC learning decision trees. We distinguish between *strictly-proper* learning, where the size s' of the hypothesis decision tree has to exactly match the optimal size s of the target decision tree, and *weakly-proper* learning, where s' can be larger than s. Koch et al. (2023a) therefore establishes the hardness of strictly-proper learning whereas our work establishes the hardness even of weakly-proper learning even when s' is larger than s by any constant. We now overview the fastest known algorithms for both settings.

Strictly-proper learning via dynamic programming. There is a simple $2^{O(n)}$ time algorithm for strictly-properly learning decision trees: draw a dataset of size $O(s \log n)$, dictated by the VC dimension of size-s decision trees, and run a $2^{O(n)}$-time dynamic program to find a size-s decision tree that fits the dataset perfectly. (See Guijarro et al. (1999); Mehta and Raghavan (2002) for a description of this dynamic program.) There are no known improvements to this naive algorithm if one insists on strictly-proper learning, and indeed, Koch et al. (2023a)'s result strongly suggests that there are probably none.

Weakly-proper learning via Ehrenfeucht–Haussler. The setting of weakly-proper learning allows for a markedly faster algorithm: a classic algorithm of Ehrenfeucht and Haussler (1989) runs in $n^{O(\log s)}$ time and outputs a decision tree hypothesis of size $s' = n^{O(\log s)}$. Ehrenfeucht and Haussler (1989) listed as an open problem that of designing algorithms that output smaller hypotheses, i.e. ones where s' is closer to s. There has been no algorithmic progress on this problem since 1989, and prior to our work, there were also no hardness results ruling out efficient algorithms achieving say $s' = 2s$.

2.2. Lower bounds for random example learners

The problem is also well-studied in the model of PAC learning from *random examples*, where the algorithm is only given labeled examples $(\boldsymbol{x}, f(\boldsymbol{x}))$ where $\boldsymbol{x} \sim \mathcal{D}$. Lower bounds against random example learners are substantially easier to establish, and a sequence of works has given strong evidence of the optimality Ehrenfeucht and Haussler (1989)'s weakly-proper algorithm under standard complexity-theoretic assumptions.

Pitt and Valiant (1988), in an early paper on the hardness of PAC learning, showed that strictly-proper learning of decision trees from random examples is NP-hard; they attributed this result to an unpublished manuscript of Angluin. Hancock et al. (1996) then established a superconstant inapproximability factor (assuming $\mathrm{NP} \neq \mathrm{RP}$), which was subsequently improved to polynomial by Alekhnovich et al. (2009) (assuming the Exponential Time Hypothesis (ETH)). Recent work of Koch et al. (2023b) further improves the inapproximability factor to superpolynomial (assuming ETH) and quasipolynomial (assuming the inapproximability of parameterized SET COVER), the latter of which exactly matches Ehrenfeucht and Haussler (1989)'s performance guarantee.

It is reasonable to conjecture that Ehrenfeucht and Haussler (1989)'s algorithm is optimal even for query learners. If so, our work is a step forward for proving lower bounds in the more challenging setting of query learners so that these bounds might "catch up" with those in the random example setting; historically, the race has not been close—query-learner lower bounds have lagged far behind. Just as Koch et al. (2023a) can be viewed as establishing the query-learner analogue of Pitt and Valiant (1988)'s result (i.e. the hardness of strictly-proper learning), our work can be

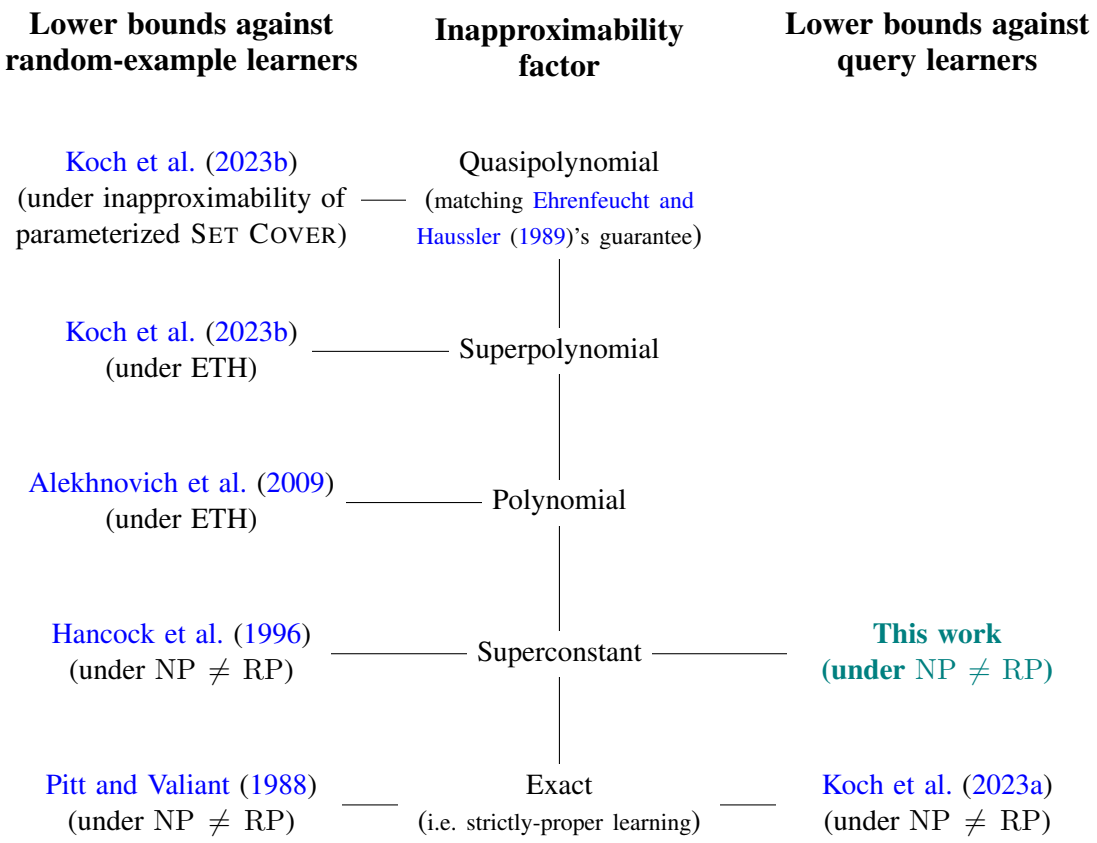

Figure 1: Summary of lower bounds for decision tree learning.

viewed as establishing the query-learner analogue of Hancock et al. (1996)'s result (i.e. superconstant inapproximability of weakly-proper learning). Figure 1 summarizes the current landscape of decision tree lower bounds and shows how our work fits into it.

We refer the reader to Section 2.1 of Koch et al. (2023a) for a discussion of the technical challenges involved in proving lower bounds against query learners, and why the techniques developed in (Pitt and Valiant, 1988; Hancock et al., 1996; Alekhnovich et al., 2009; Koch et al., 2023b) for random-example learners do not carry over.

2.3. Other related work: improper learning of decision trees

There is also vast literature on *improper* learning of decision trees, where the target function is assumed to be a small decision tree but the hypothesis does not have to be one (see e.g. (Rivest, 1987; Blum, 1992; Kushilevitz and Mansour, 1993; Hancock, 1993; Bshouty, 1993; Blum et al., 1994; Hancock et al., 1996; Jackson and Servedio, 2006; O'Donnell and Servedio, 2007; Klivans and Servedio, 2006; Gopalan et al., 2008; Kalai et al., 2009; Hazan et al., 2018; Chen and Moitra, 2019)). Examples of hypotheses that are constructed by existing algorithms include the sign of low-degree polynomials and small-depth boolean circuits.

We remark that in the machine learning literature, "decision tree learning" almost exclusively refers to the problem of constructing decision tree hypotheses. See e.g. the Wikipedia page for Decision tree learning or Chapter 18 of the textbook Shalev-Shwartz and Ben-David (2014).

3. Technical Overview

At a high level, our proof proceeds in two steps:

- **Step 1: Slight inapproximaibility.** We first give a new proof of Koch et al. (2023a)'s result. In fact, we prove a statement that is (very) slightly stronger than the hardness of strictly-proper learning: we show that it is NP-hard for query learners to construct a decision tree of size $s' = (1+\delta)\cdot s$ for small constant $\delta < 1$. While such a slight strengthening is not of much independent interest, it is important for technical reasons because it establishes *some* inapproximability factor, albeit a small one, which we then amplify in the next step.

- **Step 2: Gap amplification.** We give a reduction that for any integer r runs in time $n^{O(r)}$ and amplifies the inapproximability factor of $s'/s = 1+\delta$ from the step above into $(1+\delta)^r$. In particular, for any arbitrarily large constant C this is a reduction that runs in polynomial time and amplifies the inapproximability factor to C.

  At the heart of this reduction is a new XOR lemma for decision trees: roughly speaking, this lemma says that if decision trees of size s' incur large error when computing f, then decision trees of size $(s')^r$ incur large error when computing the r-fold XOR $f^{\oplus r}(x^{(1)},\dots,x^{(r)}) := f(x^{(1)}) \oplus \cdots \oplus f(x^{(r)})$.

There is a large body of work on XOR lemmas for decision tree complexity (Impagliazzo et al., 1994; Nisan et al., 1994; Savický, 2002; Shaltiel, 2004; Klauck et al., 2007; Jain et al., 2010; Drucker, 2012; Ben-David and Kothari, 2018; Blais and Brody, 2019; Brody et al., 2020), but our setting necessitates extremely sharp parameters that are not known to be achievable by any existing ones. Most relevant to our setting is one by Drucker (2012), but it only reasons about the error of decision trees of size $(s')^{cr}$ for some $c < 1$ instead of $(s')^r$. This constant factor loss is inherent to Drucker (2012)'s proof technique, and we explain in Remark 31 why we cannot afford even a tiny constant factor loss in the exponent.

3.1. Step 1: Slight inapproximability

Like Koch et al. (2023a), we reduce from the NP-complete problem VERTEX COVER. For every graph G there is an associated *edge indicator function* IsEdge defined as follows:

Definition 3 (IsEdge_G) *Let G be a graph with vertex set $V = \{v_1,\dots,v_n\}$. We write $\mathrm{Ind}[e] \in \{0,1\}^n$ for the encoding of an edge $e \in E$ in $\{0,1\}^n$. That is, $\mathrm{Ind}[e]_i = 1$ if and only if the vertex v_i is in e. The edge indicator function of G is the function $\mathrm{IsEdge}_G : \{0,1\}^n \to \{0,1\}$,*

$$\mathrm{IsEdge}_G(x) = \begin{cases} 1 & x = \mathrm{Ind}[e] \text{ for some } e \in E \\ 0 & \text{otherwise.} \end{cases}$$

For technical reasons, we work with a generalization of IsEdge called ℓ-IsEdge where $\ell \in \mathbb{N}$ is a tuneable "padding parameter" (we defer the definition of ℓ-IsEdge to the main body; see Definition 13). We prove that the decision tree complexity of $\ell\text{-IsEdge}_G$ scales with the vertex cover complexity of G with fairly tight quantitative parameters:

Claim 1 *Let G be a graph on n vertices and m edges. For all $\ell \geq 1$ and $\varepsilon > 0$, the following two cases hold.*

- *Yes case: if G has a vertex cover of size k, then there is a decision tree T computing $\ell\text{-IsEdge}_G$ whose size satisfies*

$$|T| \leq (\ell+1)(k+m) + mn.$$

- *No case: there is a distribution $\mathcal{D}$ such that if every vertex cover of G has size at least k', then any decision tree T that is ε-close to $\ell\text{-IsEdge}_G$ over $\mathcal{D}$ has size at least*

$$|T| \geq (\ell+1)\left(k' + (1-4\varepsilon)m\right).$$

It is known that there is a constant $\delta > 0$ such that deciding whether a graph has a vertex cover of size $\leq k$ or requires vertex cover size $\geq (1+\delta)k$ is NP-hard (Papadimitriou and Yannakakis, 1991; Håstad, 2001; Dinur and Safra, 2005). With an appropriate choice of parameters, Claim 1 translates this into a gap of $\leq s$ versus $\geq (1+\delta')s$ for some other constant $\delta' > 0$ in the decision tree complexity of ℓ-IsEdge. The NP-query hardness of learning size-s decision trees with hypotheses of size $(1+\delta')s$ follows as a corollary.

