THE COMPLEXITY OF LEARNING DECISION TREES

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Caleb Koch
December 2025

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Li-Yang Tan) Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Omer Reingold)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Aviad Rubinstein)

Approved for the Stanford University Committee on Graduate Studies

Abstract

Decision trees are one of the simplest and most widely used models in machine learning. Over the last four decades, a rich line of research has emerged on algorithms for learning decision trees from data. The fastest known algorithm for this task runs in quasipolynomial-time and a central open question has been whether this runtime can be improved.

This thesis provides strong conditional evidence that the quasipolynomial runtime is optimal, effectively resolving this open problem. We also investigate the impact of different modeling assumptions on the complexity of the problem. We show that decision tree learning remains difficult even in the more powerful query learning model, resolving another foundational question in learning theory. Finally, we exhibit new lower bounds in the low-accuracy learning regime by developing a novel connection to coding theory.

Along the way, we also derive state-of-the-art learning lower bounds for other important concept classes including juntas and DNFs. Our techniques draw on a wide variety of tools from theoretical computer science including parametrized complexity, Fourier analysis, hardness amplification, coding theory, and XOR lemmas.

Acknowledgments

I had an incredibly enriching five years at the Stanford theory group. This thesis wouldn't have been the same without the stimulating research environment there and all the people who I learned so much from.

My advisor, Li-Yang Tan, took me on as a very inexperienced first-year and quickly became an indispensable resource – in research and beyond. He showed me the importance of asking the right questions and choosing the right problems. Seeing him push an idea to its limits always amazed me (and still does). He was also an endless source of wisdom and insight about navigating grad school.

Early on, Li-Yang brought me into a project with Guy Blanc and Jane Lange, a dangerous duo with the ability to crush any and all research questions. They became frequent collaborators and a constant source of inspiration. Later, Carmen joined the group and, with her natural aptitude for research, quickly turned into an integral part of it. We had a very fruitful collaboration on decision tree learning lower bounds. All of the papers forming this thesis were joint with Carmen and Li-Yang. I also had the pleasure of working with Lorenzo Beretta, Nathan Harms, and Alex Hayderi. My collaboration with Lorenzo and Nathan grew out of a memorable summer at the Simons Institute for the Theory of Computing. Lorenzo's unbounded energy and curiosity fueled many research sessions, and Nathan's focus and drive pushed our research to its limits.

I am greatly indebted to my reading committee: Omer Reingold, Aviad Rubinstein, and Li-Yang. Omer and Aviad also graciously served on my quals committee and welcomed me on as a first-year rotation student. I am very grateful for their helpful feedback and input throughout my PhD. Likewise, I was fortunate to have a wonderful oral exam committee: Omer, Aviad, Tselil Schramm, Li-Yang, and Ellen Vitercik.

Finally, I'd like to thank my family. They have always supported my pursuits, and I never would have made it here without them.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

This thesis is about the machine learning problem of constructing decision tree representations of data. Decision trees are both a fundamental class that has appeared in many different areas of study and an important part of many machine learning workflows. Understanding how to construct them from data is therefore both intellectually natural and practically important.

Decision trees have long been recognized as one of the most powerful abstractions for organizing information. Already in the third century, the Greeks were using decision trees to synthesize information. One of the earliest examples was the Porphyrian tree which was used to model Aristotle's theory of categories [Bar03]. Other examples of decision trees throughout history include Ramon Llull's *Arbor Scientiae* for organizing different branches of scientific knowledge [Llu95], the Linnaean tree for biological taxonomy [Lin35], and genealogy trees for organizing ancestry information. Their visual appeal and simplicity, conciseness, and versatility have enabled their success.

More recently, decision trees have become an important machine learning model. There are many reasons for the abundance of decision tree models in ML. They handle sparse data with little preprocessing, their behavior is readily interpretable (each root-to-leaf path spells out an explanation), evaluation time scales with depth rather than total size, and they pair well with ensembling methods. With all this, it is not surprising that algorithms for learning decision trees have been studied for several decades. Two dominant one have emerged from this research: CART, introduced by Breiman, Friedman, Olshen, and Stone in 1984 [BFSO84] and ID3 (later C4.5) introduced by Quinlan in 1986 [Qui86]. These two decision tree learning heuristics have been cited as among the top 10 most influential data mining algorithms in history [WKRQ+08] and have inspired many subsequent research works. Beyond research, these heuristics have formed the basis of popular ML production systems. For example, Apache Spark's *mllib* decision tree, random forest, and gradient boosted tree implementations follow the CART framework for decision splits [MBY+16]. The same

CART-style trees are the base learners in random forests, a commonly used model in scikit-learn and countless data science workflows [Bre01, PVG+11]. The successor of ID3, C4.5, is shipped as J48 in Weka, one of the most widely taught data mining toolkits [HDW94].

Despite their popularity, ID3 and CART are far from perfect. One of the main downsides is that they are heuristics and lack formal guarantees. This makes it impossible to answer basic questions like how optimal is the learned decision tree and how optimal is the runtime of the learning procedure? It also leads to issues like *instability* where small perturbations to the training set can cause the algorithms to produce vastly different trees [Bre96], along with problems generalizing beyond their training sets. Quoting the scikit-learn user manual: "Decision-tree learners can create overcomplex trees that do not generalize the data well...[and] can be unstable because small variations in the data might result in completely different trees being generated" [PVG+11]. This motivates the study of *rigorous foundations* for decision tree learning. What algorithms can we hope to develop that have formal guarantees? This forms the basis for the main question of this thesis:

Question 1 (Our guiding question)**.** What is the *complexity* of learning decision trees?

The word "complexity" in this question is meant to signify two important aspects of our study. First, we are interested in the *runtime* complexity of the problem. This requires little motivation: in practice, the speed of algorithm largely determines how useful it is. Second, we are interested in rigorous, quantitative bounds. Answering the question rigorously has the benefit of guiding and informing empirical research on new algorithms, but also contributes to the much larger goal of building formal foundations for fundamental machine learning problems.

The focus of our thesis: lower bounds. The first step to answering Question 1 is to write down an algorithm for learning decision trees and show (formally) that it works well. The second step is to the prove a lower bound stating that you can't write down a better algorithm. As we will see, there is a large body of work on algorithms for learning decision trees, but far less work on lower bounds. For this reason, our focus is on proving new lower bounds. We suspect the scarcity of such lower bounds reflects a broader lack of systematic techniques for computational lower bounds in learning theory. A secondary aim of our thesis is therefore methodological – developing tools that may extend beyond decision trees to other problems in learning theory.

Circling back to existing heuristics for decision tree learning, our lower bounds help explain why there has been little progress on formally accounting for their empirical success. Quoting [KM99], "it seems fair to say that, despite their other successes, the models of computational learning theory have not yet provided significant insight into the apparent empirical success of programs like C4.5 and CART". Our results suggest a reason: efficient algorithms with strong learning guarantees likely do not exist. CART and C4.5 are best understood as heuristics that – precisely because of our lower bounds – may never have provably optimal, efficiently learnable counterparts.

Decision tree learning in the PAC model: properness and distributional assumptions. We formalize decision tree learning using the PAC model of Valiant [Val84]. Two modeling choices are central for both our lower bounds and their connection to practice: *properness* and *distribution-freeness*. Properness means the learner must output a decision tree *hypothesis* rather than a some other representation. This is the standard setting in machine learning pedagogy and practice (see e.g. the textbooks [Mit97, Bis06, SSBD14] and the "Decision tree learning" entry on Wikipedia). From a practical perspective, properness of decision tree algorithms is not just a feature but the entire point—to produce a decision tree representation of the data. Distribution-freeness requires the learner to succeed for *every* distribution over inputs. This matches the reality that data from nature seldom admits a convenient structure. Also, distribution-free lower bounds provide an important baseline in the most general learning setting against which more refined, distribution-specific results can later be understood.

1.2 Overview of results

Chapter 2: The complexity of learning decision trees in the PAC model. In this chapter, we investigate the complexity of learning decision trees in the most general setting. The de facto model for capturing learning is the PAC model of Valiant [Val84]. Here we make no assumptions on the distribution over inputs and assume that the learner's only access to the target is through random examples. See Definition 1 for a formal definition of the model. Our main result is a conditional lower bound on the runtime for any learning algorithm for this problem.

Main result of chapter 2 (informal)

Unless the parametrized set cover problem can be efficiently approximated, there is no algorithm running in time $n^{o(\log n)}$ for properly learning decision trees in the distribution-free PAC model.

The best known algorithm for properly learning decision trees in the distribution-free PAC model is due to Ehrenfeucht and Haussler [EH89] and runs in time $n^{O(\log n)}$. Therefore, our result shows that this algorithm is essentially optimal, assuming hardness of the parametrized set cover problem.[1] The main ingredient for this theorem is a generalization of a reduction due to [ABF+09] from the set cover problem to learning decision trees (which itself built on the works of [PV88, Ang]). Using our techniques, we also derive the strongest lower bounds to date for learning juntas and DNFs. We show that learning k-juntas requires time $n^{\Omega(k)}$ under ETH and that learning polynomial-size DNFs requires time $n^{\Omega(\log n)}$.

This chapter is based on the following publication:

[1] This assumption has been studied before and is the subject of ongoing research. See Section 2.2.2 for details.

[KST23b] Caleb Koch, Carmen Strassle, and Li-Yang Tan. Superpolynomial lower bounds for decision tree learning and testing. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1962–1994. SIAM, 2023

Chapter 3: The complexity of learning decision trees with queries. In Chapter 2, we investigated the complexity of learning decision trees in the standard PAC model. In this model, the learning algorithm can only access information about the target decision tree *passively* through random labeled examples. In the PAC model with queries, the learner additionally can query the target decision tree on any input. This stronger and more *active* model of learning captures the task of converting an existing high-accuracy hypothesis, such as a neural network, into a simpler representation such as a decision tree. The PAC model with queries was already present in Valiant's original paper [Val84] as a way of capturing interactions with an expert. Since its introduction, the PAC model with queries has been extensively studied. In this chapter, we give the first lower bound for learning decision trees in the PAC model with queries, thereby answering an important open problem in learning theory [Bsh93, GLR99, MR02, Fel16].

Main result of chapter 3 (informal)

Properly learning decision trees optimally in the distribution-free PAC model with queries is NP-hard.

An important word in statement of our result is "optimally". This means that the size of the decision tree output by the learner is minimal. This is in contrast with the setting of our result in Chapter 2 where no restriction is placed on the size of the decision tree output by the learner. In the more stringent setting of this this chapter, the fastest known algorithm is via dynamic programming and runs in time $2^{O(n)}$ (see e.g. [GLR99, MR02]). Assuming SAT takes exponential time, our reduction yields a near-matching $2^{n^{\Omega(1)}}$ lower bound. In a follow-up work, we show that the problem remains NP-hard even if the learner is allowed to output a hypothesis whose size is within a constant-factor of the optimal size.

En route to proving the main theorem of this chapter, we derive new, stronger lower bounds for a decision problem, decision tree minimization. We prove hardness for a "dataset" version of the problem which implies hardness for the original problem. Our lower bound recovers the lower bounds in [Sie08, ZB00] via an arguably much simpler proof.

Our follow-up work builds a new, general XOR lemma for decision trees which may be of independent interest.

This chapter is based on the following two publications:

[KST23a] Caleb Koch, Carmen Strassle, and Li-Yang Tan. Properly learning decision trees with queries is NP-hard . In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 2383–2407. IEEE Computer Society, 2023

[KST24b] Caleb Koch, Carmen Strassle, and Li-Yang Tan. Superconstant inapproximability of decision tree learning. In Shipra Agrawal and Aaron Roth, editors, *Proceedings of Thirty Seventh Conference on Learning Theory (COLT)*, volume 247 of *Proceedings of Machine Learning Research*, pages 2979–3010. PMLR, 2024

Chapter 4: The complexity of weakly learning decision trees. The lower bounds in the previous chapters crucially rely on the learner outputting a high-accuracy hypothesis. This learning regime is known as *strong learning*. The *weak learning* regime, in contrast, only requires the learner to output a hypothesis achieving slightly nontrivial accuracy. Lower bounds for weak learning based on worst-case complexity assumptions are extremely rare[2] and previously no lower bounds were known for weakly learning decision trees. In this chapter, we develop a new connection between weakly learning decision trees and the *nearest codeword problem* (NCP) and use this connection to give the first lower bounds for the former task.

Main result of chapter 4 (informal)

Unless FPT $= W[1]$*, there is no polynomial-time algorithm for properly learning decision trees to accuracy* $\frac{1}{2} + \frac{1}{\mathrm{poly}(n)}$ *in the distribution-free PAC model.*

Our reduction from NCP to weakly learning decision trees is fairly generic and enables us to derive stronger lower bounds stronger hardness assumptions such as ETH and Gap-ETH. En route to our result, we derive new lower bounds for the problem of testing juntas. We show, in the distribution-free model, that the problem of distinguishing whether a function is 0-close to a k-junta or 1/2-far from $k \cdot n^{\Omega(1/\log\log n)}$-juntas is NP-hard.

This chapter is based on the following publication:

[KST24a] Caleb Koch, Carmen Strassle, and Li-Yang Tan. Fast Decision Tree Learning Solves Hard Coding-Theoretic Problems . In *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1893–1910. IEEE Computer Society, 2024

1.3 What is not included?

I co-authored several papers during my PhD which have been excluded from this thesis. This includes three papers bringing theoretical CS tools to bear on problems in explainable machine learning [BKLT22a, BKLT22b, BKL+23]. I also worked on hardness amplification and how it relates to learning and testing problems [BKST23, BKST24, BHKT24]. More recently, I've worked on computational-statistic tradeoffs in learning [BKST25] and the problem of testing juntas with samples [BHK25]. These papers have been excluded since they do not fit within the theme of decision

[2] In fact, we are only aware of one such lower bound due to [Hir22] for weakly learning programs.

tree learning. That being said, all of these papers fall within the broader theme of using complexity theoretic tools and ideas to study fundamental problems in machine learning and may be of interest to readers of this thesis.

1.4 Preliminaries

1.4.1 Basics and probability

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. For $R \subseteq [n]$, we write $x_R \in \{0,1\}^{|R|}$ to denote the substring of x on the coordinates in R.

Distributions. We use boldface letters e.g. $\boldsymbol{x}, \boldsymbol{y}$ to denote random variables. For a distribution $\mathcal{D}$, we write $\mathrm{dist}_{\mathcal{D}}(f, g) = \Pr_{\boldsymbol{x}\sim\mathcal{D}}[f(\boldsymbol{x}) \neq g(\boldsymbol{x})]$. A function f is ε-close to g under $\mathcal{D}$ if $\mathrm{dist}_{\mathcal{D}}(f, g) \leq \varepsilon$. Similarly, f is ε-far from g under $\mathcal{D}$ if $\mathrm{dist}_{\mathcal{D}}(f, g) \geq \varepsilon$. If f is 0-close under $\mathcal{D}$ to some g having property $\mathcal{P}$, then we say that f has property $\mathcal{P}$ under $\mathcal{D}$. For a set S, we write $\mathrm{Unif}(S)$ to denote the uniform distribution over that set. A generator for a distribution $\mathcal{D}$ over $\{0,1\}^n$ is an algorithm $G : \{0,1\}^n \to \{0,1\}^n$ which takes n uniform random bits as input and outputs n bits distributed according to $\mathcal{D}$: $\Pr_{\boldsymbol{x}\sim\mathrm{Unif}(\{0,1\}^n)}[G(\boldsymbol{x}) = x] = \Pr_{\boldsymbol{x}\sim\mathcal{D}}[\boldsymbol{x} = x]$ for all $x \in \{0,1\}^n$. The support of the distribution is the set of elements with nonzero mass and is denoted $\mathrm{supp}(\mathcal{D})$.