Key ingredients in the proof of Claim 1: Patch up and hard distribution lemmas. As is often the case in reductions such as Claim 1, the upper bound in the Yes case is straightforward to establish and most of the work goes into proving the lower bound in the No case. Koch et al. (2023a)'s analysis of their No case is rather specific to the ℓ-IsEdge function, whereas we develop a new technique for proving such lower bounds on decision tree complexity. In addition to being more general and potentially useful in other settings, our technique lends itself to an "XOR-ed generalization" which we will need for gap amplification. (Koch et al. (2023a)'s technique does not appear to be amenable to such a generalization, despite our best efforts at obtaining it.[1])

There are two components to our technique, both of which are generic statements concerning a decision tree T that imperfectly computes a function f. The first is a *patch up lemma* that shows how T can be patched up so that it computes f perfectly. The cost of this patch up operation, i.e. how much larger T becomes, is upper bounded by the certificate complexity of f, a basic and well-studied complexity measure of functions. (We defer the formal definitions of the technical terms and notation used in these lemmas to Section 4.)

Lemma 4 (Patch up lemma) *Let $f : \{0,1\}^n \to \{0,1\}$ be a function and let T be a decision tree. Then*

$$\mathrm{DT}(f) \leq |T| + \sum_{x \in f^{-1}(1)} \mathrm{Cert}(f_{\pi(x)}, x)$$

where $\pi(x)$ denote the path followed by x in T and $f_{\pi(x)}$ is the restriction of f by $\pi(x)$.

1. On a more technical level, Koch et al. (2023a)'s technique requires them to reason about the complexity of PARTIAL VERTEX COVER, a generalization of VERTEX COVER, whereas our simpler approach bypasses the need for this.

The second component is a *hard distribution lemma* that shows how a hard distribution $\mathcal{D}$ can be designed so that the error of T with respect to f under $\mathcal{D}$ is large. Roughly speaking, the more weight that $\mathcal{D}$ places on "highly sensitive" points, the larger the error is:

Lemma 5 (Hard distribution lemma) *Let $f : \{0,1\}^n \to \{0,1\}$ be a nonconstant function. Then for all nonempty $C \subseteq f^{-1}(1)$, there is a distribution over C and all of its sensitive neighbors such that for any decision tree T, we have*

$$\mathrm{error}_{\mathcal{D}}(T, f) \geq \frac{1}{2|C|\mathrm{Sens}(f)} \sum_{x \in S} |\mathrm{Sens}(f_{\pi(x)}, x)|$$

where $\pi(x)$ is the path followed by x in T and $f_{\pi(x)}$ is the restriction of f by $\pi(x)$.

The No case of Claim 1 follows by applying Lemmas 4 and 5 to the ℓ-IsEdge function and reasoning about its certificate complexity and sensitivity.

3.2. Step 2: Gap amplification

As alluded to above, a key advantage of our approach is that the patch up and hard distribution lemmas lend themselves to "XOR-ed generalizations":

Lemma 6 (XOR-ed version of Patch Up Lemma, see Lemma 18 for the exact version) *Let $f : \{0,1\}^n \to \{0,1\}$ be a function and let T be a decision tree. Then for all $r \geq 1$,*

$$\mathrm{DT}(f^{\oplus r}) \leq |T| + 2^r \sum_{x \in f^{-1}(1)^r} \prod_{i=1}^{r} \max\{1, \mathrm{Cert}(f_{\pi(x)}, x^{(i)})\}$$

where $\pi(x)$ is the path followed by x in T and $f_{\pi(x)}$ is the restriction of f by $\pi(x)$.

Lemma 7 (XOR-ed version of Hard Distribution Lemma, see Lemma 19 for the exact version) *Let $f : \{0,1\}^n \to \{0,1\}$ be a nonconstant function, $C \subseteq f^{-1}(1)$ be nonempty, and T be a decision tree. There is a distribution $\mathcal{D}$ over the inputs in C and their sensitive neighbors such that for all $r \geq 1$,*

$$\mathrm{error}_{\mathcal{D}^{\otimes r}}(T, f^{\oplus r}) \geq \left(\frac{1}{2|C|\mathrm{Sens}(f)}\right)^r \sum_{x \in C^r} \prod_{i=1}^{r} \max\{1, |\mathrm{Sens}(f_{\pi(x)}, x^{(i)})|\}$$

where $\pi(x)$ is the path followed by x in T and $f_{\pi(x)}$ is the restriction of f by $\pi(x)$.

Just as how Lemmas 4 and 5 combine to yield Claim 1, combining their XOR-ed generalizations Lemmas 4 and 19 yields the following amplified version of of Claim 1:

Claim 2 *Let G be a graph on n vertices and m edges. For all $\ell, r \geq 1$ and $\varepsilon > 0$, the following two cases hold.*

- *Yes case: if G has a vertex cover of size k, then there is a decision tree T computing ℓ-IsEdge$_G^{\oplus r}$ whose size satisfies*

$$|T| \leq \left[(\ell+1)(k+m) + mn\right]^r.$$

- *No case: there is a distribution $\mathcal{D}$ such that if every vertex cover of G has size at least k', then any decision tree T that is ε-close to ℓ-$\mathrm{IsEdge}_G^{\oplus r}$ over $\mathcal{D}$ has size at least*

$$|T| \geq \big[(\ell+1)(k'+m)\big]^r - \varepsilon\big[8m(\ell+1)\big]^r.$$

With an appropriate choice of parameters, Claim 2 translates a gap of $\leq k$ versus $\geq (1+\delta)k$ in the vertex cover complexity of G into a gap of $\leq s^r$ versus $\geq (1+\delta')^r s^r$ in the decision tree complexity of ℓ-$\mathrm{IsEdge}_G^{\oplus r}$, where δ' is a constant that depends only on δ. Theorem 2 follows as a corollary. See Figure 2 for an illustration of this amplification and how it fits into our overall reduction from VERTEX COVER.

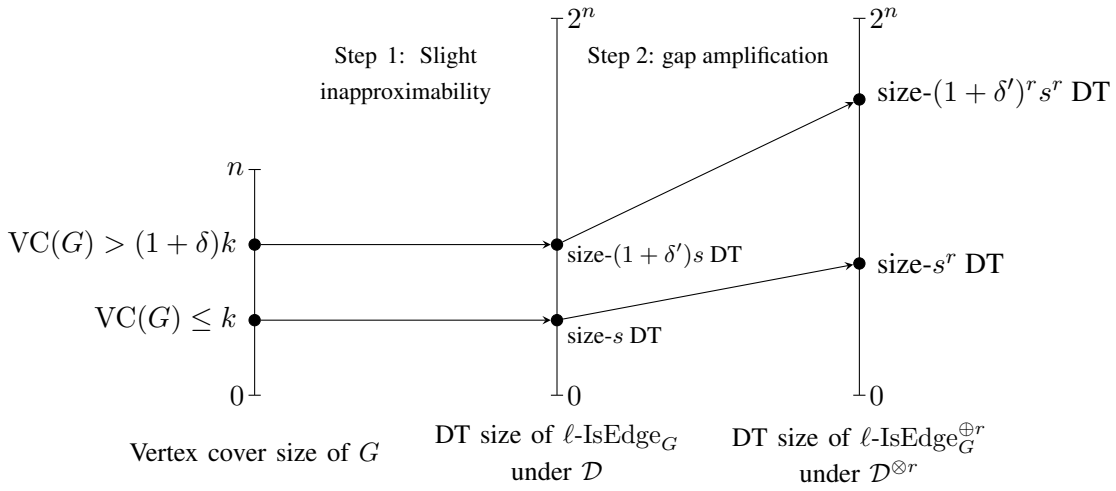

Figure 2: An illustration of main reduction from VERTEX COVER in two steps. The first step, which establishes slight inapproximability of decision tree learning, is proved in Claim 1. The second step amplifies this slight inapproximability gap using Claim 2.

4. Preliminaries

Notation and naming conventions. We write $[n]$ to denote the set $\{1, 2, \ldots, n\}$. We use lower case letters to denote bitstrings e.g. $x, y \in \{0,1\}^n$ and subscripts to denote bit indices: x_i for $i \in [n]$ is the ith index of x. The string $x^{\oplus i}$ is x with its ith bit flipped. We use superscripts to denote multiple bitstrings of the same dimension, e.g. $x^{(1)}, x^{(2)}, ..., x^{(j)} \in \{0,1\}^n$. For a set S and an integer $r \geq 1$, we write S^r to denote the r-ary Cartesian product of the set.

Distributions. We use boldface letters e.g. $\boldsymbol{x}, \boldsymbol{y}$ to denote random variables. For a distribution $\mathcal{D}$, we write $\mathrm{error}_{\mathcal{D}}(f,g) = \Pr_{\boldsymbol{x}\sim\mathcal{D}}[f(\boldsymbol{x}) \neq g(\boldsymbol{x})]$. The support of the distribution is the set of elements with nonzero mass and is denoted $\mathrm{supp}(\mathcal{D})$. For $r \geq 1$, we write $\mathcal{D}^{\otimes r}$ to denote the r-wise product distribution $\underbrace{\mathcal{D} \times \cdots \times \mathcal{D}}_{r \text{ times}}$.

Decision trees. The size of a decision tree T is its number of leaves and is denoted $|T|$. In an abuse of notation, we also write T for the function computed by the decision tree T. We say T computes

a function $f : \{0,1\}^n \to \{0,1\}$ if $T(x) = f(x)$ for all $x \in \{0,1\}^n$. The decision tree complexity of a function f is the size of the smallest decision tree computing f and is denoted $\mathrm{DT}(f)$.

Restrictions and decision tree paths. A restriction ρ is a set $\rho \subseteq \{x_1, \overline{x}_1, \ldots, x_n, \overline{x}_n\}$ of literals, and f_ρ is the subfunction obtained by restricting f according to ρ: $f_\rho(x^\star) = f(x^\star|_\rho)$ where $x^\star|_\rho$ is the string obtained from $x^\star$ by setting its ith coordinate to 1 if $x_i \in \rho$, 0 if $\overline{x}_i \in \rho$, and otherwise setting it to $x^\star_i$. We say an input $x^\star$ is consistent with ρ if $x_i \in \rho$ implies $x^\star_i = 1$ and $\overline{x}_i \in \rho$ implies $x^\star_i = 0$.

Paths in decision trees naturally correspond to restrictions. A depth-d path can be identified with a set of d literals: $\pi = \{\ell_1, \ell_2, \ldots, \ell_d\}$ where each ℓ_i corresponds to a query of an input variable and is unnegated if π follows the right branch and negated if π follows the left branch.

Boolean Functions. We use f to denote an arbitrary n-bit Boolean function, $f : \{0,1\}^n \to \{0,1\}$. For a set $D \subseteq \{0,1\}^n$, we write $f : D \to \{0,1\}$ for the partial Boolean function defined on D. We use both partial and total functions and specify the setting by writing either $f : \{0,1\}^n \to \{0,1\}$ or $f : D \to \{0,1\}$. For $f : D \to \{0,1\}$, the sensitivity of f on $x \in D$ and the certificate complexity of f's value on x are defined as

$$\begin{aligned}\mathrm{Sens}(f,x) &= \{x^{\oplus i} \in D : f(x) \neq f(x^{\oplus i}) \text{ for } i \in [n]\}\\ \mathrm{Cert}(f,x) &= |\pi| \text{ s.t. } \pi \text{ is the shortest restriction consistent with } x \text{ and}\\ &\qquad f_\pi \text{ is a constant function.}\end{aligned}$$

Note that both of these definitions are with respect to D. Also, we refer to the *sensitivity of* f which is $\mathrm{Sens}(f) \coloneqq \max_{x \in D} |\mathrm{Sens}(f,x)|$.

Graphs. An undirected graph $G = (V, E)$ has n vertices $V \subseteq [n]$ and $m = |E|$ edges $E \subseteq V \times V$. The degree of a vertex $v \in V$ is the number of edges containing it: $|\{e \in E : v \in e\}|$. The graph G is degree-d if every vertex $v \in V$ has degree at most d. We often use letters v, u, w to denote vertices of a graph G.