1.4.2 Decision trees

A decision tree is a binary tree used to represent a Boolean function $f : \{0,1\}^n \to \{0,1\}$. Each internal node of the tree is labeled with an input but x_i for some $i \in [n]$. Each leaf is labeled with a bit value 0 or 1. A decision tree can be evaluated on an input $x \in \{0,1\}^n$ by following the root-to-leaf path consistent with x: the evaluation starts at the root node and proceeds to the left/right branch out of node x_i depending on the value of the ith bit of the input x. The output of the decision tree on x is the label of the leaf at the end of the path consistent with x. See Figure 1.1 as an example of a decision tree and the evaluation of an input on a decision tree.

For a decision tree $T : \{0,1\}^n \to \{0,1\}$, we write $L \in T$ to denote that L is a leaf of T. The size of T is its number of leaves and is denoted $|T|$. For an input $x \in \{0,1\}^n$, we write $\mathrm{depth}_T(x) \in \mathbb{N}$ to denote the *depth* of x in T, the number of variables queried on the root-to-leaf path consistent with x. The depth of a decision tree is the length of the longest root-to-leaf path, i.e. $\max_{x\in\{0,1\}^n}\{\mathrm{depth}_T(x)\}$. Every Boolean function is computed by some decision tree and we write $\mathrm{DT}(f)$ to denote the smallest size of decision tree computing f.

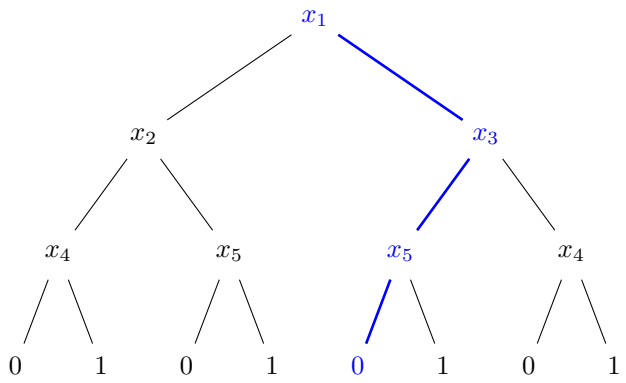


Figure 1.1: Evaluating an input on a tree T representing a function $f : \{0,1\}^5 \to \{0,1\}$. The input is $x^\star = 10000$ and its computation path through the tree is colored in blue. The decision tree leaf value here is 0 indicating that $T(x^\star) = f(x^\star) = 0$.

1.4.3 Other concept classes

Circuits. We consider Boolean circuits $\mathcal{C} : \{0,1\}^n \to \{0,1\}$ with AND, OR, NOT, and PARITY gates: $\{\wedge, \vee, \neg, \oplus\}$. The size of a circuit $|\mathcal{C}|$ is the number of gates in it. The depth of a circuit is the longest directed path from an input node to an output node.

DNF formulas. A literal is a variable or its negation. A term is a conjunction ($\wedge$) of literals. A disjunctive normal form (DNF) formula $F : \{0,1\}^n \to \{0,1\}$ is a disjunction ($\vee$) of terms, denoted $F = t_1 \vee \cdots \vee t_s$. The size of the DNF formula is $|F| = s$, the number of terms. The *width* of a term $|t_i|$ is the number of literals in it. The width of an input $x \in \{0,1\}^n$ is defined as the width of the smallest width term accepting x and 0 if no term accepts x:

$$\text{width}_F(x) := \begin{cases} \min\limits_{t_i(x)=1} |t_i| & F(x) = 1 \\ 0 & F(x) = 0. \end{cases}$$

The dual of a DNF formula is a conjunctive normal form (CNF) formula and is a conjunction of *clauses*. Each clause is a disjunction of literals.

Juntas. A function $f : \{0,1\}^n \to \{0,1\}$ is a k-junta if its output depends on $\leq k$ bits. Hence, if f is a k-junta it can be completely specified by a table of size 2^k corresponding to all possible assignments to the k relevant variables. In particular, every k-junta is a size-2^k decision tree and every size-s decision tree is an s-junta.

Parities. A function $f : \{0,1\}^n \to \{0,1\}$ is a k-parity if its output can be written as the XOR of k of its input bits: there is a fixed set of indices $i_1, \ldots, i_k \in [n]$ such that $f(x) = x_{i_1} \oplus \cdots \oplus x_{i_k}$ for all $x \in \{0,1\}^n$. If $S \subseteq [n]$ is a set of indices, we use $\chi_S : \{0,1\}^n \to \{0,1\}$ to denote the n-bit Boolean function which is the parity of the bits specified by S.

Relationship to decision trees. We recall the set of inclusions

$$\{ \text{ } k\text{-parities } \} \subseteq \{ \text{ } k\text{-juntas } \} \subseteq \{ \text{ depth-}k \text{ decision trees } \} \subseteq$$
$$\{ \text{ size-}2^k \text{ decision trees } \} \subseteq \{ \text{ size-}2^k \text{ DNFs } \} \subseteq \{ \text{ size-}2^k \text{ circuits } \}$$

with each class being strictly more expressive than the previous one. See Figure 1.2 for an illustration of the main inclusions that appear in this thesis.

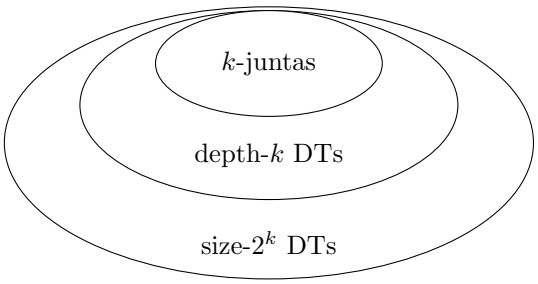


Figure 1.2: Illustration of inclusions of basic function classes

1.4.4 Learning

In the realizable PAC learning model [Val84], 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 $\varepsilon \in (0,1)$ 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 algorithm 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 *membership queries* is given oracle access to the target function f along with the example oracle $\mathrm{EX}(f, \mathcal{D})$. An learning algorithm that succeeds over any distribution is called *distribution-free*.

Formally, we use the following definition for PAC learning decision trees.

Definition 1 (PAC learning decision trees)**.** *Let $\mathcal{T} = \{T : \{0,1\}^n \to \{0,1\} \mid T \text{ is a decision tree}\}$ be the class of decision trees over a fixed domain $\{0,1\}^n$. A distribution-free learning algorithm $\mathcal{L}$*

learns $\mathcal{T}$ *in time* $t(n, s, \varepsilon)$ *if for all distributions* $\mathcal{D}$ *and for all* $T \in \mathcal{T}, \varepsilon \in (0,1)$, $\mathcal{L}$ *with oracle access to* $\mathrm{EX}(T, \mathcal{D})$ *runs in time* $t(n, |T|, \varepsilon)$ *and with probability* $2/3$ *outputs* $h : \{0,1\}^n \to \{0,1\}$ *such that* $\mathrm{dist}_{\mathcal{D}}(T, h) \leq \varepsilon$. *Furthermore,* $\mathcal{L}$ *is proper if* $h \in \mathcal{T}$.

All learning algorithms we consider are *proper* learning algorithms. When referring to "learning decision trees" we mean properly learning the concept class $\mathcal{T} = \{T : \{0,1\}^n \to \{0,1\} \mid T \text{ is a decision tree}\}$. Likewise, when referring to "learning size-s decision trees", we mean properly learning the concept class $\mathcal{T}_s = \{T : \{0,1\}^n \to \{0,1\} \mid T \text{ is a size-}s \text{ decision tree}\}$. When discussing algorithms for learning k-juntas, we assume the output of the learning algorithm is a table of size 2^k (as in e.g. [MOS04]).

Property testing and its relationship to learning. The problem of testing a concept class is closely to the problem of learning it [GGR98]. The setup for testing is the same as learning with membership queries: there is an unknown distribution $\mathcal{D}$ and some unknown *target* function $f \in \mathcal{C}$ from a fixed concept class $\mathcal{C}$. The testing algorithm takes as input $\varepsilon \in (0,1)$ and has oracle access to an example oracle $\mathrm{EX}(f, \mathcal{D})$ and to f. Then, the testing algorithm should, with high probability, output "yes" if $f \in \mathcal{C}$ and "no" if f is ε-far from $\mathcal{C}$ under $\mathcal{D}$, i.e. $\mathrm{dist}_{\mathcal{D}}(f, \mathcal{C}) \geq \varepsilon$. The algorithm's behavior is undefined in the regime where $f \notin \mathcal{C}$ and $\mathrm{dist}_{\mathcal{D}}(f, \mathcal{C}) < \varepsilon$. In the *tolerant* testing regime, the testing algorithm is given two distance parameters, $\varepsilon_1, \varepsilon_2 \in (0,1)$, and the problem is to distinguish whether f is ε_1-close to $\mathcal{C}$ or ε_2-far from $\mathcal{C}$. It is straightforward to show that learning with random examples implies tolerant testing and learning with membership queries implies non-tolerant learning.

Fact 1.4.1 (Learning implies testing)**.** *If* $\mathcal{A}$ *is an algorithm running time* t *for learning a class* $\mathcal{C}$ *under* $\mathcal{D}$ *to error* ε*, then there is a tolerant testing algorithm for* $\varepsilon_1 = 0$ *and* $\varepsilon_2 = \varepsilon$ *running in time* $\tilde{O}(t/\varepsilon^2)$ *for* $\mathcal{C}$ *under* $\mathcal{D}$*. Furthermore, if* $\mathcal{A}$ *requires membership queries to learn* $\mathcal{C}$*, then there is a non-tolerant tester with distance parameter* ε *for* $\mathcal{C}$ *under* $\mathcal{D}$ *running in time* $\tilde{O}(t/\varepsilon^2)$.

Testing lower bounds immediately yield learning lower bounds via Fact 1.4.1 and are therefore more difficult to prove. All of the learning lower bounds in this thesis are in fact derived from stronger *testing* lower bounds. Specifically, our lower bounds for learning to error ε with random samples are derived from lower bounds for tolerantly testing with distance parameters $\varepsilon_1 = 0$ and $\varepsilon_2 = \varepsilon$ and our lower bounds for learning with membership queries are derived from lower bounds for non-tolerant testing with distance parameter ε.

1.4.5 Computational hardness assumptions

All of our lower bounds are conditional, meaning they rely on the assumption that there exist problems that are computationally hard to solve. Researchers have formulated several different

hardness assumptions of varying strength. We review the main hardness assumptions we use in our results and how they relate to each other.

Hypothesis 1 (Exponential time hypothesis (ETH) [Tov84a, IP01, IPZ01])**.** *There exists a constant* $\delta > 0$ *such that* 3-SAT *on* n *variables cannot be solved in* $O(2^{\delta n})$ *time.*

Since we prove hardness against randomized algorithms, we use a randomized variant of ETH.

Hypothesis 2 (Randomized ETH, see [CIKP08, DHM+08])**.** *There exists a constant* $\delta > 0$ *such that* 3-SAT *on* n *variables cannot be solved by a randomized algorithm in* $O(2^{\delta n})$ *time with error probability at most* $1/3$.

We will also use two additional hypotheses.

Hypothesis 3 (Strong exponential time hypothesis (SETH) [IP01, IPZ01])**.** *For every* $\delta > 0$*, there exists a* $k \in \mathbb{N}$ *such that* k-CNF-SAT *on* n *variables cannot be solved in time* $O(2^{n(1-\delta)})$.

Hypothesis 4 ($W[1] \neq$ FPT, see [DF13, CFK+15a])**.** *For any computable function* $f : \mathbb{N} \to \mathbb{N}$*, no algorithm can decide if a graph* $G = (V, E)$ *contains a* k*-clique in* $f(k) \cdot \text{poly}(|V|)$ *time.*

As with randomized ETH, randomized SETH and randomized $W[1] \neq$ FPT are the respective versions of these hypotheses against randomized algorithms. Also, we remark that $W[1] \neq$ FPT is a weaker assumption than ETH which itself is weaker than SETH.

Chapter 2

The complexity of learning decision trees in the PAC model

2.1 Introduction

A classic result of Ehrenfeucht and Haussler [EH89] gives a quasipolynomial time algorithm for properly PAC learning decision trees: Given labeled examples $(\boldsymbol{x}, f(\boldsymbol{x}))$ where $f : \{0,1\}^n \to \{0,1\}$ can be computed by a size-s decision tree and $\boldsymbol{x}$ is drawn from a distribution $\mathcal{D}$ over $\{0,1\}^n$, their algorithm runs in $n^{O(\log s)}$ time and returns a decision tree hypothesis that is close to f under $\mathcal{D}$. Numerous alternative algorithms have since been designed within restricted variants of the PAC model (e.g. where $\mathcal{D}$ is assumed to be uniform) and by relaxing the problem (e.g. allowing hypotheses that are not themselves decision trees[1]) [Riv87, Blu92, Han93, KM12, KM96, Bsh93, GLR99, BM02, MR02, JS05, KS06, OS07, GKK08, KST09, HKY18, CM19, BLQT22], but Ehrenfeucht and Haussler's algorithm remains state of the art in the standard PAC model.

A related setting is when an explicit representation of the function f, and possibly also the distribution $\mathcal{D}$, are given to the algorithm. This easier setting, where the algorithm can "inspect" f, models a popular approach in explainable machine learning known as post-hoc explanations. The goal here is not to train a decision tree model for an unknown function f, but instead to turn a complicated trained model f (e.g. a neural net) into its decision tree representation. While numerous algorithms for this task have been proposed in the empirical literature [CS95, BS96, VAB07, ZH16, BKB17, VLJ+17, FH17, VS20], among those with theoretical guarantees, the fastest one remains that of Ehrenfeucht and Haussler.

[1] Such improper decision tree learning algorithms do not apply to the problem of "decision tree learning" as is meant in the context of machine learning, where it always refers to the problem of constructing decision tree *hypotheses*. See e.g. the textbooks [Mit97, Bis06, SSBD14] or the Wikipedia page for "Decision tree learning". From a practical perspective, properness of decision tree algorithms is not just a feature but the entire point—to produce a decision tree representation of the data. The focus of this paper will be on proper decision tree learning algorithms.

In parallel with these algorithmic works, there has emerged a long line of research on the *hardness* of decision tree learning [HR76, GJ79, BFJ+94, HJLT96, KPB99, ZB00, LN04, CPR+07, RRV07, Sie08, ABF+09, AH12, Rav13, BLT20]. It is interesting to note that the earliest paper here, by Hyafil and Rivest in 1976, predates Ehrenfeucht and Haussler's algorithm by more than a decade; indeed, it even predates the PAC model. Their paper, which established the NP-completeness of a certain formulation of decision tree learning with perfect accuracy, reveals that the problem was already intensively studied and recognized as central in the 1970s. Quoting the authors, "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".

2.2 Our results

We establish new hardness results for distribution-free learning of decision trees. All of our lower bounds hold even when explicit representations of both the function f and distribution $\mathcal{D}$ are given to the algorithm; lower bounds in this setting imply lower bounds for learning and testing.