Learning. In the PAC learning model, there is an unknown distribution $\mathcal{D}$ and some unknown *target* function $f \in \mathcal{C}$ from a fixed *concept* class $\mathcal{C}$ of functions over a fixed domain. An algorithm for learning $\mathcal{C}$ over $\mathcal{D}$ takes as input an error parameter $\varepsilon \in (0,1)$ and has oracle access to an *example oracle* $\mathrm{EX}(f, \mathcal{D})$. The algorithm can query the example oracle to receive a pair $(\boldsymbol{x}, f(\boldsymbol{x}))$ where $\boldsymbol{x} \sim \mathcal{D}$ is drawn independently at random. The goal is to output a *hypothesis* h such that $\mathrm{dist}_{\mathcal{D}}(f,h) \leq \varepsilon$. Since the example oracle is inherently randomized, any learning algorithm is necessarily randomized. So we require the learner to succeed with some fixed probability e.g. $2/3$. A learning algorithm is *proper* if it always outputs a hypothesis $h \in \mathcal{C}$. A learning algorithm with *queries* is given oracle access to the target function f along with the example oracle $\mathrm{EX}(f, \mathcal{D})$.

4.1. Vertex Cover

Vertex cover. A vertex cover for an undirected graph $G = (V, E)$ is a subset of the vertices $C \subseteq V$ such that every edge has at least one endpoint in C. We write $\mathrm{VC}(G)$ to denote the size of the smallest vertex cover of G. The VERTEX COVER problem is to decide whether a graph contains a vertex cover of size $\leq k$. For $a > 1$, an a-approximation of VERTEX COVER corresponds to the problem of deciding whether a graph contains a vertex cover of size $\leq k$ or every vertex cover has

size at least $a \cdot k$. There is a polynomial-time greedy algorithm for vertex cover which achieves a 2-approximation.

Hardness of approximation. Constant factor hardness of VERTEX COVER is known, even for bounded degree graphs (graphs whose degree is bounded by some universal constant). This is the main hardness result we reduce from in this work.

Theorem 8 (Hardness of approximating VERTEX COVER) *There are constants $\delta > 0$ and $d \in \mathbb{N}$ such that if* VERTEX COVER *can be $(1+\delta)$-approximated on n-vertex degree-d graphs in time $t(n)$, then* SAT *can be solved in time $t(n \operatorname{polylog} n)$.*

Theorem 8 follows from the works of Papadimitriou and Yannakakis (1991); Arora and Safra (1998); Arora et al. (1998). The fact that Theorem 8 holds for constant degree graphs will be essential for our lower bound because it allows us to assume that k is large: $\mathrm{VC}(G) = \Theta(m)$.

Fact 1 (Constant degree graphs require large vertex covers) *If G is an m-edge degree-d graph, then $\mathrm{VC}(G) \geq m/d$.*

This fact follows from the observation that in a degree-d graph each vertex can cover at most d edges.

5. Patching up a decision tree: Proof of Lemma 4

We start with the following claim about building a decision tree from scratch using certificates. See Appendix A for the proof.

Claim 3 (Building a decision tree out of certificates) *Let $f : D \to \{0,1\}$ be a function with $D \subseteq \{0,1\}^n$, then*

$$\mathrm{DT}(f) \leq 1 + \sum_{x \in f^{-1}(1)} \mathrm{Cert}(f, x).$$

The next lemma shows that we can patch up a decision tree by querying certificates. This recovers Lemma 4 in the setting where $D = \{0,1\}^n$.

Lemma 9 (Patch-up with respect to 1-inputs) *Let $f : D \to \{0,1\}$ be a function with $D \subseteq \{0,1\}^n$ and let T be a decision tree, then*

$$\mathrm{DT}(f) \leq |T| + \sum_{x \in f^{-1}(1)} \mathrm{Cert}(f_{\pi(x)}, x)$$

Proof Let Π denote the set of paths in T. Then,

$$\begin{aligned}
\mathrm{DT}(f) &\leq \sum_{\pi \in \Pi} \mathrm{DT}(f_\pi) \\
&\leq \sum_{\pi \in \Pi} \left(1 + \sum_{x \in f_\pi^{-1}(1)} \mathrm{Cert}(f_\pi, x) \right) && \text{(Claim 3)} \\
&= |T| + \sum_{\pi \in \Pi} \sum_{x \in f_\pi^{-1}(1)} \mathrm{Cert}(f_\pi, x) && (|\Pi| = |T|) \\
&= |T| + \sum_{x \in f^{-1}(1)} \mathrm{Cert}(f_{\pi(x)}, x).
\end{aligned}$$

The last equality follows from the fact that the set of paths $\pi \in \Pi$ partition $f^{-1}(1)$. ■

6. Hard distribution lemma: Proof of Lemma 5

Lemma 5 is proved for the *canonical hard distribution* for a partial function $f : D \to \{0,1\}$.

Definition 10 (Canonical hard distribution) *For a function $f : D \to \{0,1\}$ with $D \subseteq \{0,1\}^n$, the canonical hard distribution, $\mathcal{D}_f$, is defined via the following experiment*

- *sample $\boldsymbol{x} \sim f^{-1}(1)$ u.a.r.*
- *with probability $1/2$, return $\boldsymbol{y} \sim \mathrm{Sens}(f, \boldsymbol{x})$ u.a.r.*
- *with probability $1/2$, return $\boldsymbol{x}$.*

When f is clear from context, we simply write $\mathcal{D}$. We use the canonical hard distribution to prove the following result which recovers Lemma 5 in the setting where $D = \{0,1\}^n$. See Appendix B for the proof.

Lemma 11 (Hard distribution lemma) *Let $f : D \to \{0,1\}$ be a nonconstant function for $D \subseteq \{0,1\}^n$. Then for all $C \subseteq f^{-1}(1)$, there exists a distribution $\mathcal{D}$ over C and and all of its sensitive neighbors such that for any decision tree T, we have*

$$\mathrm{error}_{\mathcal{D}}(T, f) \geq \frac{1}{2|C|\mathrm{Sens}(f)} \sum_{x \in C} |\mathrm{Sens}(f_{\pi(x)}, x)|.$$

7. Hardness of learning via ℓ-IsEdge

We are interested in the following learning task.

> DT-LEARN(s, s', ε): Given queries to an unknown function $f : \{0,1\}^n \to \{0,1\}$, random samples from a distribution $\mathcal{D}$ over $\{0,1\}^n$, parameters $s, s' \in \mathbb{N}$ and $\varepsilon \in (0,1)$, and the promise that f is a size-s decision tree, construct a size-s' decision tree T that is ε-close to f under $\mathcal{D}$.

The main theorem of this section is the following reduction from approximating VERTEX COVER to DT-LEARN. This theorem recovers, via a simpler proof, the main reduction in Koch et al. (2023a), while also achieving a better dependence on the VERTEX COVER approximation factor. Later, we build on this reduction to prove our main result Theorem 1.

Theorem 12 (Reduction from VERTEX COVER to DT-LEARN) *If there is a time-$t(n, 1/\varepsilon)$ algorithm solving DT-LEARN$(s, (1+\delta)s, \varepsilon)$ over n-variable functions for any $\varepsilon > 0$, $s = O(n)$, and $\delta > 0$, then VERTEX COVER can be $(1+\delta')$-approximated on degree-d, n-vertex graphs in randomized time $O(n^2 \cdot t(n^2, 1/\varepsilon))$ for any $\delta' > (\delta + 4\varepsilon)d + \delta$.*

At a high level, Theorem 12 works by taking a graph G, and defining an "amplified" version of the edge indicator function for G (recall Theorem 3). This function is called ℓ-IsEdge and is formally defined as follows (see also Koch et al. (2023a)).

Definition 13 (The ℓ-amplified IsEdge function) *Let $G = (V, E)$ be an n-vertex graph and $\ell \in \mathbb{N}$. The ℓ-amplified edge indicator function of G is the function*

$$\ell\text{-IsEdge}_G : \{0,1\}^n \times (\{0,1\}^n)^\ell \to \{0,1\}$$

defined as follows: $\ell\text{-IsEdge}_G(v^{(0)}, v^{(1)}, \ldots, v^{(\ell)}) = 1$ if and only if

1. $\text{IsEdge}_G(v^{(0)}) = 1$ *(i.e. $v^{(0)} = \text{Ind}[e]$ for some $e \in E$), and*
2. $v_i^{(1)} = \cdots = v_i^{(\ell)} = 1$ *for all $i \in [n]$ such that $v_i^{(0)} = 1$.*

We write $\ell\text{-Ind}[e] \in (\{0,1\}^n)^{\ell+1}$ for the generalized edge indicator $(\text{Ind}[e], \ldots, \text{Ind}[e])$. Note that $\ell\text{-IsEdge}(\ell\text{-Ind}[e]) = 1$.

7.1. Decision tree size upper bound for ℓ-IsEdge$_G$: First part of Claim 1

The upper bound in Claim 1 follows from (Koch et al., 2023a, Theorem 2):

Theorem 14 (Upper bound on decision tree size of ℓ-IsEdge (Koch et al., 2023a, Theorem 2)) *Let G be an n-vertex, m-edge graph with a vertex cover of size k. Then, there is a decision tree T computing $\ell\text{-IsEdge}_G : \{0,1\}^{n\ell+n} \to \{0,1\}$ whose size satisfies*

$$|T| \leq (\ell+1)(k+m) + mn$$

and T can be computed in polynomial-time given G and a size-k vertex cover of G.

The last part of Theorem 14, the constructivity of T, is implicit in the proof of (Koch et al., 2023a, Theorem 2), but is made explicit in their paper when proving Lemma 5.2 (see Section 5.3 of Koch et al. (2023a)).

7.2. Decision tree size lower bound for ℓ-IsEdge$_G$: Second part of Claim 1

The lower bound in Claim 1 is proved with respect to the canonical hard distribution for ℓ-IsEdge$_G$:

Definition 15 (Canonical hard distribution for ℓ-IsEdge) *For a graph G, we write ℓ-$\mathcal{D}_G$ to denote the canonical hard distribution of ℓ-IsEdge and ℓ-$D_G = \mathrm{supp}(\ell\text{-}\mathcal{D}_G)$ to denote the support of the canonical hard distribution. As per Definition 10, this distribution is defined via the experiment*

- *sample ℓ-Ind$[e]$ u.a.r. among all generalized edge indicators;*
- *with probability $1/2$, return $\boldsymbol{y}$ sampled u.a.r. from the set of sensitive neighbors: $\mathrm{Sens}(\ell\text{-IsEdge}, \ell\text{-Ind}[e]) = \{\ell\text{-Ind}[e]^{\oplus i} \mid i \text{ is a 1-coordinate of } \ell\text{-Ind}[e]\}$; and*
- *with probability $1/2$, return ℓ-Ind$[e]$.*

Lemma 16 (Decision tree size lower bound for computing ℓ-IsEdge) *Let T be a decision tree for ℓ-IsEdge$_G$ satisfying $\mathrm{error}_{\ell\text{-}\mathcal{D}_G}(T, \ell\text{-IsEdge}) \le \varepsilon$, then*

$$|T| \ge (\ell+1)\cdot(k' + (1-4\varepsilon)m)$$

where m is the number of edges of G and k' is the vertex cover size of G.

We obtain Lemma 16 using an application of the general size lower bound from Lemmas 9 and 11. See Appendix C.1 for the proof.

7.3. Putting things together to prove Theorem 12

The following key lemma is used in the analysis of correctness for the reduction in Theorem 12. It follows from Claim 1 and a careful choice of parameters. We defer the proof to Appendix C.2.

Lemma 17 (Main technical lemma) *For all $\delta, \delta', \varepsilon > 0$ and $d, k \ge 1$, the following holds. Given a constant degree-d graph G with m edges and parameter k, there is a choice of $\ell = \Theta(|G|)$ and a polynomial-time computable quantity $s \in \mathbb{N}$ such that so long as $\delta' > (\delta + 4\varepsilon)d + \delta$ and $dk \ge m$ we have:*

- ***Yes case**: if G has a vertex cover of size at most k, then there is a decision tree of size at most s which computes ℓ-IsEdge $: \{0,1\}^{n\ell+n} \to \{0,1\}$; and*
- ***No case**: if every vertex cover of G has size at least $(1+\delta')k$, then $(1+\delta)s < |T|$ for any decision tree T with $\mathrm{error}_{\ell\text{-}\mathcal{D}_G}(T, \ell\text{-IsEdge}) \le \varepsilon$.*

Theorem 12 follows in a straightforward way from Lemma 17. We defer the proof of it to Appendix C.3.

8. Patching up a decision tree for $f^{\oplus r}$: Proof of Lemma 6

The following lemma recovers Lemma 6 by setting $D = \{0,1\}^n$. See Appendix D for the proof.

Lemma 18 (XOR-ed version of patchup lemma, formal statement of Lemma 6) *Let T be a decision tree and $f : D \to \{0,1\}$ be a nonconstant function for $D \subseteq \{0,1\}^n$, then*

$$\mathrm{DT}(f^{\oplus r}) \le |T| + 2^r \sum_{\substack{x \in f^{-1}(1)^r \\ f^{\oplus r}_{\pi(x)} \text{ is nonconstant}}} \prod_{i=1}^{r} \max\{1, \mathrm{Cert}(f_{\pi(x)}, x^{(i)})\}.$$

9. Hard distribution lemma for $f^{\oplus r}$: Proof of Lemma 7

The following lemma recovers Lemma 7 by setting $D = \{0,1\}^n$. See Appendix E for the proof.

Lemma 19 (Hard distribution lemma for $f^{\oplus r}$) *Let T be a decision tree, $f : D \to \{0,1\}$ be a nonconstant function and let $C \subseteq f^{-1}(1)$. There is a distribution $\mathcal{D}$ over the inputs in C and their sensitive neighbors such that for all $r \geq 1$,*

$$\operatorname{error}_{\mathcal{D}^{\otimes r}}(T, f^{\oplus r}) \geq \left(\frac{1}{2|C|\operatorname{Sens}(f)}\right)^r \sum_{\substack{x \in C^r \\ \operatorname{Sens}(f^{\oplus r}_{\pi(x)}, x) \neq \varnothing}} \prod_{i=1}^{r} \max\{1, \operatorname{Sens}(f^{(i)}_{\pi(x)}, x^{(i)})\}.$$

10. Hardness of learning via ℓ-IsEdge$^{\oplus r}$

The main theorem of this section is the following reduction from approximating VERTEX COVER to DT-LEARN. This reduction enables us to prove Theorems 1 and 2.

Theorem 20 (Main reduction from vertex cover to DT-LEARN) *For all $r \geq 1$, $\varepsilon < 2^{-3r}$, and $A > 1$ the following holds. If there is a time $t(s, 1/\varepsilon)$ algorithm for solving DT-LEARN$(s, A \cdot s, \varepsilon)$ on n-variable functions with $s = O(n^r)$, then VERTEX COVER can be $(1+\delta')$-approximated on degree-d, n-vertex graphs in randomized time $O(rn^2 \cdot t(n^{2r}, 1/\varepsilon))$ for any $\delta' > (A^{1/r} - 1 + \varepsilon^{1/r} \cdot 8)d + A^{1/r} - 1$.*

The proof largely follows the steps in proving Theorem 12 where the lower bound is obtained using a combination of Lemmas 18 and 19. We defer the proof to Appendix F.

10.1. Proof of Theorem 2

By Theorem 8, there is a constant δ such that if VERTEX COVER can be approximated to within a factor of $1+\delta$ in time $t(n)$, then SAT can be solved in time $t(n \operatorname{polylog} n)$. Given an n-vertex graph with constant degree d, the reduction in Theorem 20 produces an instance of DT-LEARN in time $n^{O(r)}$. We choose $A = 2^{\Theta(r)}$ in Theorem 20 so that $\frac{\delta}{2} > (A^{1/r} - 1)d + A^{1/r} - 1$ and $\varepsilon = 2^{-\Theta(r)}$ small enough so that $\frac{\delta}{2} > \varepsilon^{1/r} \cdot 8d$. This ensures that our reduction from VERTEX COVER produces an instance of DT-LEARN$(s, 2^{\Theta(r)} \cdot s, 2^{-\Theta(r)})$ for $s = n^{O(r)}$. The algorithm guaranteed by the theorem statement solves DT-LEARN$(s, 2^{\Theta(r)} \cdot s, 2^{-\Theta(r)})$ in time $t(n^{O(r)}, 2^{O(r)})$. Therefore, by Theorem 20, VERTEX COVER can be approximated to within a factor of $1+\delta$ in time $O(rn^2) \cdot t(n^{O(r)}, 2^{O(r)})$. The proof is completed by using the reduction from SAT to approximating VERTEX COVER. ■

10.2. Proof of Theorem 1

Let $C > 1$ be the constant from the theorem statement. Let $r \geq 1$ be a large enough constant so that the $2^{O(r)}$ term in Theorem 2 is greater than C. In this case, the error parameter $\varepsilon = 2^{-\Theta(r)}$ is also a small constant that depends on C. It follows that any polynomial-time algorithm solving the problem in the theorem statement can solve SAT in time $\tilde{O}(rn^2) \cdot \operatorname{poly}(n^{O(r)}, 2^{O(r)})$ which is polynomial when r is constant. Therefore, the task is NP-hard. ■

Acknowledgments

We thank the COLT reviewers for their helpful feedback and suggestions.

The authors are supported by NSF awards 1942123, 2211237, 2224246, a Sloan Research Fellowship, and a Google Research Scholar Award. Caleb is also supported by an NDSEG Fellowship, and Carmen by a Stanford Computer Science Distinguished Fellowship and an NSF Graduate Research Fellowship.

References

Misha Alekhnovich, Mark Braverman, Vitaly Feldman, Adam Klivans, and Toniann Pitassi. The complexity of properly learning simple concept classes. *Journal of Computer & System Sciences*, 74(1):16–34, 2009. Preliminary version in FOCS 2004.

Dana Angluin. Remarks on the difficulty of finding a minimal disjunctive normal form for boolean functions. Unpublished Manuscript.

Dana Angluin. Queries and concept learning. *Machine learning*, 2:319–342, 1988.

Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, jan 1998. ISSN 0004-5411. doi: 10.1145/273865.273901. URL https://doi.org/10.1145/273865.273901.

Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, may 1998. ISSN 0004-5411. doi: 10.1145/278298.278306. URL https://doi.org/10.1145/278298.278306.

Shalev Ben-David and Robin Kothari. Randomized query complexity of sabotaged and composed functions. *Theory of Computing*, 14(5):1–27, 2018. doi: 10.4086/toc.2018.v014a005. URL https://theoryofcomputing.org/articles/v014a005.

Eric Blais and Joshua Brody. Optimal Separation and Strong Direct Sum for Randomized Query Complexity. In Amir Shpilka, editor, *34th Computational Complexity Conference (CCC 2019)*, volume 137 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 29:1–29:17, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-116-0. doi: 10.4230/LIPIcs.CCC.2019.29. URL http://drops.dagstuhl.de/opus/volltexte/2019/10851.

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.

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

Joshua Brody, Jae Tak Kim, Peem Lerdputtipongporn, and Hariharan Srinivasulu. A strong XOR lemma for randomized query complexity. *arXiv preprint arXiv:2007.05580*, 2020.

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

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.

Irit Dinur and Samuel Safra. On the hardness of approximating minimum vertex cover. *Annals of mathematics*, pages 439–485, 2005.

Andrew Drucker. Improved direct product theorems for randomized query complexity. *computational complexity*, 21(2):197–244, 2012.

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

Vitaly Feldman. Hardness of approximate two-level logic minimization and pac learning with membership queries. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC)*, pages 363–372, 2006.

Vitaly Feldman. Hardness of proper learning. In *Encyclopedia of Algorithms*, pages 897–900. Springer New York, NY, 2016. doi: 10.1007/978-1-4939-2864-4_177. URL https://doi.org/10.1007/978-1-4939-2864-4_177.

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

David Guijarro, Vıctor Lavın, and Vijay Raghavan. Exact learning when irrelevant variables abound. *Information Processing Letters*, 70(5):233–239, 1999.

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

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.

Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, jul 2001. ISSN 0004-5411. doi: 10.1145/502090.502098. URL https://doi.org/10.1145/502090.502098.

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

Laurent Hyafil and Ronald L Rivest. Constructing optimal binary decision trees is NP-complete. *Information processing letters*, 5(1):15–17, 1976.

Russell Impagliazzo, Ran Raz, and Avi Wigderson. A direct product theorem. In *Proceedings of IEEE 9th Annual Conference on Structure in Complexity Theory*, pages 88–96, 1994.

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. URL http://www.theoryofcomputing.org/articles/v002a008.

Rahul Jain, Hartmut Klauck, and Miklos Santha. Optimal direct sum results for deterministic and randomized decision tree complexity. *Information Processing Letters*, 110(20):893–897, 2010.

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.

Hartmut Klauck, Robert Špalek, and Ronald de Wolf. Quantum and classical strong direct product theorems and optimal time-space tradeoffs. *SIAM Journal on Computing*, 36(5):1472–1493, 2007. Preliminary version in FOCS 2004.

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

Caleb Koch, Carmen Strassle, and Li-Yang Tan. Properly learning decision trees with queries is NP-hard. In *Proceedings of the 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 2383–2407, 2023a.

Caleb Koch, Carmen Strassle, and Li-Yang Tan. Superpolynomial lower bounds for decision tree learning and testing. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1962–1994, 2023b.

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

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

Noam Nisan, Steven Rudich, and Michael Saks. Products and help bits in decision trees. In *Proceedings 35th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 318–329, 1994.

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

Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–440, 1991. ISSN 0022-0000. doi: https://doi.org/10.1016/0022-0000(91)90023-X. URL https://www.sciencedirect.com/science/article/pii/002200009190023X.

Leonard Pitt and Leslie G Valiant. Computational limitations on learning from examples. *Journal of the ACM (JACM)*, 35(4):965–984, 1988.

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

Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistics Surveys*, 16:1 – 85, 2022. doi: 10.1214/21-SS133. URL https://doi.org/10.1214/21-SS133.

Petr Savickỳ. On determinism versus unambiquous nondeterminism for decision trees. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 9, 2002.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Ronen Shaltiel. Towards proving strong direct product theorems. *Computational Complexity*, 12 (1/2):1–22, 2004.

Detlef Sieling. Minimization of decision trees is hard to approximate. *Journal of Computer and System Sciences*, 74(3):394–403, 2008.

Leslie Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

Leslie G Valiant. Learning disjunction of conjunctions. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 560–566, 1985.

Wikipedia page for Decision tree learning. Wikipedia page for Decision tree learning. https://en.wikipedia.org/wiki/Decision_tree_learning.

Hans Zantema and Hans Bodlaender. Finding small equivalent decision trees is hard. *International Journal of Foundations of Computer Science*, 11(2):343–354, 2000.

Appendix A. Proof of Claim 3

Let T be the decision tree built iteratively by the following procedure. In the first iteration, pick an arbitrary $x \in f^{-1}(1)$ and fully query the indices in $\mathrm{Cert}(f, x)$. Let T_i be the tree formed after the ith iteration. Then T_{i+1} is formed by choosing $x \in f^{-1}(1)$ which has not been picked in a previous iteration. Then, at the leaf reached by x in T_i, fully query the indices in $\mathrm{Cert}(f, x)$ (ignoring those indices which have already been queried along the path followed by x in T_i). Repeating this for $|f^{-1}(1)|$ steps, yields a decision tree with at most $\sum_{x \in f^{-1}(1)} \mathrm{Cert}(f, x)$ *internal nodes*. Therefore, the number of leaves is at most $1 + \sum_{x \in f^{-1}(1)} \mathrm{Cert}(f, x)$.

It remains to show that this tree exactly computes f. Specifically, we'll argue that f_π is the constant function for every path π in the decision tree. If not, then there is an $x \in f^{-1}(1)$ so that x follows the path π and f_π is nonconstant. But this is a contradiction since π consists of a certificate of x by construction. ■

Appendix B. Hard distribution lemma: Proof of Lemma 11

First we prove a lemma which counts the error under $\mathcal{D}$ conditioned on the 1-input obtained in the first sampling step.