We obtain our results within a unified framework that builds on an active line of research on the inapproximability of Set-Cover [LY94, Fei98, CHKX06, DS14, Mos15, KLM18, CL19, Lin19, CHK20, KI21]. Connections between Set-Cover and decision tree optimization problems, both in terms of algorithms and hardness, date back to [HR76] and are present in numerous prior works; we leverage recent progress in both the parameterized and nonparameterized settings. All our lower bounds, being computational in nature, are conditioned on the randomized Exponential Time Hypothesis (ETH).

We now give a detailed overview of our results, in tandem with a discussion of how they compare with prior work.

2.2.1 Lower bounds for DT-Construction

The DT-Construction problem is the variant of decision tree learning where f and $\mathcal{D}$ are both given to the algorithm:

> DT-Construction(s, ε): Given as input a circuit representation of a function $f : \{0,1\}^n \to \{0,1\}$, a generator for a distribution $\mathcal{D}$ over $\{0,1\}^n$, parameters $s \in \mathbb{N}$ and $\varepsilon \in (0,1)$, and the promise that f is a size-s decision tree under $\mathcal{D}$, construct a decision tree T that is ε-close to f under $\mathcal{D}$.

Our first result is a superpolynomial runtime lower bound for DT Construction:

Theorem 1. *Under randomized ETH, for $s = n$ and $\varepsilon = \frac{1}{n}$ any algorithm for* DT-Construction(s, ε) *must take $n^{\tilde{\Omega}(\log\log s)}$ time.*

Prior works also focused on the parameter settings $s = n$ and $\varepsilon = \frac{1}{n}$, corresponding to strong learning of linear-size decision trees. Most recently, Alekhnovich, Braverman, Feldman, Klivans, and Pitassi [ABF+09] ruled out poly(n) time algorithms under the assumption that Sat cannot be solved in randomized subexponential time. Before that, Hancock, Jiang, Li, and Tromp [HJLT96] ruled out poly(n) time algorithms that return a decision tree hypothesis of size $n^{1+o(1)}$, under the assumption that Sat cannot be solved in randomized quasipolynomial time.

Our proof of Theorem 1 opens up a concrete route towards obtaining the optimal $n^{\Omega(\log s)}$ lower bound. We can also show an $n^{\Omega(\log s)}$ lower bound for the stricter version of DT-Construction where the algorithm has to return a decision tree of size s (instead of one of any size). We elaborate on both of these in Section 2.2.2.

Hardness of learning juntas with DNF hypotheses. We obtain Theorem 1 as a corollary of our first main result, which simultaneously allows for a stronger promise on the simplicity of the target function f and for the algorithm to return a more expressive hypothesis:

Theorem 2. *Under randomized ETH, for $s = n$ and $\varepsilon = \frac{1}{n}$ any algorithm for* DT-Construction(s, ε) *must take $n^{\tilde{\Omega}(\log\log s)}$ time, even if f is further promised to be a $(\log s)$-junta under $\mathcal{D}$ and the algorithm is allowed to return a DNF hypothesis.*

We recall the strict inclusions

$$\{(\log s)\text{-juntas}\} \subset \{\text{size-}s \text{ decision trees}\} \subset \{\text{size-}s \text{ DNFs}\}.$$

Each class is exponentially more expressive than the previous one: a size-s decision tree can depend on as many as s variables, and a size-s DNF can require a decision tree of size $2^{\Omega(s)}$.

The results of [ABF+09, HJLT96] are not known to be amenable to such a strengthening. [ABF+09] did give lower bounds for DNF-Construction, the analogue of DT-Construction where the target f is promised to be a DNF under $\mathcal{D}$ and the algorithm is expected to construct a DNF hypothesis. They ruled out poly(n) time algorithms for $s = n$ and $\varepsilon = \frac{1}{n}$. [ABF+09] gave two separate proofs of hardness for DT-Construction and DNF-Construction, reducing from Set-Cover for the former and from Chromatic-Number for the latter. Theorem 2, on the other hand, yields new lower bounds for both problems via a single proof. Furthermore, since the reduction from Chromatic-Number to learning DNFs in [ABF+09] does not yield a small junta target (indeed not even a small decision tree target), Theorem 2 is the first lower bound for learning small junta targets with DNF hypotheses.

In terms of upper bounds for learning DNFs, Valiant observed in his foundational PAC learning paper [Val84] that size-s DNFs can be learned in time $n^{O(s)}$. This is efficient when s is small, but

quickly becomes inefficiently as s approaches n. In the regime $s = n$, the fastest algorithm is due to [ABF+09] and runs in time $n^{O(\sqrt{n \log s})}$.

Hardness of properly learning juntas. Implicit in the proofs of Theorems 1 and 2 is a tight connection between algorithms for Set-Cover and algorithms for properly learning juntas. By making this connection explicit, we obtain strong lower bounds for the latter problem that hold even under the promise that the target is a *monotone disjunction*:

Theorem 3. *Under randomized ETH, for any $k \leq n^c$ where $c < 1$ is any constant and $\varepsilon = O(\frac{1}{n})$, there is no algorithm that, given as input a circuit representation of a function $f : \{0,1\}^n \to \{0,1\}$, a generator for a distribution $\mathcal{D}$ and the promise that f is a monotone k-disjunction under $\mathcal{D}$, runs in $n^{o(k)}$ time and constructs a k-junta that is ε-close to f under $\mathcal{D}$. Under randomized SETH, we get a lower bound of $O(n^{k-\lambda})$ for any constant $\lambda > 0$.*

These lower bounds nearly match the $O(n^k/\varepsilon)$ runtime algorithm of the trivial algorithm that iterates over all possible k-junta hypotheses. Previously, [ABF+09] ruled out $\mathrm{poly}(n)$-time algorithms for $k \leq O(\log n)$.

Theorem 3 shows that juntas are strictly harder to learn in the distribution-free setting than in the uniform distribution setting. This is because [Val05] (building on the breakthrough result of [MOS04]) showed that juntas can be learned in time $n^{0.6k} \cdot \mathrm{poly}(2^k, n)$ over the uniform-distribution. Theorem 3 implies that Valiant's algorithm cannot be extended to the distribution-free setting unless randomized SETH is false. This is the first computational separation we are aware of for learning a natural concept class in the uniform distribution setting versus the distribution-free setting.

2.2.2 Towards stronger lower bounds for DT-Construction

We show two ways in which the lower bounds of Theorems 1 and 2 can be further improved to $n^{\Omega(\log s)}$. First, we consider the stricter version of DT-Construction where the algorithm has to return a size-s decision tree:

Theorem 4. *Under randomized ETH, for $s = \exp(\tilde{O}(\log \log n))$ and $\varepsilon = \frac{1}{n}$ any algorithm for* DT-Construction(s, ε) *must take $n^{\Omega(\log s)}$ time if the algorithm has to return a size-s decision tree. As in Theorem 2, this holds even if f is further promised to be a $(\log s)$-junta under $\mathcal{D}$ and the algorithm is allowed to return a size-s DNF hypothesis.*

This more stringent version of DT-Construction corresponds to the notion of *strictly proper* learning of size-s decision trees, where the algorithm has to return a hypothesis that falls within the concept class. Ehrenfeucht and Haussler's algorithm is not strictly proper. On the other hand, for size-s decision trees of depth $O(\log s)$, there is a simple dynamic programming algorithm that runs in $n^{O(\log s)}$ time and is strictly proper [GLR99, MR02]. Since every $(\log s)$-junta is a decision tree of depth $\log s$, this matches the lower bound of Theorem 4.

Finally, we show how an optimal lower bound of $n^{\Omega(\log s)}$ for the original version of DT-Construction, matching the runtime of Ehrenfeucht and Haussler's algorithm, would follow from a natural and well-studied conjecture about Set-Cover:

Conjecture 1 (Optimal inapproximability of parameterized Set-Cover)**.** *There exists constants $\alpha, \beta < 1$ such that for $k \leq N^{\alpha}$, there is no $N^{o(k)}$ time algorithm that, given a size-N set cover instance, distinguishes between:*

- Yes*: There is a set cover of size k.*
- No*: Every set cover has size at least $k \cdot (1 - \beta) \ln N$.*

There is a simple and efficient $\ln N$-approximation algorithm for Set-Cover, and various hardness results are known for the problem of achieving a better approximation ratio [LY94, Fei98, DS14, Mos15, CHK20]. Conjecture 1 states that, in the regime where k is small[2], i.e. $k \leq N^{\alpha}$ for $\alpha < 1$, this hardness carries over to the *parameterized* setting. Existing ETH-based lower bounds for parameterized Set-Cover [CHKX06, KLM18, CL19, Lin19, KI21] are evidence in favor of it, and it is plausible that Conjecture 1 can be shown to hold under ETH.[3] We show:

Theorem 5. *Under Conjecture 1, for $s = n$ and $\varepsilon = \frac{1}{n}$ any algorithm for* DT-Construction(s, ε) *must take $n^{\Omega(\log s)}$ time. As in Theorem 2, this holds even if f is further promised to be a $(\log s)$-junta under $\mathcal{D}$ and the algorithm is allowed to return a DNF hypothesis.*

Table 4.1 summarizes our results for DT-Construction and shows how they compare with the prior state of the art.

[2] See [CKW09] for a subexponential-time approximation algorithm in the regime where k is large.
[3] See [MPW19, GKMP20] for further discussions of this conjecture and its implications for proof complexity.

Table 2.1: Algorithms and lower bounds for DT-Construction. All our results are conditioned on randomized ETH, the lower bounds of Theorems 4 and 5 are optimal.

Reference	Target	Hypothesis	Time complexity
[ABF+09]	size-s DT	DT	$n^{\omega(1)}$ lower bound
[ABF+09]	size-s DNF	DNF	$n^{\omega(1)}$ lower bound
[EH89]	size-s DT	DT	$n^{O(\log s)}$ upper bound
Theorem 2	$(\log s)$-junta	DNF	$n^{\tilde{\Omega}(\log\log s)}$ lower bound
Theorem 4	$(\log s)$-junta	size-s DNF	$n^{\Omega(\log s)}$ lower bound
Theorem 5	$(\log s)$-junta	DNF	$n^{\Omega(\log s)}$ lower bound under Conjecture 1

2.3 Our techniques

The starting point of all our reductions is the parameterized version of Set-Cover. For a set cover instance $\mathcal{S}$, we write $\mathrm{opt}(\mathcal{S})$ to denote the size of the smallest set cover.

Definition 2. *The (k,k')-Set-Cover problem is the following. Given as input a set cover instance $\mathcal{S}$ and parameters $k,k' \in \mathbb{N}$, output* Yes *if $\mathrm{opt}(\mathcal{S}) \le k$ and* No *if $\mathrm{opt}(\mathcal{S}) > k'$.*

Reducing from Set-Cover to juntas vs. DNFs. Our key lemma, which is the crux of our lower bounds for both DT-Construction, is a reduction from (k,k')-Set-Cover to the problem of distinguishing small juntas from large DNF formulas, where "small" and "large" are functions of k and k' respectively:

Lemma 2.3.1. *There is an algorithm that, given a size-N instance $\mathcal{S}$ of (k,k')-Set-Cover with n sets and a parameter $\ell \le N$, runs in $\mathrm{poly}(N)$ time and outputs a circuit representation of a function $f : (\{0,1\}^\ell)^n \to \{0,1\}$ and a generator for a distribution $\mathcal{D}$ over $(\{0,1\}^\ell)^n$ satisfying:*

- *If $\mathrm{opt}(\mathcal{S}) \le k$, then f is a $k\ell$-junta under $\mathcal{D}$.*
- *If $\mathrm{opt}(\mathcal{S}) > k'$, then any DNF of size $\le \exp(O(k'\ell))$ is $\Omega(\frac{1}{N})$-far from f under $\mathcal{D}$.*

We obtain Theorems 1, 2 and 4 by combining Lemma 2.3.1 with a recent result on the ETH-hardness of (k, k')-Set-Cover for $k' = \frac{1}{2}\left(\frac{\log N}{\log\log N}\right)^{1/k}$, where N is the size of the instance [Lin19]. Similarly, we obtain Theorem 5 by combining Lemma 2.3.1 with Conjecture 1. For Theorem 3, we only need a simpler special case of Lemma 2.3.1, which we combine with the ETH- and SETH-hardness of $(k, k+1)$-Set-Cover (i.e. the hardness of solving parameterized Set-Cover *exactly*) [CHKX06, PW10].

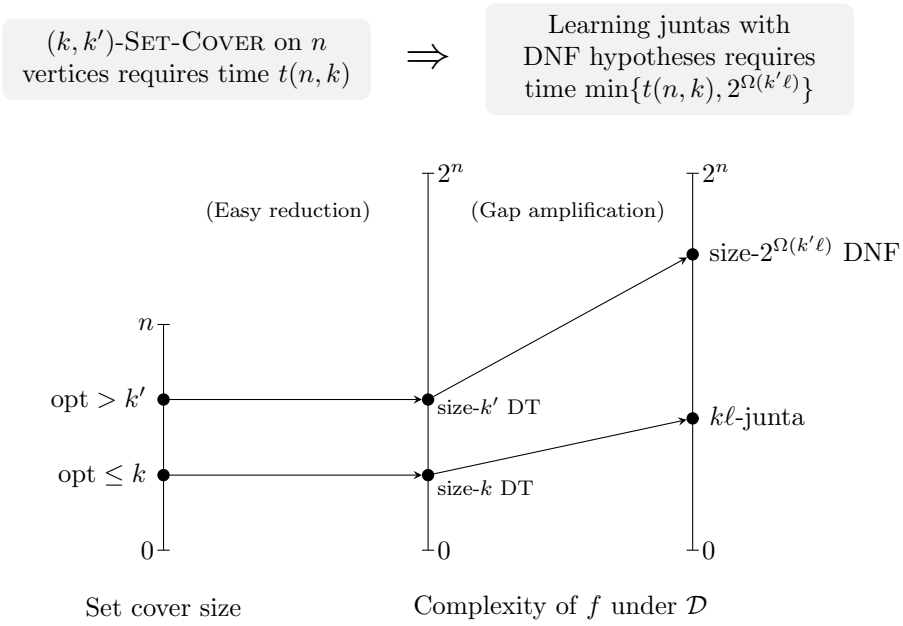


Figure 2.1: An illustration of Lemma 2.3.1 as a gap amplification technique. We take an instance of (k, k')-Set-Cover where the gap between k and k' is small and first construct a distribution and a function whose decision tree complexity under the distribution exactly reflects the set cover gap. Then we amplify the distribution and the function to obtain an even more drastic gap in the complexity of the function under the distribution. Lemma 2.3.1 is quite versatile and underlies the proofs of Theorems 1, 2, 4 and 5.

Gap amplification. We view Lemma 2.3.1 as a gap amplification procedure. Specifically, given a (k, k')-Set-Cover instance it is straightforward to construct an instance of DT-Construction, a target function f and distribution $\mathcal{D}$, where the decision tree complexity of f under $\mathcal{D}$ exactly reflects the gap (k, k'): if $\mathrm{opt}(\mathcal{S}) \leq k$ then f is a size-k decision tree under $\mathcal{D}$, and otherwise f requires decision trees of size $\geq k'$. To obtain stronger lower bounds we amplify this gap into a much larger gap in the complexity of f under $\mathcal{D}$: if $\mathrm{opt}(\mathcal{S}) \leq k$ then f is a small junta under $\mathcal{D}$, and if $\mathrm{opt}(\mathcal{S}) > k'$ then f is a large-size DNF under $\mathcal{D}$. This reduction enables us to translate lower bounds for (k, k')-Set-Cover into strong lower bounds for DT-Construction. See Figure 2.1 for an illustration of this gap amplification.

Building hard instances of DT-CONSTRUCTION**.** Our construction of f and $\mathcal{D}$ in Lemma 2.3.1 is based on the one in [ABF+09], which in turn builds on [Hau88, HJLT96]. [ABF+09] also gave a gap amplifying reduction from (k, k')-SET-COVER to the problem of distinguishing whether f has small or large decision tree complexity under $\mathcal{D}$. Lemma 2.3.1 is a strengthening of their reduction where the same gap in set cover sizes leads to a more dramatic gap in f's complexity under $\mathcal{D}$. While the construction of f and $\mathcal{D}$ is similar to the one in [ABF+09], our analysis is entirely different and is, in our opinion, simpler. Notably, our analysis enables us to obtain lower bounds even against DNF hypotheses whereas previous works relied crucially on the hypothesis being a decision tree.

2.4 Subsequent developments

Since the publication of our results, [Bsh23] has extended Theorem 1 to the setting of monotone targets, thereby showing that all of our decision tree learning lower bounds hold in the setting of learning *monotone* decision trees. There is a standard reduction from learning monotone DNFs to learning DNFs [KV08] but it is unclear whether a similar reduction holds for decision trees. So while our lower bounds here for DNF learning immediately extend to monotone DNF learning, they do not address the complexity of learning monotone decision trees. [Bsh23] fills this gap via a new construction that establishes lower bounds in the monotone decision tree setting.

2.5 Set Cover Preliminaries

Set Cover. Given a bipartite graph $\mathcal{S} = (S, U, E)$ on N-vertices, the SET-COVER problem is to find a minimum size subset $C \subseteq S$ such that every vertex in U is adjacent to some vertex in C.[4] We write $\mathrm{opt}(\mathcal{S}) \in \mathbb{N}$ to denote the size of the smallest set cover for $\mathcal{S}$. We will often write n to denote the size of $|S| \leq N$. The set of neighbors of a vertex $u \in U$ is $\mathcal{N}_{\mathcal{S}}(u) = \{s \in S : (s, u) \in E\}$. We identify a vertex $u \in U$ with its neighborhood set $\mathcal{N}_{\mathcal{S}}(u)$. Each set $\mathcal{N}_{\mathcal{S}}(u)$ can be viewed as a string in $\{0,1\}^{|S|}$ where a 1 in the string indicates an edge between u and the corresponding vertex $s \in S$. Hence, each vertex $u \in U$ can likewise be encoded as a string in $\{0,1\}^{|S|}$.[5]

Hitting Set. Given a bipartite graph $\mathcal{H} = (S, U, E)$, the HITTING-SET problem is to find a minimum size subset $I \subseteq U$ which "hits" every vertex $s \in S$: $\mathcal{N}_{\mathcal{S}}(s) \cap I \neq \varnothing$ for all $s \in S$. We write $\mathrm{opt}(\mathcal{H})$ for the size of the smallest hitting set.

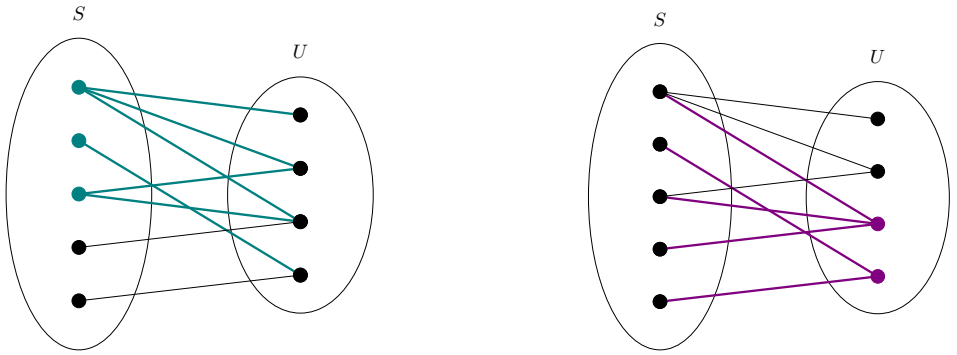
[4] Typically, the set cover problem is cast as a combinatorial problem: given subsets $S_1, \ldots, S_m \subseteq [n]$ of some universe $[n]$, find the minimum size subcollection $S_{i_1}, \ldots, S_{i_k}$ whose union is $[n]$. We consider the graph theoretic formulation because it makes the connection to the hitting set problem more transparent.

[5] We assume without loss of generality that each $\mathcal{N}_{\mathcal{S}}(u)$ is unique so that a vertex u can be identified by its neighborhood set $\mathcal{N}_{\mathcal{S}}(u)$ (if $\mathcal{N}_{\mathcal{S}}(u) = \mathcal{N}_{\mathcal{S}}(u')$ for $u \neq u'$ we can simply delete u' without affecting the set cover complexity)

An instance $\mathcal{H} = (S, U, E)$ of Hitting-Set can equivalently be viewed as an instance $\mathcal{H} = (U, S, E)$ of Set-Cover.

Fact 2.5.1 (Set-Cover and Hitting-Set are equivalent)**.** *Set-Cover and Hitting-Set are equivalent to each other under approximation-preserving reductions. In particular, any instance $\mathcal{S}$ of Set-Cover can be transformed in linear-time into an instance $\mathcal{H}$ of hitting set such that $\mathrm{opt}(\mathcal{S}) = \mathrm{opt}(\mathcal{H})$ and vice versa.*

The results of [ABF+09] are formulated in terms of hitting set. Though for consistency, in this work we will only refer to Set-Cover. See Figure 2.2 for an illustration of a set cover instance and a hitting set instance on a single bipartite graph.



(a) A set cover of size 3 for G highlighted in teal (b) A hitting set of size 2 for G highlighted in purple

Figure 2.2: A bipartite graph $G = (S, U, E)$ viewed on the left as a set cover instance and on the right as a hitting set instance.

2.5.1 Existing results on the hardness of Set-Cover

Throughout, we use several different hardness results for Set-Cover and approximating Set-Cover. We start with the following theorem due to [Lin19] about the hardness of approximating set cover. We have slightly modified the theorem from its original form to fit our setting. We discuss Lin's original theorem and our modifications in Appendix A.1.

Theorem 6 ([Lin19])**.** *Assuming randomized ETH, there is a constant $c \in (0, 1)$ such that for any $k \in \mathbb{N}$ with $k \leq \frac{1}{2} \cdot \frac{\log\log N}{\log\log\log N}$, there is no randomized N^{ck} time algorithm that can solve $\left(k, \frac{1}{2}\left(\frac{\log N}{\log\log N}\right)^{1/k}\right)$-Set-Cover on N vertices with high probability.*

We will also use results on the inapproximability of *unparameterized* Set-Cover:

Theorem 7 ([DS14, Mos15])**.** *Under randomized ETH, for every $0 < \beta < 1$, any algorithm that approximates size-N instances of Set-Cover to within $(1 - \beta) \ln N$ w.h.p. requires $2^{N^{\Omega(\beta)}}$ time.*

By a standard search-to-decision reduction, Theorem 7 implies the following lower bound for (k, k')-Set-Cover where, unlike in the parameterized setting, k is no longer guaranteed to be "small":

Theorem 8. *Under randomized ETH, for every $0 < \beta < 1$, there exists $k \leq N$ such that any algorithm that solves size-N instances of (k, k')-Set-Cover where $k' = k(1-\beta)\ln N$ w.h.p. requires $2^{N^{\Omega(\beta)}}$ time.*

Finally, we will also use existing lower bounds in the *ungapped* setting:

Theorem 9 (Ungapped hardness of Set-Cover from $W[1] \neq$ FPT [CHKX06, Theorem 5.6])**.** *Assuming $W[1] \neq$ FPT, for all constants $c \in (0,1)$ and for all $k \leq n^c$, any $(k, k+1)$-Set-Cover instance $\mathcal{S} = (S, U, E)$ cannot be solved in time $|S|^{o(k)}$.*

Furthermore, there are even stronger set cover lower bounds assuming SETH.

Theorem 10 (Ungapped hardness of Set-Cover from SETH [PW10, Theorem 2.3])**.** *Assuming SETH, for all constants $c, \delta \in (0,1)$ and for all $k \leq n^c$, any $(k, k+1)$-Set-Cover instance $\mathcal{S} = (S, U, E)$ cannot be solved in time $O(|S|^{k-\delta})$.*

2.6 Lower bounds for DT-Construction

In this section we prove Lemma 2.3.1 and use it to derive Theorems 2 and 3. The high-level idea behind Lemma 2.3.1 is to show how, given a set cover instance $\mathcal{S}$, we can construct a function f and a distribution $\mathcal{D}$ such that the optimal set cover size for $\mathcal{S}$ is reflected in the the complexity of f under $\mathcal{D}$.

Definition 3 ($\Gamma_{\mathcal{S}}$ and $\mathcal{D}_{\mathcal{S}}$)**.** *Let $\mathcal{S} = (S, U, E)$ be a set cover instance with $|S| = n$. We identify each universe element $u \in U$ with a vector $\{0,1\}^n$, the indicator vector of its neighborhood set $\mathcal{N}_{\mathcal{S}}(u)$ (i.e. the indicator vector of the sets that contain u). We define the partial function $\Gamma_{\mathcal{S}} : \{0,1\}^n \to \{0,1\}$ as follows:*

$$\Gamma_{\mathcal{S}}(x) = \begin{cases} 0 & x = 0^n \\ 1 & x = u,\ u \in U. \end{cases}$$

The distribution $\mathcal{D}_{\mathcal{S}}$ over the support of $\Gamma_{\mathcal{S}}$ is given by the pmf

$$\mathcal{D}_{\mathcal{S}}(x) = \begin{cases} \frac{1}{2} & x = 0^n \\ \frac{1}{2|U|} & x = u, u \in U. \end{cases}$$

When $\mathcal{S}$ is clear from context we will drop the subscript and simply write Γ and $\mathcal{D}$. We observe that given any set cover $C \subseteq S$, the monotone disjunction of the variables in C computes Γ over $\mathcal{D}$. In particular, we have:

Fact 2.6.1. *If* $\mathrm{opt}(\mathcal{S}) \le k$ *then* Γ *is a monotone disjunction of* k *variables under* $\mathcal{D}$.

We now define a "parity-amplified" version of Γ. While Γ is a function over the domain $\{0,1\}^n$, this new function will be over the domain $(\{0,1\}^\ell)^n$ for some parameter $\ell \in \mathbb{N}$.

Notation. For a string $y \in (\{0,1\}^\ell)^n$, we write $y_i \in \{0,1\}^\ell$ to denote the ith block of y, and $(y_i)_j$ to denote the jth entry of the ith block. We define the function $\mathrm{BlockwisePar} : (\{0,1\}^\ell)^n \to \{0,1\}^n$:

$$\mathrm{BlockwisePar}(y) \coloneqq (\oplus y_1, \ldots, \oplus y_n),$$

where $\oplus y_i$ denotes the parity of the bits in y_i.

Definition 4 ($\Gamma_{\oplus\ell}$ and $\mathcal{D}_{\oplus\ell}$)**.** *For* Γ *and* $\mathcal{D}$ *as defined in Definition 3 and an integer* $\ell \in \mathbb{N}$*, we define the partial function* $\Gamma_{\oplus\ell} : (\{0,1\}^\ell)^n \to \{0,1\}$*,*

$$\Gamma_{\oplus\ell}(y) = \Gamma(\mathrm{BlockwisePar}(y)).$$

The distribution $\mathcal{D}_{\oplus\ell}$ *over the support of* $\Gamma_{\oplus\ell}$ *is defined as follows: to sample from* $\mathcal{D}_{\oplus\ell}$*,*

1. *First sample* $\boldsymbol{x} \sim \mathcal{D}$.
2. *For each* $i \in [n]$*, sample* $\boldsymbol{y}_i \sim \{0,1\}^\ell$ *u.a.r. among all strings satisfying* $\oplus \boldsymbol{y}_i = \boldsymbol{x}_i$*. Equivalently, sample* $\boldsymbol{y} \sim (\{0,1\}^\ell)^n$ *u.a.r. among all strings satisfying* $\mathrm{BlockwisePar}(\boldsymbol{y}) = \boldsymbol{x}$.

Fact 2.6.2 (Blockwise parity of $\mathcal{D}_{\oplus\ell}$ induces $\mathcal{D}$)**.** *For* $\boldsymbol{y} \sim \mathcal{D}_{\oplus\ell}$*, we have that* $\mathrm{BlockwisePar}(\boldsymbol{y})$ *is distributed according to* $\mathcal{D}$.

We have the following analogue of Fact 2.6.1:

Fact 2.6.3. *If* $\mathrm{opt}(\mathcal{S}) \le k$ *then* $\Gamma_{\oplus\ell}$ *is a* $k\ell$*-junta (a disjunction of* k *many parities, each over* ℓ *variables) under* $\mathcal{D}_{\oplus\ell}$.

An equivalent way of sampling from $\mathcal{D}_{\oplus\ell}$. For our proof of Lemma 2.3.1, it will be useful for us consider a different, but equivalent, way of sampling from $\mathcal{D}_{\oplus\ell}$. For $z \in (\{0,1\}^{\ell-1})^n$, $x \in \{0,1\}^n$, and $j \in [\ell]$, we write $\mathrm{ParComplete}_j(z,x)$ to denote the string $y \in (\{0,1\}^\ell)^n$ where for each block $i \in [n]$,

- All except the jth coordinate of $y_i \in \{0,1\}^\ell$ are filled in according to $z_i \in \{0,1\}^{\ell-1}$.

$$((y_i)_1, \ldots, (y_i)_{j-1}, (y_i)_{j+1}, \ldots, (y_i)_\ell) = ((z_i)_1, \ldots, (z_i)_{\ell-1}).$$

- The jth coordinate of y_i is filled in with the unique bit so that $\oplus y_i = x_i$.

Example. Consider $n = 4$ and $\ell = 3$ and $j = 2$. Then, we can view $z = (z_1, \ldots, z_4) \in (\{0,1\}^2)^4$ as a 4×2 matrix where the ith row is z_i. In this case, we may have for example:

$$z = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 1 & 1 \\ 1 & 0 \end{bmatrix} \quad x = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} \quad \longrightarrow \quad \mathrm{ParComplete}_j(z, x) = \begin{bmatrix} 1 & \mathbf{0} & 0 \\ 0 & \mathbf{1} & 0 \\ 1 & \mathbf{0} & 1 \\ 1 & \mathbf{0} & 0 \end{bmatrix}.$$

Note that the first and third columns of $\mathrm{ParComplete}_j(z, x)$, colored teal, are exactly the first and second columns of z respectively, and that the second column of $\mathrm{ParComplete}_j(z, x)$, colored purple, is filled in so that parity of each row of matches the corresponding row of x.

Definition 5 (The distribution $\mathcal{D}^j_{\oplus\ell}$)**.** *For $j \in [\ell]$, the distribution $\mathcal{D}^j_{\oplus\ell}$ is obtained via the following sampling procedure: sample $\boldsymbol{x} \sim \mathcal{D}$, $\boldsymbol{z} \sim (\{0,1\}^{\ell-1})^n$ u.a.r., and output $\mathrm{ParComplete}_j(\boldsymbol{z}, \boldsymbol{x})$.*

The following proposition on the equivalence between $\mathcal{D}_{\oplus\ell}$ and $\mathcal{D}^j_{\oplus\ell}$ can be easily verified. We defer the calculation to Appendix A.2.