Lemma 21 (Error with respect to the canonical hard distribution conditioned on a 1-input) *Let T be a decision tree, $f : D \to \{0, 1\}$ be a function, and let $\mathcal{D}$ be the canonical hard distribution. For all $x \in f^{-1}(1)$,*

$$\Pr_{\boldsymbol{z} \sim \mathcal{D}_f}[T(\boldsymbol{z}) \neq f(\boldsymbol{z}) \mid \textit{first sampling } x \textit{ from } f^{-1}(1)] \geq \frac{1}{2} \cdot \frac{|\mathrm{Sens}(f_{\pi(x)}, x)|}{\max\{1, |\mathrm{Sens}(f, x)|\}}.$$

Proof If $|\mathrm{Sens}(f, x)| = 0$ then $|\mathrm{Sens}(f_{\pi(x)}, x)| = 0$. And if the RHS is 0 then the bound is vacuously true. Assume that $\mathrm{Sens}(f_{\pi(x)}, x) \neq \varnothing$. Both x and all $y \in \mathrm{Sens}(f_{\pi(x)}, x)$ follow the same path in T and have the same leaf label. Since $f(x) = 1$ and $f(y) = 0$, we can write

$$\begin{aligned}
&\Pr_{\boldsymbol{z} \sim \mathcal{D}_f}\left[T(\boldsymbol{z}) \neq f(\boldsymbol{z}) \mid \text{first sampling } x \text{ from } f^{-1}(1)\right] \\
&\qquad \geq \min\left\{ \Pr_{\boldsymbol{z} \sim \mathcal{D}_f}\left[\boldsymbol{z} = x \mid \text{first sampling } x \text{ from } f^{-1}(1)\right], \right. \\
&\qquad\qquad \left. \Pr_{\boldsymbol{z} \sim \mathcal{D}_f}\left[\boldsymbol{z} \in \mathrm{Sens}(f_{\pi(x)}, x) \mid \text{first sampling } x \text{ from } f^{-1}(1)\right] \right\} \\
&\qquad = \min\left\{ \frac{1}{2}, \frac{1}{2} \cdot \frac{|\mathrm{Sens}(f_{\pi(x)}, x)|}{|\mathrm{Sens}(f, x)|} \right\} \geq \frac{1}{2} \cdot \frac{|\mathrm{Sens}(f_{\pi(x)}, x)|}{|\mathrm{Sens}(f, x)|}
\end{aligned}$$

where the inequality follows from the fact that the probability of an error is lower bounded by the probability that $\boldsymbol{z} = x$ when the label for the path in T is 0 and is lower bounded by the probability that $\boldsymbol{z} \in \mathrm{Sens}(f_{\pi(x)}, x)$ when the label for the path is 1. Note that if $\mathrm{Sens}(f_{\pi(x)}, x) \neq \varnothing$ then necessarily $|\mathrm{Sens}(f, x)| \geq 1$ and $\max\{1, |\mathrm{Sens}(f, x)|\} = |\mathrm{Sens}(f, x)|$. Therefore, the proof is complete. ■

Proof [Proof of Lemma 11] We prove the statement for $C = f^{-1}(1)$. If $C \neq f^{-1}(1)$, we can consider the function $f : C \cup C' \to \{0, 1\}$ where C' denotes the set of sensitive neighbors of strings

in C, and the same proof holds. Let $\mathcal{D}$ denote the distribution from Definition 10 and notice that the support of this distribution is C and all of its sensitive neighbors. We have

$$\begin{aligned}
\mathrm{error}_{\mathcal{D}}(T, f) &= \Pr_{\boldsymbol{z}\sim\mathcal{D}}[T(\boldsymbol{z}) \neq f(\boldsymbol{z})] \\
&= \sum_{x\in C} \frac{1}{|C|} \cdot \Pr_{\boldsymbol{z}\sim\mathcal{D}}[T(\boldsymbol{z}) \neq f(\boldsymbol{z}) \mid \text{ first sampling } x \text{ from } f^{-1}(1)] \\
&\geq \frac{1}{2|C|} \sum_{x\in C} \frac{|\mathrm{Sens}(f_{\pi(x)}, x)|}{\max\{1, |\mathrm{Sens}(f, x)|\}} && \text{(Lemma 21)} \\
&\geq \frac{1}{2|C| \cdot \mathrm{Sens}(f)} \sum_{x\in C} |\mathrm{Sens}(f_{\pi(x)}, x)| \quad (\max\{1, |\mathrm{Sens}(f, x)|\} \leq \mathrm{Sens}(f) \text{ for all } x)
\end{aligned}$$

where the last step uses the fact that $\mathrm{Sens}(f) \geq 1$ since f is nonconstant. In particular, we have that $\max\{1, |\mathrm{Sens}(f, x)|\} \leq \mathrm{Sens}(f)$ for all x. ■

Appendix C. Deferred proofs for Theorem 12

C.1. Proof of Lemma 16

First, we give a basic zero-error lower bound on ℓ-IsEdge and observe some properties about the sensitivity and certificate complexity of ℓ-IsEdge in Lemma 22, Theorem 23, and Proposition 24, respectively.

Lemma 22 (Zero-error lower bound for ℓ-IsEdge_G; see (Koch et al., 2023a, Claim 6.6)) *Let G be an n-vertex, m-edge graph where every vertex cover has size at least k'. Then, any decision tree T computing ℓ-$\mathrm{IsEdge}_G : \ell\text{-}D_G \to \{0,1\}$ over ℓ-D_G, the support of the canonical hard distribution, must have size*

$$|T| \geq (\ell + 1)(k' + m).$$

Proof The same lower bound is proved in Claim 6.6 of Koch et al. (2023a) under a slightly different subset of inputs. Specifically, they prove $\mathrm{DT}(\ell\text{-}\mathrm{IsEdge}_G) \geq (\ell+1)(k'+m)$ where ℓ-$\mathrm{IsEdge}_G : D' \to \{0,1\}$ for the set $D' = \ell\text{-}D_G \cup \{0^{n+\ell n}\}$ which adds the all 0s input. This small difference doesn't change the lower bound since any decision tree T computing ℓ-IsEdge_G over the set ℓ-D_G also computes it over D'. Indeed, every 1-input to ℓ-IsEdge_G is sensitive on every 1-coordinate and so if T satisfies $T(x) = \ell\text{-}\mathrm{IsEdge}_G(x)$ for every $x \in \ell\text{-}D_G$, then it must query every 1-coordinate of each 1-input. Therefore, we can assume without loss of generality that $T(0^{n+\ell n}) = 0$. ■

Proposition 23 (Sensitivity of ℓ-IsEdge_G) *For a graph G, $\ell \geq 1$, and ℓ-$\mathrm{IsEdge}_G : \ell\text{-}D_G \to \{0,1\}$, we have*

$$\mathrm{Sens}(\ell\text{-}\mathrm{IsEdge}_G) = 2(\ell + 1).$$

Proof Let ℓ-$\mathrm{Ind}[e]$ be an edge indicator for an edge $e \in E$. Let $i \in [n\ell + n]$ denote the index of a 1-coordinate of ℓ-$\mathrm{Ind}[e]$. By definition, there are $2(\ell+1)$ many such i and each i is sensitive: $\ell\text{-}\mathrm{IsEdge}_G(\ell\text{-}\mathrm{Ind}[e]^{\oplus i}) = 0$. Therefore, $|\mathrm{Sens}(\ell\text{-}\mathrm{IsEdge}_G, \ell\text{-}\mathrm{Ind}[e])| = 2(\ell+1)$. Conversely, for every sensitive neighbor $\ell\text{-}\mathrm{Ind}[e]^{\oplus i}$, we have $\mathrm{Sens}(\ell\text{-}\mathrm{IsEdge}_G, \ell\text{-}\mathrm{Ind}[e]^{\oplus i}) = \{\ell\text{-}\mathrm{Ind}[e]\}$ and so $|\mathrm{Sens}(\ell\text{-}\mathrm{IsEdge}_G, \ell\text{-}\mathrm{Ind}[e]^{\oplus i})| = 1$. Thus the overall sensitivity is $\mathrm{Sens}(\ell\text{-}\mathrm{IsEdge}_G) = 2(\ell+1)$. ■

Proposition 24 (Sensitivity equals certificate complexity of 1-inputs) *Let G be a graph and ℓ-IsEdge : ℓ-$D_G \to \{0,1\}$, the corresponding edge function. For all edge indicators $x = \ell\text{-Ind}[e]$ and for all restrictions π, we have*

$$\mathrm{Cert}(\ell\text{-IsEdge}_\pi, x) = |\mathrm{Sens}(\ell\text{-IsEdge}_\pi, x)|.$$

Proof By definition $|\mathrm{Sens}(\ell\text{-IsEdge}_\pi, x)|$ is the number of 1-coordinates in x which are not restricted by π. The set of 1-coordinates of x not restricted by π forms a certificate of $\ell\text{-IsEdge}_\pi$ since fixing these coordinates forces ℓ-IsEdge to be the constant 1-function. It follows that $\mathrm{Cert}(\ell\text{-IsEdge}_\pi, x) = |\mathrm{Sens}(\ell\text{-IsEdge}_\pi, x)|$. ■

Proof [Proof of Lemma 16] For a graph consisting of m edges, the number of 1-inputs to ℓ-IsEdge : ℓ-$D_G \to \{0,1\}$ is m. Therefore,

$$\begin{aligned}
\varepsilon &\geq \frac{1}{2m \cdot \mathrm{Sens}(\ell\text{-IsEdge})} \sum_{x \in \ell\text{-IsEdge}^{-1}(1)} |\mathrm{Sens}(\ell\text{-IsEdge}_{\pi(x)}, x)| && \text{(Lemma 11)} \\
&= \frac{1}{2m \cdot \mathrm{Sens}(\ell\text{-IsEdge})} \sum_{x \in \ell\text{-IsEdge}^{-1}(1)} \mathrm{Cert}(\ell\text{-IsEdge}_{\pi(x)}, x) && \text{(Proposition 24)} \\
&\geq \frac{1}{2m \cdot \mathrm{Sens}(\ell\text{-IsEdge})} \left(\mathrm{DT}(\ell\text{-IsEdge}, \ell\text{-}D_G) - |T|\right). && \text{(Lemma 9)}
\end{aligned}$$

Rearranging the above, we obtain

$$\begin{aligned}
|T| &\geq \mathrm{DT}(\ell\text{-IsEdge}, \ell\text{-}D_G) - 2\varepsilon m \cdot \mathrm{Sens}(\ell\text{-IsEdge}) \\
&\geq (\ell+1)(k'+m) - 2\varepsilon m \cdot \mathrm{Sens}(\ell\text{-IsEdge}) && \text{(Lemma 22)} \\
&= (\ell+1)(k'+m) - 4\varepsilon m(\ell+1) && \text{(Theorem 23)}
\end{aligned}$$

which completes the proof. ■

C.2. Proof of Lemma 17

This lemma is a consequence of the following proposition along with the upper and lower bounds we have obtained for ℓ-IsEdge. The proposition is a calculation involving the parameters that come into play in Lemma 17. We state it on its own, since we will reuse the calculation later when proving Theorem 2.

Proposition 25 *For all $\delta, \delta', \alpha > 0$ and $\ell, m, n, k, d \geq 1$ satisfying $m \leq dk$ and $\delta' > (\delta+\alpha)d + \delta + \frac{(1+\delta)mn}{k(\ell+1)}$, we have*

$$(1+\delta)\left[(\ell+1)(k+m) + mn\right] < (\ell+1)\left[(1+\delta')k + (1-\alpha)m\right].$$

Proof The proof is a calculation. We can write

$$\begin{aligned}
(1+\delta')k - (1+\delta)k &= (\delta'-\delta)k \\
&> \left(\delta + (\delta+\alpha)d + \frac{(1+\delta)mn}{k(\ell+1)} - \delta\right)k && \text{(Assumption on } \delta') \\
&= (\delta+\alpha)dk + \frac{(1+\delta)mn}{\ell+1} \\
&\geq (\delta+\alpha)m + \frac{(1+\delta)mn}{\ell+1} && (m \leq dk) \\
&= (1+\delta)m - (1-\alpha)m + \frac{(1+\delta)mn}{\ell+1}.
\end{aligned}$$

Therefore, rearranging we obtain

$$(1+\delta)\left[(\ell+1)(k+m)+mn\right] < (\ell+1)\left[(1+\delta')k + (1-\alpha)m\right]$$

which completes the proof. ■

Proof [Proof of Lemma 17] Given a degree-d, m-edge, n-vertex graph G and parameter k, we choose $\ell = \Theta(n)$ so that $\delta' > (\delta+4\varepsilon)d + \delta + \frac{(1+\delta)mn}{k(\ell+1)}$ and set $s = (\ell+1)(k+m)+mn$. Note that such an ℓ exists since $k = \Theta(n)$ for constant-degree graphs. We now prove the two points separately.