Proposition 2.6.4 ($\mathcal{D}^j_{\oplus\ell}$ is equivalent to $\mathcal{D}_{\oplus\ell}$)**.** *For all $j \in [\ell]$ and $y \in (\{0,1\}^\ell)^n$,*

$$\Pr_{\boldsymbol{y} \sim \mathcal{D}_{\oplus\ell}}[\boldsymbol{y} = y] = \Pr_{\boldsymbol{y} \sim \mathcal{D}^j_{\oplus\ell}}[\boldsymbol{y} = y].$$

Constructiveness of $\Gamma_{\oplus\ell}$ and $\mathcal{D}_{\oplus\ell}$

We can efficiently compute both a circuit representation of $\Gamma_{\oplus\ell}$ and a generator for the distribution $\mathcal{D}_{\oplus\ell}$ from a given set cover instance.

Lemma 2.6.5 (Constructiveness of $\Gamma_{\oplus\ell}$ and $\mathcal{D}_{\oplus\ell}$)**.** *Let $\mathcal{S} = (S, U, E)$ be an N-vertex set cover instance with $|S| = n$ and let $\ell \leq N$ be a parameter. Then there is an algorithm that runs in $\mathrm{poly}(N)$ time and outputs a circuit representation of $\Gamma_{\oplus\ell}$ over $\mathcal{D}_{\oplus\ell}$ and a generator for the distribution $\mathcal{D}_{\oplus\ell}$.*

Proof. We separate the proof into two parts. First, we give a circuit representation of $\Gamma_{\oplus\ell}$, then we give a generator for $\mathcal{D}_{\oplus\ell}$.

A circuit for $\Gamma_{\oplus\ell}$. Recall that a circuit $\mathcal{C} : (\{0,1\}^\ell)^n \to \{0,1\}$ represents $\Gamma_{\oplus\ell} : (\{0,1\}^\ell)^n \to \{0,1\}$ over $\mathcal{D}_{\oplus\ell}$ if $\mathrm{dist}_{\mathcal{D}_{\oplus\ell}}(\mathcal{C}, \Gamma_{\oplus\ell}) = 0$. The function $\Gamma : \{0,1\}^n \to \{0,1\}$ is computed over $\mathcal{D}$ by the disjunction of all n variables. That is, $\mathrm{dist}_{\mathcal{D}}(\Gamma, x_1 \vee \cdots \vee x_n) = 0$.[6] Therefore, for $y = (y_1, \ldots, y_n) \in$

[6] This observation can equivalently be viewed as an application of Fact 2.6.1 plus the fact that $\mathrm{opt}(\mathcal{S}) \leq |S| = n$ holds for all $\mathcal{S}$.

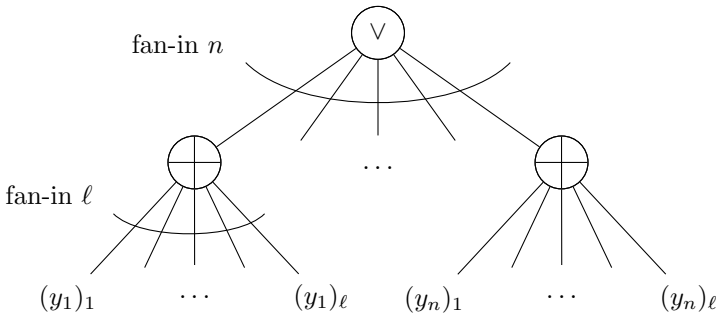


Figure 2.3: A depth-2 circuit for $\Gamma_{\oplus\ell}$ consisting of one top gate that is an OR connected to n PARITY gates, each of which is connected to a disjoint block of ℓ input variables.

$\mathrm{supp}(\mathcal{D}_{\oplus\ell})$,

$$\begin{aligned}\Gamma_{\oplus\ell}(y) &= \Gamma(\oplus y_1, \ldots, \oplus y_n) && \text{(Definition of } \Gamma_{\oplus\ell}\text{)}\\ &= (\oplus y_1) \vee \ldots \vee (\oplus y_n) && \text{(BlockwisePar}(y) \in \mathrm{supp}(\mathcal{D})\text{)}\end{aligned}$$

It follows that the circuit given by

$$\mathcal{C}(y) \coloneqq \bigvee_{i\in[n]} \bigoplus_{j\in[\ell]} (y_i)_j$$

computes $\Gamma_{\oplus\ell}$ over $\mathcal{D}_{\oplus\ell}$. See Figure 2.3 for an illustration of $\mathcal{C}$. Since this circuit has size $n \cdot \ell$ and depth 3, the first part of the lemma statement follows.

A generator for $\mathcal{D}_{\oplus\ell}$. Recall that a generator for a distribution takes uniform random bits as input and outputs bits distributed according to the desired distribution. First, we observe that there is an efficient generator for $\mathcal{D}$ using $1 + \log|U|$ uniform random bits. Specifically, use 1 uniform random bit to decide between the two cases:

(1) output 0^n

(2) output $u \in U$ uniformly at random.

The second case can be accomplished with $\log|U|$ uniform random bits. Then the following procedure generates the distribution $\mathcal{D}^1_{\oplus\ell}$:

(1) use $n(\ell-1)$ uniform random bits to select $z \in (\{0,1\}^{\ell-1})^n$

(2) use $1 + \log|U|$ bits to sample $\boldsymbol{x} \sim \mathcal{D}$

(3) output $\mathrm{ParComplete}_1(\boldsymbol{z}, \boldsymbol{x})$.

By Proposition 2.6.4, this procedure equivalently generates the distribution $\mathcal{D}_{\oplus\ell}$. The procedure uses $n(\ell-1)+1+\log|U|$ bits. We can assume without loss of generality that $1+\log|U| \le |S| = n$[7] so that $n(\ell-1)+1+\log|U| \le n\ell$. It follows that this procedure efficiently generates $\mathcal{D}_{\oplus\ell}$ from $n\ell$ uniform random bits. □

2.6.1 Warmup for Lemma 2.3.1: Lower bounds against decision tree hypotheses

We will prove Lemma 2.3.1 with the function being $\Gamma_{\oplus\ell}$ and the distribution being $\mathcal{D}_{\oplus\ell}$. The first bullet of the lemma statement is given by Fact 2.6.3, and so the bulk of the remaining work goes into establishing the second bullet of the lemma statement.

We begin with a warmup, showing the weaker statement that $\Gamma_{\oplus\ell}$ is far from any small *decision tree* under $\mathcal{D}_{\oplus\ell}$. This proof will illustrate many of the key ideas in the actual proof for DNFs, which we give in the next subsection. Furthermore, this lower bound is already sufficient to establish Theorem 1, and will be the starting point of our lower bounds for DT-Estimation that we prove in the next section.

Lemma 2.6.6. *Let $\mathcal{S} = (S, U, E)$ be an N-vertex set cover instance and let $\ell \ge 2$. If $T : (\{0,1\}^\ell)^n \to \{0,1\}$ is a decision tree of size $|T| < 2^{\mathrm{opt}(\mathcal{S})\ell/8}$, then $\mathrm{dist}_{\mathcal{D}_{\oplus\ell}}(T, \Gamma_{\oplus\ell}) \ge 1/(4N)$.*

High level idea. There are three main steps:

1. No decision tree with small average depth can approximate Γ under $\mathcal{D}$ (Claim 2.6.7).
2. Any decision tree with small average depth that approximates $\Gamma_{\oplus\ell}$ under $\mathcal{D}_{\oplus\ell}$ can be used to construct decision tree of much smaller average depth that approximates Γ under $\mathcal{D}$ (Claim 2.6.8). This is the key claim.
3. Any small size decision tree must have small average depth with respect to $\mathcal{D}_{\oplus\ell}$ (Claim 2.6.10).

Together, these three claims imply that no small size decision tree can approximate $\Gamma_{\oplus\ell}$ under $\mathcal{D}_{\oplus\ell}$, thereby yielding Lemma 2.6.6.

Claim 2.6.7 (Good approximators for Γ require large depth)**.** *Let $T : \{0,1\}^n \to \{0,1\}$ be a decision tree and $\mathcal{S} = (S, U, E)$ be an N-vertex set cover instance with $|S| = n$. If $\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[\mathrm{depth}_T(\boldsymbol{x})] < \mathrm{opt}(\mathcal{S})/2$ then $\mathrm{dist}_{\mathcal{D}}(T, \Gamma) \ge 1/(2N)$.*

Proof. Let T be a decision tree satisfying $\mathbb{E}_{\boldsymbol{x}\sim D}[\mathrm{depth}_T(\boldsymbol{x})] < \mathrm{opt}(\mathcal{S})/2$. We actually prove the stronger claim that $\mathrm{dist}_{\mathcal{D}}(T, \Gamma) \ge 1/(2|U|)$. Suppose for contradiction that $\mathrm{dist}_{\mathcal{D}}(T, \Gamma) < 1/(2|U|)$. Each $x \in \mathrm{supp}(\mathcal{D})$ has mass $\ge 1/(2|U|)$ under $\mathcal{D}$ and so we must have $\mathrm{dist}_{\mathcal{D}}(T, \Gamma) = 0$. Let

[7] If $|S| < 1 + \log|U|$, we just replicate sets until $|U| \le |S|$. This change at most doubles N and does not affect $\mathrm{opt}(\mathcal{S})$.

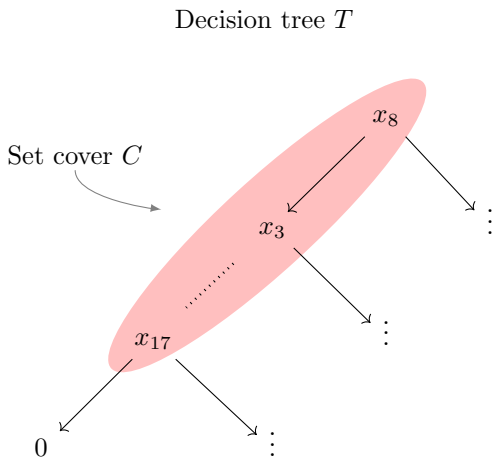


Figure 2.4: Any decision tree for Γ implicitly defines a set cover of $\mathcal{S}$ consisting of the variables highlighted in red.

$C \subseteq [n] = S$ be the set of vertices that T queries in the computation of 0^n (equivalently, C is the leftmost root-to-leaf path in T). See Figure 2.4 for an illustration of C. Since $\mathrm{dist}_{\mathcal{D}}(T, \Gamma) = 0$, we have that $T(0^n) = \Gamma(0^n) = 0$.

We claim C is a valid set cover for $\mathcal{S}$. Indeed, if some $u \in U$ is not covered by C, then $\mathcal{N}_{\mathcal{S}}(u) \cap C = \varnothing$, and u would follow this same path C as 0^n in T. This would imply that $0 = T(u) \neq \Gamma(u) = 1$, contradicting the fact that $\mathrm{dist}_{\mathcal{D}}(T, \Gamma) = 0$.

Since C is a valid set cover, it follows that $|C| \geq \mathrm{opt}(\mathcal{S})$ and so:

$$\begin{aligned} \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{D}}[\mathrm{depth}_T(\boldsymbol{x})] &\geq \Pr_{\boldsymbol{x} \sim \mathcal{D}}[\boldsymbol{x} = 0^n] \cdot |C| && (\mathrm{depth}_T(0^n) = |C|) \\ &= \frac{|C|}{2} && (\mathcal{D} \text{ places weight } \tfrac{1}{2} \text{ on } 0^n) \\ &\geq \frac{\mathrm{opt}(\mathcal{S})}{2} \end{aligned}$$

which contradicts our original assumption on the average depth of T. □

Our high-level proof strategy for the next claim is loosely inspired by [BKLS20] (which itself built on [BB19]). This proof also crucially relies on Proposition 2.6.4.

Claim 2.6.8 (Good approximators for $\Gamma_{\oplus \ell}$ yield good approximators for Γ)**.** *Let $T : (\{0,1\}^\ell)^n \to \{0,1\}$ be a decision tree such that*

$$\mathrm{dist}_{\mathcal{D}_{\oplus \ell}}(T, \Gamma_{\oplus \ell}) \leq \varepsilon \quad \textit{and} \quad \mathop{\mathbb{E}}_{\boldsymbol{y} \sim \mathcal{D}_{\oplus \ell}}[\mathrm{depth}_T(\boldsymbol{y})] \leq d.$$

Then there is a restriction $T^* : \{0,1\}^n \to \{0,1\}$ of T satisfying

$$\mathrm{dist}_{\mathcal{D}}(T^*, \Gamma) \le 2\varepsilon \quad \textit{and} \quad \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}}[\mathrm{depth}_{T^*}(\boldsymbol{x})] \le \frac{2d}{\ell}.$$

Proof. Recalling the notation from Definition 5, when $\underline{z} \in (\{0,1\}^{\ell-1})^n$ and $\underline{j} \in [\ell]$ are fixed, the function $x \mapsto \mathrm{ParComplete}_{\underline{j}}(\underline{z}, x)$ is a function from $\{0,1\}^n$ to $(\{0,1\}^\ell)^n$. Our proof proceeds by finding a suitable $\underline{z}$ and $\underline{j}$ so that $x \mapsto T(\mathrm{ParComplete}_{\underline{j}}(\underline{z}, x))$ is a tree of much smaller average depth and computes Γ accurately over $\mathcal{D}$. Restricting T according to the values specified by $\underline{z}$ and $\underline{j}$ yields the desired decision tree.