Yes case. In this case, we have by Theorem 14 that there is a decision tree T computing ℓ-IsEdge : $\{0,1\}^{n\ell+n} \to \{0,1\}$ whose size satisfies

$$|T| \leq (\ell+1)(k+m)+mn = s.$$

No case. Let T be a decision tree satisfying $\mathrm{error}_{\ell\text{-}\mathcal{D}_G}(T, \ell\text{-IsEdge}) \leq \varepsilon$. Then, using our assumptions on the parameters:

$$\begin{aligned}
(1+\delta)s &= (1+\delta)\left[(\ell+1)(k+m)+mn\right] && \text{(Definition of } s) \\
&< (\ell+1)\left[(1+\delta')k + (1-4\varepsilon)m\right] && \text{(Proposition 25 with } \alpha = 4\varepsilon) \\
&\leq |T|. && \text{(Lemma 16 with } k' = (1+\delta')k)
\end{aligned}$$

We've shown the desired bounds in both the Yes and No cases so the proof is complete. ■

C.3. Proof of Theorem 12

Let G be a constant degree-d, n-vertex graph and $k \in \mathbb{N}$, a parameter. Let $\mathcal{A}$ be the algorithm for DT-LEARN from the theorem statement. We'll use $\mathcal{A}$ to approximate VERTEX COVER on G.

The reduction. First, we check whether $dk \geq m$. If $dk < m$, our algorithm outputs "No" as G cannot have a vertex cover of size at most k. Otherwise, we proceed under the assumption that $dk \geq m$. Let $s \in \mathbb{N}$ be the quantity from Lemma 17. We will run $\mathcal{A}$ over the distribution ℓ-$\mathcal{D}_G$ and on the function ℓ-IsEdge$_G$: $\{0,1\}^N \to \{0,1\}$ where ℓ is as in Lemma 17. Note that $N = n\ell + n = O(n^2)$ and $s = O(n^2) = O(N)$. See Figure 3 for the exact procedure we run.

VERTEX COVER$(k, (1+\delta')\cdot k)$:

Given: G, an m-edge degree-d graph over n vertices and $k \in \mathbb{N}$

Run: DT-LEARN$(s, (1+\delta)\cdot s, \varepsilon)$ for $t(N, 1/\varepsilon)$ time steps providing the learner with

- *queries*: return ℓ-IsEdge$(v^{(0)}, \ldots, v^{(\ell)})$ for a query $(v^{(0)}, \ldots, v^{(\ell)}) \in \{0,1\}^N$; and
- *random samples*: return $(\boldsymbol{v}^{(0)}, \ldots, \boldsymbol{v}^{(\ell)}) \sim \ell\text{-}\mathcal{D}_G$ for a random sample.

$T_{\text{hyp}} \leftarrow$ decision tree output of the learner

$\varepsilon_{\text{hyp}} \leftarrow \text{error}_{\ell\text{-}\mathcal{D}_G}(T_{\text{hyp}}, \ell\text{-IsEdge})$

Output: YES if and only if $|T_{\text{hyp}}| \le (1+\delta)\cdot s$ and $\varepsilon_{\text{hyp}} \le \varepsilon$

Figure 3: Using an algorithm for DT-LEARN to solve VERTEX COVER.

Runtime. Any query to ℓ-IsEdge can be answered in $O(N)$ time. Similarly, a random sample can be obtained in $O(N)$ time. The algorithm uses $O(N \cdot t(N, 1/\varepsilon))$ time to run DT-LEARN. Finally, computing $\text{error}_{\ell\text{-}\mathcal{D}_G}(T_{\text{hyp}}, \ell\text{-IsEdge})$ takes $O(N^2)$. Since $t(N, 1/\varepsilon) \ge N$, the overall runtime is $O(N \cdot t(N, 1/\varepsilon)) = O(n^2 t(n^2, 1/\varepsilon))$.

Correctness. Correctness follows from Lemma 17. Specifically, in the **Yes case**, if G has a vertex cover of size at most k, then there is a decision tree of size at most s computing ℓ-IsEdge. Therefore, by the guarantees of DT-LEARN, we have $|T_{\text{hyp}}| \le (1+\delta)\cdot s$ and $\varepsilon_{\text{hyp}} \le \varepsilon$ and our algorithm correctly outputs "Yes".

In the **No case**, every vertex cover of G has size at least $(1+\delta')k$. If $\varepsilon_{\text{hyp}} > \varepsilon$ then our algorithm for VERTEX COVER correctly outputs "No". Otherwise, assume that $\varepsilon_{\text{hyp}} \le \varepsilon$. Then, Lemma 17 ensures that $(1+\delta)s < |T_{\text{hyp}}|$ and so our algorithm correctly outputs "No" in this case as well. This completes the proof. ■

Appendix D. Proof of Lemma 18

We require the following generalization of a result from Savickỳ (2002). Savický proved that for functions $f_1 : \{0,1\}^n \to \{0,1\}$ and $f_2 : \{0,1\}^n \to \{0,1\}$, it holds that $\text{DT}(f_1 \oplus f_2) \ge \text{DT}(f_1) \cdot \text{DT}(f_2)$ (Savickỳ, 2002, Lemma 2.1). We will use the following analogous statement for partial functions.

Theorem 26 (Generalization of Savický (Savickỳ, 2002)) *Let $f^{(1)}, \ldots, f^{(r)}$ be functions, $f^{(i)} : D^{(i)} \to \{0,1\}$ with $D^{(i)} \subseteq \{0,1\}^{n^{(i)}}$ for each $i = 1, \ldots, r$. Then,*

$$\text{DT}(f^{(1)} \oplus \cdots \oplus f^{(r)}) = \prod_{i=1}^{r} \text{DT}(f^{(i)}).$$

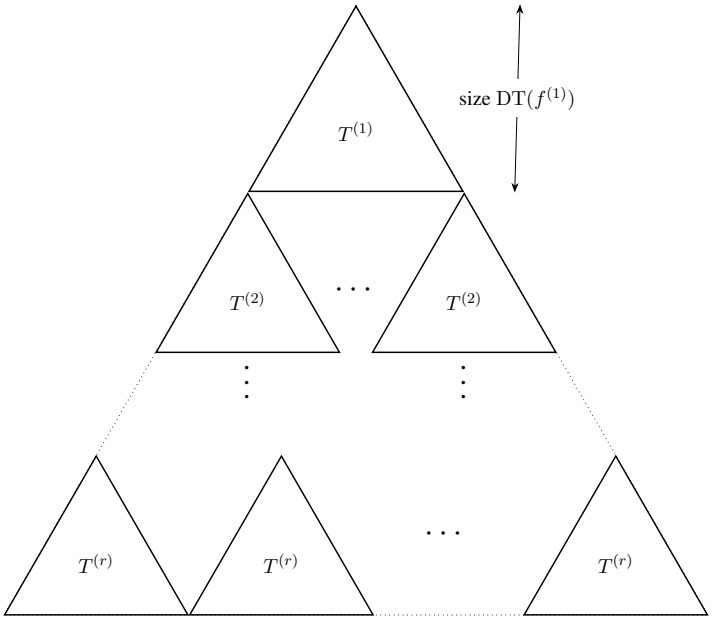

Figure 4: Illustration of a stacked decision tree for a function $f^{(1)} \oplus \cdots \oplus f^{(r)}$. For an input $x = (x^{(1)}, \ldots, x^{(r)})$, the decision tree sequentially computes $f^{(i)}(x^{(i)})$ for each $i = 1, \ldots, r$ using a decision tree $T^{(i)}$ of size $\mathrm{DT}(f^{(i)})$ for $f^{(i)}$. Then at the leaf it outputs $f^{(1)}(x^{(1)}) \oplus \cdots \oplus f^{(r)}(x^{(r)})$. The overall size of the decision tree is $\prod_{i=1}^{r} \mathrm{DT}(f^{(i)})$.

Proof First, the upper bound $\mathrm{DT}(f^{(1)} \oplus \cdots \oplus f^{(r)}) \leq \prod_{i=1}^{r} \mathrm{DT}(f^{(i)})$ follows by considering the decision tree for $f^{(1)} \oplus \cdots \oplus f^{(r)}$ which sequentially computes $f^{(i)}(x)$ for each $i = 1, \ldots, r$ using a decision tree of size $\mathrm{DT}(f^{(i)})$. See Figure 4 for an illustration of this decision tree.

The lower bound is by induction on $\sum_{i \in [r]} n^{(i)}$, the total number of input variables. In the base case, $n = 0$ and the bound is trivially true: the constant function requires a decision tree of size 1. For the inductive step, let T be a decision tree for $f^{(1)} \oplus \cdots \oplus f^{(r)}$ of size $\mathrm{DT}(f^{(1)} \oplus \cdots \oplus f^{(r)})$, and let x_j be the variable queried at the root. Assume without loss of generality that x_j belongs to $f^{(1)}$. The subfunctions computed at the left and right branches of the root of T are $f^{(1)}_{x_j \leftarrow 0} \oplus \cdots \oplus f^{(r)}$ and $f^{(1)}_{x_j \leftarrow 1} \oplus \cdots \oplus f^{(r)}$, respectively. Each is a function on $\left(\sum_{i \in [r]} n^{(i)}\right) - 1$ many variables and

so we can apply the inductive hypothesis. Therefore:

$$\begin{aligned}
\mathrm{DT}(f^{(1)} \oplus \cdots \oplus f^{(r)}) &= |T| \\
&\geq \mathrm{DT}(f^{(1)}_{x_j \leftarrow 0} \oplus \cdots \oplus f^{(r)}) + \mathrm{DT}(f^{(1)}_{x_j \leftarrow 1} \oplus \cdots \oplus f^{(r)}) \\
&\qquad \text{(Root of } T \text{ is } x_j^{(1)}\text{)} \\
&\geq \mathrm{DT}(f^{(1)}_{x_j \leftarrow 0}) \prod_{i=2}^{r} \mathrm{DT}(f^{(i)}) + \mathrm{DT}(f^{(1)}_{x_j \leftarrow 1}) \prod_{i=2}^{r} \mathrm{DT}(f^{(i)}) \\
&\qquad \text{(Inductive hypothesis)} \\
&= \left(\mathrm{DT}(f^{(1)}_{x_j \leftarrow 0}) + \mathrm{DT}(f^{(1)}_{x_j \leftarrow 1})\right) \prod_{i=2}^{r} \mathrm{DT}(f^{(i)}) \\
&\geq \prod_{i=1}^{r} \mathrm{DT}(f^{(i)})
\end{aligned}$$

where the last step follows from the fact that $\mathrm{DT}(f^{(1)}_{x_j \leftarrow 0}) + \mathrm{DT}(f^{(1)}_{x_j \leftarrow 1}) \geq \mathrm{DT}(f^{(1)})$. Indeed, one can construct a decision tree for $f^{(1)}$ of size $\mathrm{DT}(f^{(1)}_{x_j \leftarrow 0}) + \mathrm{DT}(f^{(1)}_{x_j \leftarrow 1})$ by querying x_j at the root and on the left branch placing a tree for $f^{(1)}_{x_j \leftarrow 0}$ and on the right branch placing a tree for $f^{(1)}_{x_j \leftarrow 1}$. ■

Proof [Proof of Lemma 18] Let Π denote the set of paths. For each path $\pi \in \Pi$, we write $\pi^{(i)}$ for $i \in [r]$ to denote the part of π corresponding to the ith block of input variables. This way, the restricted function $f_\pi^{\oplus r}$ corresponds to the function $f_{\pi^{(1)}} \oplus \cdots \oplus f_{\pi^{(r)}}$ Then we have

$$\begin{aligned}
\mathrm{DT}(f_\pi^{\oplus r}) &\leq \sum_{\pi \in \Pi} \mathrm{DT}(f_\pi^{\oplus r}) \\
&= \sum_{\substack{\pi \in \Pi \\ f_\pi^{\oplus r} \text{ is constant}}} \mathrm{DT}(f_\pi^{\oplus r}) + \sum_{\substack{\pi \in \Pi \\ f_\pi^{\oplus r} \text{ is nonconstant}}} \mathrm{DT}(f_\pi^{\oplus r}) \\
&= |T| + \sum_{\substack{\pi \in \Pi \\ f_\pi^{\oplus r} \text{ is nonconstant}}} \mathrm{DT}(f_\pi^{\oplus r}) \qquad (\mathrm{DT}(f_\pi^{\oplus r}) = 1 \text{ when } f_\pi^{\oplus r} \text{ is constant}) \\
&= |T| + \sum_{\substack{\pi \in \Pi \\ f_\pi^{\oplus r} \text{ is nonconstant}}} \prod_{i=1}^{r} \mathrm{DT}(f_{\pi^{(i)}}) \qquad \text{(Theorem 26)} \\
&\leq |T| + \sum_{\substack{\pi \in \Pi \\ f_\pi^{\oplus r} \text{ is nonconstant}}} \prod_{i=1}^{r} \left(1 + \sum_{x : f_{\pi^{(i)}}(x) = 1} \mathrm{Cert}(f_{\pi^{(i)}}, x)\right) \qquad \text{(Claim 3)}
\end{aligned}$$

We use the fact that $1 + a \le 2\max\{1, a\}$ for all $a \in \mathbb{R}$ to rewrite the above in a simpler form, while suffering a factor of 2^r:

$$\begin{aligned}
|T| + &\sum_{\substack{\pi \in \Pi \\ f_\pi^{\oplus r} \text{ is nonconstant}}} \prod_{i=1}^{r} \left(1 + \sum_{x : f_{\pi(i)}(x) = 1} \mathrm{Cert}(f_{\pi(i)}, x)\right) \\
&\le |T| + \sum_{\substack{\pi \in \Pi \\ f_\pi^{\oplus r} \text{ is nonconstant}}} \prod_{i=1}^{r} \left(2\max\{1, \sum_{x : f_{\pi(i)}(x) = 1} \mathrm{Cert}(f_{\pi(i)}, x)\}\right) \\
&\le |T| + 2^r \sum_{\substack{\pi \in \Pi \\ f_\pi^{\oplus r} \text{ is nonconstant}}} \prod_{i=1}^{r} \left(\sum_{x : f_{\pi(i)}(x) = 1} \max\{1, \mathrm{Cert}(f_{\pi(i)}, x)\}\right) \\
&= |T| + 2^r \sum_{\substack{\pi \in \Pi \\ f_\pi^{\oplus r} \text{ is nonconstant}}} \sum_{x^{(1)} : f_{\pi(1)}(x^{(1)}) = 1} \cdots \sum_{x^{(r)} : f_{\pi(r)}(x^{(r)}) = 1} \prod_{i=1}^{r} \max\{1, \mathrm{Cert}(f_{\pi(i)}, x^{(i)})\} \\
&= |T| + 2^r \sum_{\substack{x \in f^{-1}(1)^r \\ f_{\pi(x)}^{\oplus r} \text{ is nonconstant}}} \prod_{i=1}^{r} \max\{1, \mathrm{Cert}(f_{\pi(i)}, x^{(i)})\}
\end{aligned}$$

where the last equality follows from the fact that Π partitions the input space and so for every $x \in f^{-1}(1)^r$ there is exactly one path $\pi \in \Pi$ such that $f_{\pi(i)}(x^{(i)}) = 1$ for all $i \in [r]$. ■

Appendix E. Proof of Lemma 19

Lemma 27 (Error of $f^{\oplus r}$ conditioned on an input) *Let T be a decision tree, $r \ge 1$, $f : D \to \{0,1\}$ be a nonconstant function, $x \in f^{-1}(1)^r$, $\mathcal{D}$ be the canonical hard distribution, and π be the path followed by x in T and assume that $\mathrm{Sens}(f_{\pi(x)}^{\oplus r}, x) \neq \varnothing$, then*

$$\Pr_{\boldsymbol{z} \sim \mathcal{D}^{\otimes r}}[f^{\oplus r}(\boldsymbol{z}) \neq T(\boldsymbol{z}) \mid \textit{first sampling } x] \ge 2^{-r} \prod_{i=1}^{r} \frac{\max\{1, |\mathrm{Sens}(f_\pi^{(i)}, x^{(i)})|\}}{\max\{1, |\mathrm{Sens}(f^{(i)}, x^{(i)})|\}}.$$

Proof Let $y \in \mathrm{Sens}(f_\pi^{\oplus r}, x)$ and let $j \in [r]$ be such that $y^{(j)} \in \mathrm{Sens}(f_\pi^{(j)}, x^{(j)})$. Since $f^{\oplus r}(y) \neq f^{\oplus r}(x)$, we have

$$\begin{aligned}
&\Pr_{\boldsymbol{z}\sim\mathcal{D}^{\otimes r}}[f^{\oplus r}(\boldsymbol{z}) \neq T(\boldsymbol{z}) \mid \text{ first sampling } x] \\
&\quad \geq \min\Big\{ \Pr_{\boldsymbol{z}\sim\mathcal{D}^{\otimes r}}[\boldsymbol{z} = x \mid \text{ first sampling } x], \\
&\qquad \Pr_{\boldsymbol{z}\sim\mathcal{D}^{\otimes r}}[\boldsymbol{z} = (x^{(1)}, ..., y, ..., x^{(r)}) \text{ for } y \in \mathrm{Sens}(f_\pi^{(j)}, x^{(j)}) \mid \text{ first sampling } x]\Big\} \\
&\qquad\qquad \text{(Either } x \text{ or } (x^{(1)}, ..., y, ..., x^{(r)}) \text{ makes an error)} \\
&\quad = \min\left\{ 2^{-r}, 2^{-(r-1)} \frac{|\mathrm{Sens}(f_\pi^{(j)}, x^{(j)})|}{|\mathrm{Sens}(f^{(j)}, x^{(j)})|} \right\} \qquad \text{(Definition of } \mathcal{D}^{\otimes r}) \\
&\quad \geq 2^{-r} \frac{|\mathrm{Sens}(f_\pi^{(j)}, x^{(j)})|}{|\mathrm{Sens}(f^{(j)}, x^{(j)})|} \\
&\quad \geq 2^{-r} \frac{|\mathrm{Sens}(f_\pi^{(j)}, x^{(j)})|}{|\mathrm{Sens}(f^{(j)}, x^{(j)})|} \cdot \prod_{i\in[r]\setminus\{j\}} \frac{\max\{1, |\mathrm{Sens}(f_\pi^{(i)}, x^{(i)})|\}}{\max\{|\mathrm{Sens}(f^{(i)}, x^{(i)})|\}} \\
&\qquad\qquad (\max\{1, |\mathrm{Sens}(f_\pi^{(i)}, x^{(i)})|\} \leq \max\{1, |\mathrm{Sens}(f^{(i)}, x^{(i)})|\} \text{ for all } i) \\
&\quad = 2^{-r} \prod_{i=1}^{r} \frac{\max\{1, |\mathrm{Sens}(f_\pi^{(i)}, x^{(i)})|\}}{\max\{1, |\mathrm{Sens}(f^{(i)}, x^{(i)})|\}}. \qquad (|\mathrm{Sens}(f_\pi^{(j)}, x^{(j)})| \neq 0 \text{ by assumption})
\end{aligned}$$

■

We can now prove the main result of this section.

Proof [Proof of Lemma 19] As in the case of Lemma 11, we assume without loss of generality that $C = f^{-1}(1)$. We lower bound the error as follows

$$\begin{aligned}
\varepsilon &\geq \Pr_{\boldsymbol{z}\sim\mathcal{D}^{\otimes r}}[T(\boldsymbol{z}) \neq f^{\oplus r}(\boldsymbol{z})] \\
&= \sum_{\substack{x\in C^r \\ \mathrm{Sens}(f_{\pi(x)}^{\oplus r}, x)\neq\varnothing}} \frac{1}{|C|^r} \cdot \Pr_{\boldsymbol{z}\sim\mathcal{D}^{\otimes r}}[T(\boldsymbol{z}) \neq f^{\oplus r}(\boldsymbol{z}) \mid \text{ first sampling } x \text{ from } C^r] \\
&\geq (2|C|)^{-r} \sum_{\substack{x\in C^r \\ \mathrm{Sens}(f_{\pi(x)}^{\oplus r}, x)\neq\varnothing}} \prod_{i=1}^{r} \frac{\max\{1, |\mathrm{Sens}(f_\pi^{(i)}, x^{(i)})|\}}{\max\{1, |\mathrm{Sens}(f^{(i)}, x^{(i)})|\}} \qquad \text{(Lemma 27)} \\
&\geq (2|C| \cdot \mathrm{Sens}(f))^{-r} \sum_{\substack{x\in C^r \\ \mathrm{Sens}(f_{\pi(x)}^{\oplus r}, x)\neq\varnothing}} \prod_{i=1}^{r} \max\{1, |\mathrm{Sens}(f_\pi^{(i)}, x^{(i)})|\} \\
&\qquad\qquad (|\mathrm{Sens}(f^{(i)}, x^{(i)})| \leq \mathrm{Sens}(f))
\end{aligned}$$

which completes the proof. ■

Appendix F. Proof of Theorem 20

Before proving this theorem, we establish a few properties of ℓ-IsEdge$^{\oplus r}$ which will be helpful for our analysis.

Theorem 28 (Decision tree size lower bound for computing ℓ-IsEdge$^{\oplus r}$) *Let T be a decision tree for ℓ-IsEdge$_G^{\oplus r}$ with $\ell, r \geq 1$. Let k be the minimum vertex cover size of G and let m denote the number of edges of G. Then, if $\mathrm{error}_{\ell\text{-}\mathcal{D}_G^{\otimes r}}(T, \ell\text{-IsEdge}^{\oplus r}) \leq \varepsilon$ for the canonical hard distribution ℓ-$\mathcal{D}_G$, we have*

$$|T| \geq \big[(\ell+1)(k+m)\big]^r - \varepsilon\big[8m(\ell+1)\big]^r$$

Proof Since ℓ-IsEdge has m many 1-inputs over the dataset ℓ-D_G and $\mathrm{Sens}(\ell\text{-IsEdge}) = 2(\ell+1)$, we have

$$\begin{aligned}
\varepsilon &\geq \left(\frac{1}{4m(\ell+1)}\right)^r \sum_{\substack{x \in \ell\text{-}D_G^r \\ \mathrm{Sens}(\ell\text{-IsEdge}^{\oplus r}_{\pi(x)}, x) \neq \varnothing}} \prod_{i=1}^{r} \max\{1, \mathrm{Sens}(\ell\text{-IsEdge}^{(i)}_{\pi(x)}, x^{(i)})\} && \text{(Lemma 19)} \\
&= \left(\frac{1}{4m(\ell+1)}\right)^r \sum_{\substack{x \in \ell\text{-}D_G^r \\ \mathrm{Sens}(\ell\text{-IsEdge}^{\oplus r}_{\pi(x)}, x) \neq \varnothing}} \prod_{i=1}^{r} \max\{1, \mathrm{Cert}(\ell\text{-IsEdge}^{(i)}_{\pi(x)}, x^{(i)})\} && \text{(Proposition 24)} \\
&\geq \left(\frac{1}{4m(\ell+1)}\right)^r \sum_{\substack{x \in \ell\text{-}D_G^r \\ \ell\text{-IsEdge}^{\oplus r}_{\pi(x)} \text{ is nonconstant}}} \prod_{i=1}^{r} \max\{1, \mathrm{Cert}(\ell\text{-IsEdge}^{(i)}_{\pi(x)}, x^{(i)})\} \\
&\geq \left(\frac{1}{8m(\ell+1)}\right)^r (\mathrm{DT}(\ell\text{-IsEdge}^{\oplus r}) - |T|). && \text{(Lemma 18)}
\end{aligned}$$

In this derivation, we used the fact that if an input $x \in \ell$-D_G^r is such that ℓ-IsEdge$^{\oplus r}_{\pi(x)}$ is nonconstant, then it must be the case that there is some block $i \in [r]$ where the path π does not fully restrict the sensitive coordinates in the edge indicator for block i, and therefore it must also be the case that $\mathrm{Sens}(\ell\text{-IsEdge}^{\oplus r}_{\pi(x)}, x) \neq \varnothing$. Now, we can rearrange this lower bound on ε to obtain:

$$\begin{aligned}
|T| &\geq \mathrm{DT}(\ell\text{-IsEdge}^{\oplus r}) - \varepsilon\big[8m(\ell+1)\big]^r \\
&= \mathrm{DT}(\ell\text{-IsEdge})^r - \varepsilon\big[8m(\ell+1)\big]^r && \text{(Theorem 26)} \\
&\geq \big[(\ell+1)(k+m)\big]^r - \varepsilon\big[8m(\ell+1)\big]^r && \text{(Lemma 22)}
\end{aligned}$$

which completes the proof. ■

The following proposition allows us to translate the above lower bound into a slightly simpler form.