For $j \in [\ell]$ and $y \in (\{0,1\}^\ell)^n$, write $q_j(y)$ for the number of times that T, on the input y, queries $(y_i)_j$ for some $i \in [n]$. Thus, $\mathrm{depth}_T(y) = \sum_{j\in[\ell]} q_j(y)$ and likewise

$$\sum_{j\in[\ell]} \mathop{\mathbb{E}}_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[q_j(\boldsymbol{y})] = \mathop{\mathbb{E}}_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[\mathrm{depth}_T(\boldsymbol{y})] \le d.$$

Let $\underline{j} \in [\ell]$ be the index that minimizes $\mathop{\mathbb{E}}_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[q_j(\boldsymbol{y})]$. By averaging, this $\underline{j}$ must satisfy $\mathop{\mathbb{E}}_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[q_{\underline{j}}(\boldsymbol{y})] \le d/\ell$. By Proposition 2.6.4, we can write

$$\begin{aligned}
\frac{d}{\ell} &\ge \mathop{\mathbb{E}}_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[q_{\underline{j}}(\boldsymbol{y})] \\
&= \mathop{\mathbb{E}}_{\boldsymbol{y}\sim\mathcal{D}^{\underline{j}}_{\oplus\ell}}[q_{\underline{j}}(\boldsymbol{y})] && \text{(Proposition 2.6.4)} \\
&= \mathop{\mathbb{E}}_{\boldsymbol{z}\sim\mathcal{U}_{n(\ell-1)}}\left[\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}}\left[q_{\underline{j}}(\mathrm{ParComplete}_{\underline{j}}(\boldsymbol{z},\boldsymbol{x}))\right]\right]. && \text{(Definition of } \mathcal{D}^{\underline{j}}_{\oplus\ell}\text{)}
\end{aligned}$$

Similarly, we also have:

$$\begin{aligned}
\varepsilon &\ge \Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[T(\boldsymbol{y}) \ne \Gamma_{\oplus\ell}(\boldsymbol{y})] \\
&= \Pr_{\boldsymbol{y}\sim\mathcal{D}^{\underline{j}}_{\oplus\ell}}[T(\boldsymbol{y}) \ne \Gamma_{\oplus\ell}(\boldsymbol{y})] && \text{(Proposition 2.6.4)} \\
&= \mathop{\mathbb{E}}_{\boldsymbol{z}\sim\mathcal{U}_{n(\ell-1)}}\left[\Pr_{\boldsymbol{x}\sim\mathcal{D}}\left[T(\mathrm{ParComplete}_{\underline{j}}(\boldsymbol{z},\boldsymbol{x})) \ne \Gamma(\boldsymbol{x})\right]\right]. && \text{(Definition of } \mathcal{D}^{\underline{j}}_{\oplus\ell}\text{)}
\end{aligned}$$

Applying Markov's inequality twice, we have

$$\begin{aligned}
&\Pr_{\boldsymbol{z}\sim\mathcal{U}_{n(\ell-1)}}\left[\Pr_{\boldsymbol{x}\sim\mathcal{D}}\left[T(\mathrm{ParComplete}_{\underline{j}}(\boldsymbol{z},\boldsymbol{x})) \ne \Gamma(\boldsymbol{x})\right] > 2\varepsilon\right] < \frac{1}{2} \\
\text{and} \quad &\Pr_{\boldsymbol{z}\sim\mathcal{U}_{n(\ell-1)}}\left[\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}}\left[q_{\underline{j}}(\mathrm{ParComplete}_{\underline{j}}(\boldsymbol{z},\boldsymbol{x}))\right] > \frac{2d}{\ell}\right] < \frac{1}{2}.
\end{aligned}$$

And thus by a union bound, there is some fixed $\underline{z} \in \{0,1\}^{n(\ell-1)}$ satisfying

$$\Pr_{\boldsymbol{x}\sim\mathcal{D}}\left[T(\text{ParComplete}_{\underline{j}}(\underline{z},\boldsymbol{x})) \neq \Gamma(\boldsymbol{x})\right] \leq 2\varepsilon \quad \text{and} \quad \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}}\left[q_{\underline{j}}(\text{ParComplete}_{\underline{j}}(\underline{z},\boldsymbol{x}))\right] \leq \frac{2d}{\ell}.$$

The tree T^* is formed by restricting T according to $\underline{z}$ and $\underline{j}$. Also, this tree T^* satisfies $\text{depth}_{T^*}(x) = q_{\underline{j}}(\text{ParComplete}_{\underline{j}}(\underline{z},x))$ by construction. The claim then follows. □

To prove Claim 2.6.10, we first need a simple proposition stating that the probability a string $\boldsymbol{y} \sim \mathcal{D}_{\oplus\ell}$ matches some fixed substring decays exponentially with the length of the substring.

Proposition 2.6.9 ($\mathcal{D}_{\oplus\ell}$ is uniform-like)**.** *Let $\ell \geq 2$. For all $R \subseteq [\ell]$, $r \in \{0,1\}^{|R|}$, $i \in [n]$, and $b \in \{0,1\}$, we have*

$$\Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[(\boldsymbol{y}_i)_R = r \mid \oplus\boldsymbol{y}_i = b] \leq 2^{-|R|/2}$$

where $(\boldsymbol{y}_i)_R \in \{0,1\}^{|R|}$ is the substring of $\boldsymbol{y}_i \in \{0,1\}^{\ell}$ consisting of the coordinates specified by R.

Proof. We first consider the case when $|R| < \ell$. By the definition of $\mathcal{D}_{\oplus\ell}$, the conditional distribution in question is the uniform distribution over all strings in $\{0,1\}^{\ell}$ whose parity is b. The marginal distribution of this distribution over any set of $|R| < \ell$ coordinates is uniform, and therefore:

$$\Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[(\boldsymbol{y}_i)_R = r \mid \oplus\boldsymbol{y}_i = b] = 2^{-|R|}.$$

If $|R| = \ell$, then depending on whether the parity of the bits in r match b, we have:

$$\Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[\boldsymbol{y}_i = r \mid \oplus\boldsymbol{y}_i = b] = \begin{cases} 0 & \text{if } \oplus r \neq b \\ 2^{-|R|+1} & \text{if } \oplus r = b. \end{cases}$$

In either case, we have the desired probability bound. □

Claim 2.6.10 (Small trees have small average depth)**.** *Let T be a size-s decision tree, then*

$$\mathop{\mathbb{E}}_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[\text{depth}_T(\boldsymbol{y})] \leq 2\log s.$$

Proof. We start by upper bounding $\Pr[\boldsymbol{y}$ reaches $L]$ for any fixed leaf L of T. For each block $i \in [n]$, we write $R_i(L)$ to denote the variables from the ith block queried on the root-to-L path, and $r_i(L) \in \{0,1\}^{R_i(L)}$ to denote the values that the path assigns to these variables. Note that $\sum_{i\in[n]} |R_i(L)| = |L|$, the depth of L in T. With this notation in hand, for any fixed $x \in \{0,1\}^n$, we

have

$$\begin{aligned}
&\Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[\boldsymbol{y} \text{ reaches } L \mid \text{BlockwisePar}(\boldsymbol{y}) = x] \\
&= \prod_{i\in[n]} \Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[(\boldsymbol{y}_i)_{R_i(L)} = r_i(L) \mid \text{BlockwisePar}(\boldsymbol{y}) = x] \qquad \text{(Independence of the } \boldsymbol{y}_i\text{'s for fixed } x) \\
&= \prod_{i\in[n]} \Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[(\boldsymbol{y}_i)_{R_i(L)} = r_i(L) \mid \oplus\boldsymbol{y}_i = x_i] \\
&\le \prod_{i\in[n]} 2^{-|R_i(L)|/2} \qquad \text{(Proposition 2.6.9)} \\
&= 2^{-|L|/2}.
\end{aligned}$$

Since this holds for every x, it follows that

$$\Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[\boldsymbol{y} \text{ reaches } L] \le 2^{-|L|/2}. \tag{2.1}$$

We therefore conclude that

$$\begin{aligned}
\tfrac{1}{2}\cdot \mathop{\mathbb{E}}_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[\text{depth}_T(\boldsymbol{y})] &= \mathop{\mathbb{E}}_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}\left[\log\left(2^{\text{depth}_T(\boldsymbol{y})/2}\right)\right] \\
&\le \log\left(\mathop{\mathbb{E}}_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[2^{\text{depth}_T(\boldsymbol{y})/2}]\right) \qquad \text{(Concavity of } \log(\cdot)) \\
&= \log\left(\sum_{L\in T} \Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[y \text{ reaches } L]\cdot 2^{|L|/2}\right) \\
&\le \log\left(\sum_{L\in T} 2^{-|L|/2}\cdot 2^{|L|/2}\right) \qquad \text{(Equation (2.1))} \\
&= \log s.
\end{aligned}$$

Rearranging completes the proof. □

Putting things together: Proof of Lemma 2.6.6. Suppose there is some tree T computing $\Gamma_{\oplus\ell}$ with $|T| \le 2^{\text{opt}(\mathcal{S})\ell/8}$. We show that $\text{dist}(T, \Gamma_{\oplus\ell}) \ge 1/(4N)$. Suppose for contradiction that $\text{dist}(T, \Gamma_{\oplus\ell}) < 1/(4N)$. By Claim 2.6.10, we have $\mathop{\mathbb{E}}_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[\text{depth}_T(\boldsymbol{y})] < 2\cdot\log\left(2^{\text{opt}(\mathcal{S})\ell/8}\right) = \text{opt}(\mathcal{S})\ell/4$. Then by Claim 2.6.8 there is a decision tree T^* satisfying

$$\text{dist}_{\mathcal{D}}(T^*, \Gamma) < \frac{1}{2N} \qquad \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}}[\text{depth}_{T^*}(\boldsymbol{x})] < \frac{\text{opt}(\mathcal{S})}{2}.$$

But this contradicts Claim 2.6.7. □

2.6.2 Proof of Lemma 2.3.1: Lower bounds against DNF hypotheses

We extend Lemma 2.6.6 to show that $\Gamma_{\oplus\ell}$ cannot even be approximated by small DNFs. This extension will allow us to complete the proof of Lemma 2.3.1. For this section, we use the negation of Γ:

$$\overline{\Gamma}(x) = \begin{cases} 1 & x = 0^n \\ 0 & x = u, u \in U \end{cases}.$$

Analogous to Fact 2.6.1, any set cover $C \subseteq S$ yields a conjunction of k literals which computes Γ under $\mathcal{D}$.

Fact 2.6.11. *If* $\mathrm{opt}(\mathcal{S}) \leq k$*, then* $\overline{\Gamma}$ *is a conjunction of* k *literals under* $\mathcal{D}$*.*

The literals in this case are the negation of the variables in the set cover $C \subseteq S$. We will likewise use the negation of $\Gamma_{\oplus\ell}$:

$$\overline{\Gamma}_{\oplus\ell}(y) = \overline{\Gamma}(\mathrm{BlockwisePar}(y)).$$

The analogue of Fact 2.6.3 becomes:

Fact 2.6.12. *If* $\mathrm{opt}(\mathcal{S}) \leq k$ *then* $\overline{\Gamma}_{\oplus\ell}$ *is a* $k\ell$*-junta (a conjunction of* k *many parities, each over* ℓ *variables) under* $\mathcal{D}_{\oplus\ell}$*.*

Ultimately, this change allows us to prove that $\overline{\Gamma}_{\oplus\ell}$ cannot be approximated by small-size DNF formulas. If instead, one were interested in proving hardness against CNF formulas, one could work directly with the unnegated $\Gamma_{\oplus\ell}$. We find that working with DNFs is slightly less cumbersome than with CNFs which is why we focus on the negated function in this section. Specifically, we prove the following extension of Lemma 2.6.6.[8]

Lemma 2.6.13. *Let* $\mathcal{S} = (S, U, E)$ *be an* N*-vertex set cover instance and let* $\ell \geq 2$*. If* $F : (\{0,1\}^\ell)^n \to \{0,1\}$ *is a DNF of size* $|F| < 2^{\mathrm{opt}(\mathcal{S})\ell/16}$*, then* $\mathrm{dist}_{\mathcal{D}_{\oplus\ell}}(\overline{\Gamma}_{\oplus\ell}, F) \geq 1/(4N)$*.*

The high level proof strategy follows that of Lemma 2.6.6 and can be divided into the same three steps outlined in Section 2.6.1. The only difference is that "average depth" is no longer a well-defined quantity with DNF formulas. Instead, we consider "average width" which is a generalization of average depth suited to our purposes.

Claim 2.6.14 (Good approximators for $\overline{\Gamma}$ require large width)**.** *Let* $F : \{0,1\}^n \to \{0,1\}$ *be a DNF formula and* $\mathcal{S} = (S, U, E)$ *be an* N*-vertex set cover instance with* $|S| = n$*. If* $\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[\mathrm{width}_F(\boldsymbol{x})] < \mathrm{opt}(\mathcal{S})/2$*, then* $\mathrm{dist}_{\mathcal{D}}(F, \overline{\Gamma}) \geq 1/(2N)$*.*

[8] The lemma is indeed an "extension" because any size-s decision tree computing $\Gamma_{\oplus\ell}$ yields a size-s decision tree computing $\overline{\Gamma}_{\oplus\ell}$ simply by flipping leaf labels, and so Lemma 2.6.6 can equivalently be viewed as a statement about $\overline{\Gamma}_{\oplus\ell}$.

Proof. Let $F = t_1 \vee \cdots \vee t_s$ be a DNF formula. If $F(0^n) = 0$, then $\mathrm{dist}_{\mathcal{D}}(F, \overline{\Gamma}) \geq 1/2$ since $\overline{\Gamma}(0^n) = 1$. Otherwise, let t_i be the smallest width term such that $t_i(0^n) = 1$ so that $|t_i| = \mathrm{width}_F(0^n)$. Since t_i accepts the all 0s input, it is a conjunction of $|t_i|$ negated variables. Let $C \subseteq S$ be the set of variables in t_i. Since

$$\frac{|t_i|}{2} = \Pr_{\boldsymbol{x} \sim \mathcal{D}}[\boldsymbol{x} = 0^n] \cdot \mathrm{width}_F(0^n) \leq \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{D}}[\mathrm{width}_F(\boldsymbol{x})] < \frac{\mathrm{opt}(\mathcal{S})}{2},$$

C is not a set cover. Let $u \in U$ be some vertex not covered by C: $\mathcal{N}_{\mathcal{S}}(u) \cap C = \varnothing$. Then, u is encoded with 0s for all variables in C. It follows that $t_i(u) = 1$ and $F(u) = 1 \neq 0 = \overline{\Gamma}(u)$. Therefore:

$$\mathrm{dist}_{\mathcal{D}}(F, \overline{\Gamma}) \geq \Pr_{\boldsymbol{x} \sim \mathcal{D}}[\boldsymbol{x} = u] = \frac{1}{2|U|} \geq \frac{1}{2N}. \qquad \square$$

Claim 2.6.15 (Good approximators for $\overline{\Gamma}_{\oplus \ell}$ yield good approximators for $\overline{\Gamma}$)**.** *Let $F : (\{0,1\}^{\ell})^n \to \{0,1\}$ be a DNF formula such that*

$$\mathrm{dist}_{\mathcal{D}_{\oplus \ell}}(F, \overline{\Gamma}_{\oplus \ell}) \leq \varepsilon \quad \textit{and} \quad \mathop{\mathbb{E}}_{\boldsymbol{y} \sim \mathcal{D}_{\oplus \ell}}[\mathrm{width}_F(\boldsymbol{y})] \leq w.$$

Then there is a restriction $F^* : \{0,1\}^n \to \{0,1\}$ of F satisfying

$$\mathrm{dist}_{\mathcal{D}}(F^*, \overline{\Gamma}) \leq 2\varepsilon \quad \textit{and} \quad \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{D}}[\mathrm{width}_{F^*}(\boldsymbol{x})] \leq \frac{2w}{\ell}.$$

Proof. The proof is similar to that of Claim 2.6.8. First, let $q_j(y)$ denote the number of variables of the form $(y_i)_j$ for some $i \in [n]$ appearing in the smallest width term that accepts y and 0 if no term accepts y. Then, $\mathrm{width}_F(y) = \sum_{j \in [\ell]} q_j(y)$ for all $y \in \mathrm{supp}(D_{\oplus \ell})$. Therefore:

$$\sum_{j \in [\ell]} \mathop{\mathbb{E}}_{\boldsymbol{y} \sim \mathcal{D}_{\oplus \ell}}[q_j(\boldsymbol{y})] \leq w.$$

Let $\underline{j} \in [\ell]$ be the index that minimizes $\mathop{\mathbb{E}}_{\boldsymbol{y} \sim \mathcal{D}_{\oplus \ell}}[q_j(\boldsymbol{y})]$. By averaging, $\underline{j}$ satisfies $\mathop{\mathbb{E}}_{\boldsymbol{y} \sim \mathcal{D}_{\oplus \ell}}\left[q_{\underline{j}}(\boldsymbol{y})\right] \leq w/\ell$. Using Proposition 2.6.4:

$$\begin{aligned}
\frac{w}{\ell} &\geq \mathop{\mathbb{E}}_{\boldsymbol{y} \sim \mathcal{D}_{\oplus \ell}}\left[q_{\underline{j}}(\boldsymbol{y})\right] \\
&= \mathop{\mathbb{E}}_{\boldsymbol{y} \sim \mathcal{D}^{\underline{j}}_{\oplus \ell}}\left[q_{\underline{j}}(\boldsymbol{y})\right] && \text{(Proposition 2.6.4)} \\
&= \mathop{\mathbb{E}}_{\boldsymbol{z} \sim \mathcal{U}_{n(\ell-1)}}\left[\mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{D}}\left[q_{\underline{j}}(\mathrm{ParComplete}_{\underline{j}}(\boldsymbol{z}, \boldsymbol{x}))\right]\right]. && \text{(Definition of } \mathcal{D}^{\underline{j}}_{\oplus \ell})
\end{aligned}$$