Proposition 29 *For all $a, b, r > 0$ such that $a \geq b$, we have $a^r - b^r \geq (a-b)^r$.*

Proof Since $a \geq b$, we have

$$1 \geq \left(1 - \frac{b}{a}\right)^r + \left(\frac{b}{a}\right)^r.$$

Multiplying both sides of the inequality by a^r and rearranging gives the desired bound. ■

With Theorems 28 and 29, we are able to prove the main technical lemma used for our reduction.

Lemma 30 (Main technical lemma for Theorem 20) *For all $\delta, \delta', \varepsilon > 0$ and $d, k, r \geq 1$, the following holds. Given a constant degree-d graph G with m edges, n vertices, and parameter k, there is a choice of $\ell = \Theta(n)$ and a polynomial-time computable quantity $s \in \mathbb{N}$ such that so long as $\delta' > (\delta + 8\varepsilon^{1/r})d + \delta$, $dk \geq m$, and $\varepsilon < 2^{-3r}$ we have:*

- ***Yes case**: if G has a vertex cover of size at most k, then there is a decision tree of size at most s which computes ℓ-IsEdge$^{\oplus r} : \{0,1\}^{r(n\ell+n)} \to \{0,1\}$; and*
- ***No case**: if every vertex cover of G has size at least $(1+\delta')k$, then $(1+\delta)^r s < |T|$ for any decision tree T with $\mathrm{error}_{\ell\text{-}\mathcal{D}_G^{\otimes r}}(T, \ell\text{-IsEdge}) \leq \varepsilon$.*

Proof Given a degree-d, m-edge, n-vertex graph G and parameter k, we choose $\ell = \Theta(n)$ so that $\delta' > (\delta + 8\varepsilon^{1/r})d + \delta + \frac{(1+\delta)mn}{k(\ell+1)}$ and set $s = [(\ell+1)(k+m) + mn]^r$. Note that such an ℓ exists since $k = \Theta(n)$ for constant-degree graphs. We now prove the two points separately.

Yes case. In this case, we have

$$\begin{aligned} \mathrm{DT}(\ell\text{-IsEdge}^{\oplus r}) &= \mathrm{DT}(\ell\text{-IsEdge})^r && \text{(Theorem 26)} \\ &\leq [(\ell+1)(k+m) + mn]^r = s. && \text{(Theorem 14)} \end{aligned}$$

No case. In this case, let T be a decision tree with $\mathrm{error}_{\ell\text{-}\mathcal{D}_G^{\otimes r}}(T, \ell\text{-IsEdge}) \leq \varepsilon$. Then we have

$$\begin{aligned} (1+\delta)^r s &= \Big[(1+\delta)[(\ell+1)(k+m) + mn]\Big]^r \\ &< \Big[(\ell+1)(1+\delta')k + (\ell+1)(1 - 8\varepsilon^{1/r})m\Big]^r && \text{(Proposition 25 with } \alpha = 8\varepsilon^{1/r}) \\ &\leq \big[(\ell+1)(k+m)\big]^r - \varepsilon\big[8m(\ell+1)\big]^r && \text{(Theorem 29)} \\ &\leq |T|. && \text{(Theorem 28)} \end{aligned}$$

We've shown the desired bounds in both the Yes and No cases so the proof is complete. ■

Proof [Proof of Theorem 20] Let $\mathcal{A}$ be the algorithm for DT-LEARN from the theorem statement. Given an n-vertex, m-edge graph G of constant degree d, we'll use $\mathcal{A}$ to approximate VERTEX COVER on G.

The reduction. First, we check whether $dk \geq m$. If $dk < m$, our algorithm outputs "No" as G cannot have a vertex cover of size at most k (see Fact 1). Otherwise, we proceed under the assumption that $dk \geq m$. Let $s \in \mathbb{N}$ be the quantity from Theorem 30. We run $\mathcal{A}$ over the distribution $\ell\text{-}\mathcal{D}_G^{\otimes r}$ and on the function $\ell\text{-IsEdge}^{\oplus r} : \{0,1\}^N \to \{0,1\}$ where ℓ is as in Theorem 30. Note that $N = r \cdot (n\ell + n) = O(rn^2)$ and $s = O(n^{2r})$. See Figure 5 for the exact procedure we run.

VERTEX COVER$(k, (1+\delta') \cdot k)$:

Given: G, an m-edge degree-d graph over n vertices and $k \in \mathbb{N}$

Run: DT-LEARN$(s, A \cdot s, \varepsilon)$ for $t(s, 1/\varepsilon)$ time steps providing the learner with

- *queries*: return ℓ-IsEdge$^{\oplus r}(x)$ for a query $x \in \{0,1\}^N$; and
- *random samples*: return $\boldsymbol{x} \sim \ell\text{-}\mathcal{D}_G^{\otimes r}$ for a random sample.

$T_{\text{hyp}} \leftarrow$ decision tree output of the learner

$\varepsilon_{\text{hyp}} \leftarrow \text{error}_{\ell\text{-}\mathcal{D}_G^{\otimes r}}(T_{\text{hyp}}, \ell\text{-IsEdge}^{\oplus r})$

Output: YES if and only if $|T_{\text{hyp}}| \le A \cdot s$ and $\varepsilon_{\text{hyp}} \le \varepsilon$

Figure 5: Using an algorithm for DT-LEARN on ℓ-IsEdge$^{\oplus r}$ to solve VERTEX COVER.

Runtime. Any query to ℓ-IsEdge$_G^{\oplus r}$ can be answered in $O(N)$ time. Similarly, a random sample can be obtained in $O(N)$ time. The algorithm requires time $O(N \cdot t(N^r, 1/\varepsilon))$ to run the learner and then $O(N^2)$ time to compute $\text{dist}_{\mathcal{D}^{\otimes r}}(T, \ell\text{-IsEdge}_G^{\oplus r})$. This implies an overall runtime of $O(N \cdot t(N^r, 1/\varepsilon)) = O(rn^2 \cdot t(n^{2r}, 1/\varepsilon))$.

Correctness. If we let $\delta := A^{1/r} - 1$, then the assumption of the theorem statement is that $\delta' > (\delta + 8\varepsilon^{1/r})d + \delta$. Therefore, we are able to apply Theorem 30 from which we deduce correctness.

In the **Yes case**, if G has a vertex cover of size at most k, then there is a decision tree of size at most s computing ℓ-IsEdge$_G^{\oplus r}$. So by the guarantees of DT-LEARN, our algorithm correctly outputs "Yes".

In the **No case**, every vertex cover of G has size at least $(1+\delta')k$. If $\varepsilon_{\text{hyp}} > \varepsilon$ then our algorithm for VERTEX COVER correctly outputs "No". Otherwise, assume that $\varepsilon_{\text{hyp}} \le \varepsilon$. Then, Theorem 30 ensures that $(1+\delta)^r s < |T_{\text{hyp}}|$ and so our algorithm correctly outputs "No" in this case as well. ■

Remark 31 (Why we require such sharp lower bounds in the proof of Theorem 20) *A key step in the analysis of the correctness of our reduction is Theorem 30. Since our upper bound for ℓ-*IsEdge *is of the form s^r, we require an equally strong lower bound of the form $(s')^r$. A weaker lower bound $(s')^{cr}$ for some $c < 1$ would be insufficient, since the parameter s would no longer separate the Yes and No cases in Theorem 30.*

Appendix G. Decision tree minimization given a subset of inputs

First, we recall the problem of decision tree minimization (Zantema and Bodlaender, 2000; Sieling, 2008).

Definition 32 (Decision tree minimization) DT-MIN(s, s') *is the following. Given a decision tree T over n variables and parameters $s, s' \in \mathbb{N}$, distinguish between*

- ***Yes case**: there is a size-s decision tree T' such that $T'(x) = T(x)$ for all $x \in \{0,1\}^n$; and*
- ***No case**: all decision trees T' such that $T'(x) = T(x)$ for all $x \in \{0,1\}^n$ have size at least s'.*

Sieling (2008) proves the following hardness results for DT-MIN:

Theorem 33 (Hardness of approximating DT-MIN (Sieling, 2008)) *The following hardness results hold for* DT-MIN*:*

- *for all constants $C > 1$,* DT-MIN(s, Cs) *is* NP*-hard; and*
- *for all constants $\gamma < 1$, there is no quasipolynomial time algorithm for* DT-MIN$(s, 2^{(\log s)^\gamma} \cdot s)$ *unless* $\mathrm{NP} \subseteq \mathrm{DTIME}(n^{\mathrm{polylog}(n)})$

We observe that our proof of Theorem 2 recovers Theorem 33 and also strengthens the hardness results to hold even when the no case in Theorem 32 is strengthened to: there is an explicit set of inputs D and an explicit distribution $\mathcal{D}$ over D such that any decision tree T' which agrees with T with probability $1 - \varepsilon$ for $\boldsymbol{x} \sim \mathcal{D}$ has size at least s'. This is a strict strengthening since any decision tree T' such that $T'(x) = T(x)$ for all $x \in \{0,1\}^n$ also agrees with T over the distribution $\mathcal{D}$.

Theorem 34 (Hardness of approximating DT-DATASET-MIN) *Let* DT-DATASET-MIN(s, s') *be the variant of* DT-MIN(s, s') *where the input includes a subset of inputs $D \subseteq \{0,1\}^n$, the pmf of a distribution $\mathcal{D}$ over D, and a parameter ε, and the No case is changed to "all decision trees T' such that $T'(\boldsymbol{x}) = T(\boldsymbol{x})$ with probability $1 - \varepsilon$ for $\boldsymbol{x} \sim \mathcal{D}$ have size at least s'." Then the following hardness results hold*

- *for all constants $C > 1$ there is a constant $\varepsilon > 0$ such that* DT-DATASET-MIN(s, Cs) *with error parameter ε is* NP*-hard; and*
- *for all constants $\gamma < 1$, there is a parameter $\varepsilon = 2^{-(\log s)^\gamma}$ such that there is no quasipolynomial time algorithm for* DT-MIN$(s, 2^{(\log s)^\gamma} \cdot s)$ *with error parameter ε unless* $\mathrm{NP} \subseteq \mathrm{DTIME}(n^{\mathrm{polylog}(n)})$

Proof These hardness results follow from the reduction in Theorem 20. Specifically, we construct a decision tree $T^\star$ computing ℓ-IsEdge$^{\oplus r}$ over $\{0,1\}^{r(n\ell+n)}$. The set of all n vertices of G trivially forms a vertex cover of G. Therefore, we can apply Theorem 14 to obtain a decision tree for ℓ-IsEdge of size $(\ell+1)(n+m)+mn$. We can stack r independent copies of this decision tree as in the proof of Theorem 26 (see Figure 4) to get a decision tree for ℓ-IsEdge$^{\oplus r}$ whose size is $[(\ell+1)(n+m)+mn]^r$. We then choose $\ell\text{-}D_G^r = \mathrm{supp}(\ell\text{-}\mathcal{D}_G^{\otimes r})$ to be the subset of inputs for the minimization instance. Moreover, it is straightforward to compute the pmf of the distribution $\ell\text{-}\mathcal{D}_G^{\otimes r}$ and provide this to the algorithm for DT-DATASET-MIN.

Therefore, as in the proof of Theorem 1, $(1+\delta')$-approximating VERTEX COVER reduces in polynomial-time to DT-DATASET-MIN(s, Cs). This completes the proof of the first point in the theorem statement.

For the second point, let $\gamma < 1$ be given. We choose r large enough so that $(1+\delta)^r > 2^{(\log s)^\gamma}$ where s and δ are parameters from Theorem 30. Since $s = O(n^{2r})$, any $r = \operatorname{polylog} n$ satisfying $r^{1-\gamma} \geq \Omega((\log n)^\gamma)$ is sufficient. For this choice of r, our reduction runs in quasipolynomial-time and reduces $(1+\delta')$-approximating VERTEX COVER to DT-DATASET-MIN$(s, 2^{(\log s)^\gamma} \cdot s)$. Therefore, the proof is complete. ■