Similarly:

$$\begin{aligned}
\varepsilon &\geq \Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}\left[F(\boldsymbol{y}) \neq \overline{\Gamma}_{\oplus\ell}(\boldsymbol{y})\right] \\
&= \Pr_{\boldsymbol{y}\sim\mathcal{D}^{\underline{j}}_{\oplus\ell}}\left[F(\boldsymbol{y}) \neq \overline{\Gamma}_{\oplus\ell}(\boldsymbol{y})\right] && \text{(Proposition 2.6.4)} \\
&= \mathop{\mathbb{E}}_{\boldsymbol{z}\sim\mathcal{U}_{n(\ell-1)}}\left[\Pr_{\boldsymbol{x}\sim\mathcal{D}}\left[F(\mathrm{ParComplete}_{\underline{j}}(\boldsymbol{z},\boldsymbol{x})) \neq \overline{\Gamma}(\boldsymbol{x})\right]\right]. && \text{(Definition of } \mathcal{D}^{\underline{j}}_{\oplus\ell}\text{)}
\end{aligned}$$

Applying Markov's inequality twice, we have

$$\begin{aligned}
&\Pr_{\boldsymbol{z}\sim\mathcal{U}_{n(\ell-1)}}\left[\Pr_{\boldsymbol{x}\sim\mathcal{D}}\left[F(\mathrm{ParComplete}_{\underline{j}}(\boldsymbol{z},\boldsymbol{x})) \neq \overline{\Gamma}(\boldsymbol{x})\right] > 2\varepsilon\right] < \frac{1}{2} \\
\text{and}\quad &\Pr_{\boldsymbol{z}\sim\mathcal{U}_{n(\ell-1)}}\left[\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}}\left[q_{\underline{j}}(\mathrm{ParComplete}_{\underline{j}}(\boldsymbol{z},\boldsymbol{x}))\right] > \frac{2w}{\ell}\right] < \frac{1}{2}.
\end{aligned}$$

And thus by a union bound, there is some fixed $\underline{z} \in \{0,1\}^{n(\ell-1)}$ satisfying

$$\Pr_{\boldsymbol{x}\sim\mathcal{D}}\left[F(\mathrm{ParComplete}_{\underline{j}}(\underline{z},\boldsymbol{x})) \neq \overline{\Gamma}(\boldsymbol{x})\right] \leq 2\varepsilon \quad \text{and} \quad \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}}\left[q_{\underline{j}}(\mathrm{ParComplete}_{\underline{j}}(\underline{z},\boldsymbol{x}))\right] \leq \frac{2w}{\ell}.$$

The DNF formula F^* is formed by restricting F according to the string $\underline{z}$. Also, this F^* satisfies $\mathrm{width}_{F^*}(x) = q_{\underline{j}}(\mathrm{ParComplete}_{\underline{j}}(\underline{z},x))$ by construction. The claim then follows. □

Claim 2.6.16 (Small DNFs have small average width)**.** *Let F be a size-s DNF formula for $s \geq 4$ such that $\mathrm{dist}_{\mathcal{D}_{\oplus\ell}}(F,\overline{\Gamma}_{\oplus\ell}) \leq 1/4$, then*

$$\mathop{\mathbb{E}}_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}\left[\mathrm{width}_F(\boldsymbol{y})\right] \leq 4\log(s).$$

Proof. Let $F = t_1 \vee \cdots \vee t_s$ be a DNF formula with s terms satisfying $\mathrm{dist}_{\mathcal{D}_{\oplus\ell}}(F,\overline{\Gamma}_{\oplus\ell}) \leq 1/4$. We start by upper bounding the conditional probability $\Pr[t(\boldsymbol{y}) = 1 \mid F(\boldsymbol{y}) = 1]$ for any fixed term $t \in \{t_1,\ldots,t_s\}$. We bound the probabilities $\Pr[t(\boldsymbol{y}) = 1]$ and $\Pr[F(\boldsymbol{y}) = 1]$ separately.

(1) $\Pr[F(\boldsymbol{y}) = 1] \geq 1/4$. We write

$$\begin{aligned}
\frac{1}{4} &\geq \mathrm{dist}_{\mathcal{D}_{\oplus\ell}}(F,\overline{\Gamma}_{\oplus\ell}) \\
&\geq \left|\Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}\left[F(\boldsymbol{y}) = 1\right] - \Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}\left[\overline{\Gamma}_{\oplus\ell}(\boldsymbol{y}) = 1\right]\right| \\
&= \left|\Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}\left[F(\boldsymbol{y}) = 1\right] - \frac{1}{2}\right|
\end{aligned}$$

which implies $\Pr[F(\boldsymbol{y}) = 1] \geq 1/4$.

(2) $\Pr[t(\boldsymbol{y}) = 1] \leq 2^{-|t|/2}$. For each $i \in [n]$, let $R_i(t)$ denote the variables from the ith block

which appear in the term t and let $r_i(t) \in \{0,1\}^{R_i(t)}$ denote the values assigned by those variables (i.e. 1 if the variable is unnegated in t and 0 if the variable is negated in t). Then $\sum_{i\in[n]} |R_i(t)| = |t|$, the width of t. Using this notation, for any fixed $x \in \text{supp}(\mathcal{D})$:

$$\begin{aligned}
&\Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[t(\boldsymbol{y}) = 1 \mid \text{BlockwisePar}(\boldsymbol{y}) = x] \\
&= \prod_{i\in[n]} \Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}\left[(\boldsymbol{y}_i)_{R_i(t)} = r_i(t) \mid \text{BlockwisePar}(\boldsymbol{y}) = x\right] \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(Independence of the } \boldsymbol{y}_i\text{'s for fixed } x) \\
&= \prod_{i\in[n]} \Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}\left[(\boldsymbol{y}_i)_{R_i(t)} = r_i(t) \mid \oplus\boldsymbol{y}_i = x_i\right] \\
&\leq \prod_{i\in[n]} 2^{-|R_i(t)|/2} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(Proposition 2.6.9)} \\
&= 2^{-|t|/2}.
\end{aligned}$$

Since this holds for any fixed x, it follows that

$$\Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[t(\boldsymbol{y}) = 1] \leq 2^{-|t|/2}.$$

Together, these two points imply

$$\Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[t(\boldsymbol{y}) = 1 \mid F(\boldsymbol{y}) = 1] = \frac{\Pr[t(\boldsymbol{y}) = 1]}{\Pr[F(\boldsymbol{y}) = 1]} \leq 2^{-|t|/2+2}. \tag{2.2}$$

Lastly:

$$\begin{aligned}
\frac{1}{2} \cdot \mathop{\mathbb{E}}_{\boldsymbol{y} \sim \mathcal{D}_{\oplus \ell}} [\mathrm{width}_F(\boldsymbol{y})] - 2 &= \mathop{\mathbb{E}}_{\boldsymbol{y} \sim \mathcal{D}_{\oplus \ell}} \left[\log \left(2^{\mathrm{width}_F(\boldsymbol{y})/2-2} \right) \right] \\
&\leq \log \left(\mathop{\mathbb{E}}_{\boldsymbol{y} \sim \mathcal{D}_{\oplus \ell}} \left[2^{\mathrm{width}_F(\boldsymbol{y})/2-2} \right] \right) && \text{(Concavity of log)} \\
&= \log \left(\sum_{b \in \{0,1\}} \Pr_{\boldsymbol{y} \sim \mathcal{D}_{\oplus \ell}} [F(\boldsymbol{y}) = b] \cdot \mathop{\mathbb{E}}_{\boldsymbol{y} \sim \mathcal{D}_{\oplus \ell}} \left[2^{\mathrm{width}_F(\boldsymbol{y})/2-2} \mid F(\boldsymbol{y}) = b \right] \right) \\
&\leq \log \left(\mathop{\mathbb{E}}_{\boldsymbol{y} \sim \mathcal{D}_{\oplus \ell}} \left[2^{\mathrm{width}_F(\boldsymbol{y})/2-2} \mid F(\boldsymbol{y}) = 1 \right] \right) \\
& && \text{($\mathrm{width}_F(\boldsymbol{y}) = 0$ if $F(\boldsymbol{y}) = 0$ and $\Pr[F(\boldsymbol{y}) = b] \leq 1$)} \\
&\leq \log \left(\sum_{i \in [s]} 2^{|t_i|/2} \cdot \Pr_{\boldsymbol{y} \sim \mathcal{D}_{\oplus \ell}} [t_i(\boldsymbol{y}) = 1 \mid F(\boldsymbol{y}) = 1] \right) \\
&= \log \left(\sum_{i \in [s]} 2^{|t_i|/2-2} \cdot 2^{-|t_i|/2+2} \right) && \text{(Equation (2.2))} \\
&= \log s.
\end{aligned}$$

Rearranging and applying the assumption that $2 \leq \log(s)$ completes the proof. □

Putting things together: Proof of Lemma 2.6.13 Suppose there is some DNF formula F computing $\overline{\Gamma}_{\oplus \ell}$ with $|F| \leq 2^{\mathrm{opt}(\mathcal{S})\ell/16}$. We show that $\mathrm{dist}(F, \overline{\Gamma}_{\oplus \ell}) \geq 1/(4N)$. Suppose for contradiction that $\mathrm{dist}(T, \overline{\Gamma}_{\oplus \ell}) < 1/(4N) \leq 1/4$. If $|F| < 4$, we add dummy terms (e.g. by replicating the terms already in F) so that $|F| \geq 4$. We can then apply Claim 2.6.16: $\mathop{\mathbb{E}}_{\boldsymbol{y} \sim \mathcal{D}_{\oplus \ell}} [\mathrm{width}_F(\boldsymbol{y})] < 4 \cdot \log \left(2^{\mathrm{opt}(\mathcal{S})\ell/16} \right) = \mathrm{opt}(\mathcal{S})\ell/4$. Then by Claim 2.6.15, there is a DNF formula F^* satisfying

$$\mathrm{dist}_{\mathcal{D}}(F^*, \Gamma) < \frac{1}{2N} \qquad \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{D}} [\mathrm{depth}_{F^*}(\boldsymbol{x})] < \frac{\mathrm{opt}(\mathcal{S})}{2}.$$

But such an F^* contradicts Claim 2.6.14. □

The last steps: finishing the proof of Lemma 2.3.1. We prove the following lemma which immediately implies Lemma 2.3.1.

Lemma 2.6.17 ($\overline{\Gamma}_{\oplus \ell}$ proves Lemma 2.3.1)**.** *Let $\mathcal{S} = (S, U, E)$ be an N-vertex instance of (k, k')-*Set-Cover *and $\ell \leq N$. Then there is an algorithm that runs in $\mathrm{poly}(N)$ time and outputs a circuit representation of $\overline{\Gamma}_{\oplus \ell}$ under $\mathcal{D}_{\oplus \ell}$ and a generator for $\mathcal{D}_{\oplus \ell}$ which satisfies:*

- *If $\mathrm{opt}(\mathcal{S}) \leq k$, then $\overline{\Gamma}_{\oplus \ell}$ is a $k\ell$-junta under $\mathcal{D}_{\oplus \ell}$.*
- *If $\mathrm{opt}(\mathcal{S}) > k'$, then any DNF of size $\leq 2^{k'\ell/16}$ is $\frac{1}{4N}$-far from $\overline{\Gamma}_{\oplus \ell}$ under $\mathcal{D}_{\oplus \ell}$.*

Proof. By Lemma 2.6.5, there is an algorithm that runs in poly(N) time and outputs a circuit representation of $\Gamma_{\oplus\ell}$ and a generator for $\mathcal{D}_{\oplus\ell}$. Augmenting the circuit for $\Gamma_{\oplus\ell}$ with a single NOT gate yields a circuit for $\overline{\Gamma}_{\oplus\ell}$. Moreover, we have shown:

- if $\mathrm{opt}(\mathcal{S}) \leq k$, then $\overline{\Gamma}_{\oplus\ell}$ is a $k\ell$-junta under $\mathcal{D}_{\oplus\ell}$; (Fact 2.6.12)
- if $\mathrm{opt}(\mathcal{S}) > k'$, then any DNF of size $\leq 2^{k'\ell/16}$ is $\frac{1}{4N}$-far from $\overline{\Gamma}_{\oplus\ell}$ under $\mathcal{D}_{\oplus\ell}$; (Lemma 2.6.13)

which completes the proof of the lemma. □

2.6.3 Implications of Lemma 2.3.1

Proofs of Theorem 1 and Theorem 2

In this section, we prove the following theorem.

Theorem 11. *Let $\mu : \mathbb{N} \to \mathbb{N}$ be any computable, non-decreasing function satisfying $\mu(n) = o\left(\frac{\log\log n}{\log\log\log n}\right)$. Assuming randomized ETH, there is some constant $\lambda \in (0,1)$, a function $f : \{0,1\}^n \to \{0,1\}$, and distribution $\mathcal{D}$ over $\{0,1\}^n$ such that* DT-Construction$(s, 1/n)$ *cannot be solved in time*

$$s^{\lambda\cdot\left(\frac{\log\log s}{\mu(n)\log\log\log s}\right)}$$

for f and for any $s \leq n^{\mu(n)}$, even if f is promised to be a $(\log n)$-junta over $\mathcal{D}$ and the algorithm returns a DNF hypothesis.

Theorems 1 and 2 immediately follow as a consequence of this theorem by choosing $\mu(n) = 1$.

Proof of Theorem 11. We give a reduction from gapped set cover. Let $\mathcal{S} = (S, U, E)$ be an N-vertex $\left(k, \frac{1}{2}\left(\frac{\log N}{\log\log N}\right)^{1/k}\right)$-Set-Cover instance where k is taken to be

$$k = \frac{1}{2}\cdot\frac{\log\log N}{\log\log\log N}.$$

Using Lemma 2.6.17 with $\ell = \log(N)/k$, we obtain the target function $\overline{\Gamma}_{\oplus\ell} : \{0,1\}^{N\ell} \to \{0,1\}$ and the distribution $\mathcal{D}_{\oplus\ell}$.[9]

Let $\mu : \mathbb{N} \to \mathbb{N}$ be as in the theorem statement. Set $s := (N\ell)^{\mu(N\ell)}$. We show that any algorithm for DT-Construction$(s, 1/(4N))$ running in time $s^{\lambda\cdot\left(\frac{\log\log s}{\mu(N\ell)\log\log\log s}\right)}$ for $0 < \lambda \leq 1/128$ can be used to solve $\mathcal{S}$ in time $N^{8\lambda\cdot k}$ even if the output of the algorithm is a DNF formula.

We run the algorithm for DT-Construction$(s, 1/(4N))$ on $\overline{\Gamma}_{\oplus\ell}$ and $\mathcal{D}_{\oplus\ell}$ and terminate it after

$$N^{4\lambda\cdot\left(\frac{\log\log N}{\log\log\log N}\right)} = N^{8\lambda k}$$

[9] Technically, $\overline{\Gamma}_{\oplus\ell}$ is a function defined on $|S|\ell$ bits, but as $|S| \leq N$ we can pad the inputs to be $N\ell$ bits long.

times steps. The algorithm outputs some DNF formula F. We estimate the error of F and $\overline{\Gamma}_{\oplus\ell}$ over the distribution $\mathcal{D}_{\oplus\ell}$ and output "YES" if the error is $\leq 1/(4N)$ and "NO" otherwise.

Runtime. Constructing the circuit for $\overline{\Gamma}_{\oplus\ell}$ and the generator for $\mathcal{D}_{\oplus\ell}$ requires $\mathrm{poly}(N)$ time by Lemma 2.6.17. We can efficiently sample from the distribution $\mathcal{D}_{\oplus\ell}$ to efficiently estimate the error of the output decision tree via random samples. So the overall runtime of our algorithm is $\leq N^{8\lambda k}$.

Correctness. To prove the reduction is correct, we show that if there is a size k set cover for $\mathcal{S}$ then we output YES with high probability and otherwise if S requires a set cover of size at least

$$\frac{1}{2}\left(\frac{\log N}{\log\log N}\right)^{1/k}$$

then we output NO with high probability.

YES **case:** $\mathrm{opt}(S) \leq k$**.** In this case, by Lemma 2.6.17, $\overline{\Gamma}_{\oplus\ell}$ is computed exactly by a $\mathrm{opt}(S)\ell \leq k\ell = \log N$-junta over $\mathcal{D}_{\oplus\ell}$. Hence, it is computed by a DNF of width $k\ell$. The size of this DNF is at most $2^{k\cdot\ell} = N \leq (N\ell)^{\mu(N\ell)} = s$. To upper bound the runtime, we start by calculating

$$\begin{aligned}
\frac{\log\log s}{\log\log\log s} &\leq \frac{\log\left(2\mu(N^2)\log N\right)}{\log\log\log N} && (N \leq s \leq N^{2\mu(N^2)}) \\
&\leq \frac{\log\left((\log N)^2\right)}{\log\log\log N} && (\text{Assumption on } \mu\colon\ 2\mu(N^2) \leq \log N) \\
&= 4k. && (2.3)
\end{aligned}$$

By our assumption on DT-CONSTRUCTION$(s, 1/(4N))$, in the yes case, the algorithm runs for

$$\begin{aligned}
s^{\lambda\cdot\left(\frac{\log\log s}{\mu(N\ell)\log\log\log s}\right)} &\leq s^{4\lambda k/\mu(N\ell)} && (\text{Equation (2.3)}) \\
&= (N\ell)^{4\lambda k} && (s = (N\ell)^{\mu(N\ell)}) \\
&\leq N^{8\lambda k} && (N\ell \leq N^2)
\end{aligned}$$

time steps and outputs a size-s DNF formula with error $\leq 1/(4N)$. Therefore, our algorithm outputs YES with high probability (where the probability is taken over the random sampling procedure).

NO **case:** $\mathrm{opt}(S) > \frac{1}{2}\left(\frac{\log N}{\log\log N}\right)^{1/k}$. By Lemma 2.6.17 any DNF for $\overline{\Gamma}_{\oplus\ell}$ with size at most $2^{\mathrm{opt}(\mathcal{S})\ell/16}$ has error at least $1/(4N)$. The runtime bound on our algorithm serves as an upper bound on the size of the DNF built by the DT-CONSTRUCTION algorithm. Therefore, it is sufficient to show that

$$N^{8\lambda\cdot k} < 2^{\mathrm{opt}(S)\ell/16} \tag{2.4}$$

because this bound shows that our DNF must have error at least $1/(4N)$. Recalling that $k = \frac{1}{2} \cdot \frac{\log\log N}{\log\log\log N}$, we have $(2k^2)^k < \frac{\log N}{\log\log N}$. We observe

$$\begin{aligned} \mathrm{opt}(\mathcal{S}) &> \frac{1}{2}\left(\frac{\log N}{\log\log N}\right)^{1/k} \\ &> k^2 \\ &\geq 128\lambda k^2 && (128\lambda \leq 1) \end{aligned}$$

which shows $k\ell(8\lambda k) < \mathrm{opt}(\mathcal{S})\ell/16$. Exponentiating both sides and using the fact that $N = 2^{k\ell}$ completes the calculation and establishes Equation (2.4). It follows that our algorithm finds the error to be $> 1/(4N)$ and outputs No with high probability.

Refuting randomized ETH. We now have an algorithm for solving $\left(k, \frac{1}{2}\left(\frac{\log N}{\log\log N}\right)^{1/k}\right)$-Set-Cover in time $N^{8\lambda k}$ with high probability. By Theorem 6, there is a constant $c \in (0,1)$ such that $\left(k, \frac{1}{2}\left(\frac{\log N}{\log\log N}\right)^{1/k}\right)$-Set-Cover cannot be solved with high probability in time N^{ck}. Therefore, we derive a contradiction for any $\lambda \leq \min\{c/8, 1/128\}$. □

PAC learning hardness

In this section, we discuss corollaries of Theorem 11.

Corollary 2.6.18 (Hardness of learning decision trees, DNFs, and CNFs)**.** *Assuming randomized ETH, there is a constant $\lambda \in (0,1)$ such that decision trees cannot be distribution-free, properly PAC learned to accuracy $\varepsilon = 1/n$ in time $s^{\lambda \frac{\log\log s}{\log\log\log s}}$ where s is the size of the target. The same result also holds for properly learning DNFs and CNFs with size-s targets.*

Proof. Let $\mathcal{L}$ be a distribution-free, proper learning algorithm for the class $\mathcal{T}$ of decision trees. We claim $\mathcal{L}$ can be used to solve DT-Construction. In particular, let $f : \{0,1\}^n \to \{0,1\}$ and $\mathcal{D}$ be an instance of DT-Construction$(s, 1/n)$. We run the learning algorithm on f and $\mathcal{D}$ and $\varepsilon = 1/n$. If $\mathcal{L}$ requests a random sample, we generate $x \sim \mathcal{D}$ using the generator for $\mathcal{D}$ and evaluate $f(x)$ using the circuit for f and return $(x, f(x))$ to $\mathcal{L}$. Since generating a sample from $\mathcal{D}$ and evaluating the circuit for f are both $\mathrm{poly}(n)$-time operations the overall runtime is dominated by the runtime of $\mathcal{L}$. Theorem 11 then implies the desired time bound by setting $\mu(n) = 1$.

If $\mathcal{L}$ is a learning algorithm for DNFs, we obtain the same hardness as in the decision tree case since any size-s decision tree target is equivalently a size-s DNF target. Moreover, Theorem 11 also applies when the output of the DT-Construction algorithm is a DNF formula. A symmetric argument works similarly for CNFs. □

Proof of Theorem 3

In this section, we observe that the number of relevant inputs to $\Gamma_{\mathcal{S}}$ exactly characterizes the set cover complexity of $\mathcal{S}$. As a result, hardness of approximating set cover can be directly translated into hardness of distribution-free, proper PAC learning k-juntas. The next theorem formalizes this observation and was already implicit in [ABF+09].

Theorem 12 (Learning k-juntas is as hard as Set-Cover)**.** *Suppose there is a distribution-free PAC learning algorithm that runs in time $t(n,k)$ and learns the class of k-juntas over $\{0,1\}^n$ to accuracy $\varepsilon = O(1/n)$ by hypotheses which are $g(k,n)$-juntas for some function $g : \mathbb{N}^2 \to \mathbb{N}$ satisfying $k \leq g(k,n)$. Then $(k, g(k,n))$-Set-Cover can be solved with high probability in time $t(n,k)$.*

Proof. Let $\mathcal{S} = (S, U, E)$ be an instance of $(k, g(k,n))$-Set-Cover. We construct the function $\Gamma : \{0,1\}^{|S|} \to \{0,1\}$ and the distribution $\mathcal{D}$ over $\{0,1\}^{|S|}$. Run the learning algorithm on Γ and $\mathcal{D}$ with $\varepsilon = 1/(4|S|)$ for $t(k,|S|)$ time steps. It outputs some truth table representation of a junta. We output Yes if and only if this truth table has size at most $2^{g(k,n)}$ and has error at most $1/(4|S|)$. The correctness of the reduction follows from Fact 2.6.1. □

Corollary 2.6.19. *There is no distribution-free PAC learning algorithm for properly learning k-juntas to accuracy $\varepsilon = O(1/n)$ over $\{0,1\}^n$ that runs in time:*

- *$n^{o(k)}$, assuming randomized $W[1] \neq \mathrm{FPT}$;*
- *$O(n^{k-\lambda})$, for all $\lambda > 0$, assuming randomized SETH.*

These results hold in the regime where $k \leq n^c$ for some absolute constant $c < 1$.

Proof. By Theorem 12, distribution-free properly PAC learning k-juntas is equivalent to $(k, k+1)$-Set-Cover. The first bullet follows by combining Theorems 9 and 12. The second bullet follows by combining Theorems 10 and 12. □

2.7 Proof of Theorem 4

The PAC learning lower bound from Section 2.6.3 applies to properly learning decision trees. In this setting, the concept class is $\mathcal{T} = \{T : \{0,1\}^n \to \{0,1\} \mid T \text{ is a decision tree}\}$. So the learner is allowed to output a decision tree hypothesis that may be much larger than the target. One could instead consider the problem of properly learning the class of size-s decision trees: $\mathcal{T}_s = \{T : \{0,1\}^n \to \{0,1\} \mid T \text{ is a size-}s \text{ decision tree}\}$. This problem is strictly harder than learning decision trees since the output must satisfy a size constraint. Indeed, against this class, we are able to adapt the proof of Theorem 11 to obtain a stronger lower bound.

Theorem 13. *Assuming randomized ETH, there is a constant $\lambda \in (0,1)$ such that* DT-CONSTRUCTION$(s, \frac{1}{n})$ *cannot be solved in time $n^{\lambda \log s}$ if the algorithm has to return a size-s DNF hypothesis, even when the function is promised to be a $\log s$-junta.*

Proof. This proof is a combination of the proofs of Theorems 11 and 14. The analysis is similar so we only outline the important details here. In particular, let $\mathcal{S} = (S, U, E)$ be an N-vertex $\left(k, \frac{1}{2}\left(\frac{\log N}{\log\log N}\right)^{1/k}\right)$-SET-COVER instance where k is taken to be $k = \frac{1}{2} \cdot \frac{\log\log N}{\log\log\log N}$ for N large enough so that $32k < \frac{1}{2}\left(\frac{\log N}{\log\log N}\right)^{1/k}$. Using Theorem 6, there is a constant $c \in (0,1)$ such that $\mathcal{S}$ cannot be solved in time N^{ck}. We derive a contradiction for any algorithm for DT-CONSTRUCTION$(s, \frac{1}{n})$ that returns a size-s DNF and runs in time $n^{\lambda \log s}$ for $\lambda \le c/5$.

Use Lemma 2.6.13 with $\ell = 4$ (as in Theorem 14) to obtain the target function $\overline{\Gamma}_{\oplus 4} : \{0,1\}^{4N} \to \{0,1\}$ and the distribution $\mathcal{D}_{\oplus 4}$. Run DT-CONSTRUCTION$(s, \frac{1}{4N})$ with $s \coloneqq 2^{4k}$ on $\overline{\Gamma}_{\oplus 4} : \{0,1\}^{4N}$ and $\mathcal{D}_{\oplus 4}$ for $N^{5\lambda k}$ time steps where $\lambda \le c/5$. Output yes if and only if the DNF formula returned by the algorithm as size at most 2^{4k} and error less than $1/(4N)$. The correctness of the no case follows from the fact that $32k < \frac{1}{2}\left(\frac{\log N}{\log\log N}\right)^{1/k}$ and so the DNF lower bound from Lemma 2.6.13 ensures $2^{\mathrm{opt}(\mathcal{S})\ell/16} > 2^{4k}$. Since the algorithm returns a size-2^{4k} DNF formula if one exists, this separation between the DNF sizes is sufficient to establish correctness. □

As in the case of Corollary 2.6.18, this theorem yields hardness of properly PAC learning the class of size-s decision trees.

Corollary 2.7.1. *Assuming randomized ETH, there is a constant $\lambda \in (0,1)$ such that the class $\mathcal{T}_s$ of size-s decision trees cannot be distribution-free, properly PAC learned to accuracy $\varepsilon = 1/n$ in time $n^{\lambda \log s}$. The same result also holds for properly learning size-s DNFs and CNFs.*

2.8 Proof of Theorem 5

In this section, we outline a concrete path towards proving optimal lower bounds for DT-CONSTRUCTION. In particular, we show that better lower bounds for gapped set cover yields better lower bounds for DT-CONSTRUCTION. Specifically, the main theorem assumes Conjecture 1 and proves an $n^{\Omega(\log s)}$ lower bound for DT-CONSTRUCTION$(s, \frac{1}{n})$.

Theorem 14. *Assume Conjecture 1, then there is a constant $\lambda \in (0,1)$ such that* DT-CONSTRUCTION$(s, \frac{1}{n})$ *cannot be solved in time $n^{\lambda \log s}$, even when the target is a $\log s$-junta and the algorithm is allowed to return a DNF hypothesis.*

Proof. Let $\beta \in (0,1)$ be as in the statement of the Conjecture 1. Assume there is an algorithm for DT-CONSTRUCTION$(s, \frac{1}{n})$ running in time $n^{\lambda \cdot \log s}$ for any $\lambda \le (1-\beta)/(40 \log e)$. Then, following the proof strategy of Theorem 11, we derive a contradiction by solving $(k, k \cdot (1-\beta)\ln N)$-SET-COVER

over N vertices in time $N^{5\lambda k}$. Let $\mathcal{S} = (S, U, E)$ be an N-vertex $(k, k \cdot (1-\beta)\ln N)$-Set-Cover instance for $k \in \mathbb{N}$. Using Lemma 2.6.17 with $\ell = 4$, we obtain the target function $\overline{\Gamma}_{\oplus 4} : \{0,1\}^{4N} \to \{0,1\}$ and the distribution $\mathcal{D}_{\oplus 4}$. We run the algorithm for DT-Construction$(s, \frac{1}{4N})$ on $\overline{\Gamma}_{\oplus 4}$ and $\mathcal{D}_{\oplus 4}$ with $s \coloneqq 2^{4k}$ and terminate it after $N^{5\lambda k}$ time steps. The output is some DNF formula F. We estimate the error of F over $\mathcal{D}_{\oplus 4}$ and output Yes if it's less than $1/(4N)$ and No otherwise.

Runtime. By Lemma 2.6.17, we can construct the circuit for $\overline{\Gamma}_{\oplus 4} : \{0,1\}^{4N} \to \{0,1\}$ and generator for $\mathcal{D}_{\oplus 4}$ in $\mathrm{poly}(N)$-time. Moreover, we can use random sampling to efficiently estimate the error of F over $\mathcal{D}_{\oplus 4}$. Therefore, the runtime of the reduction is dominated by $N^{5\lambda k}$.

Correctness. We handle the yes case and the no case separately.

Yes case: $\mathrm{opt}(\mathcal{S}) \le k$. By Lemma 2.6.17, $\overline{\Gamma}_{\oplus 4}$ is a $4k$-junta over $\mathcal{D}_{\oplus 4}$. Therefore, it is a decision tree of size $s = 2^{4k}$ and DT-Construction$(s, \frac{1}{4N})$ runs in time

$$(4N)^{\lambda \cdot \log s} = (2N)^{4\lambda k} \le N^{5\lambda k}.$$

The output is DNF formula with error at most $1/(4N)$. It follows that our algorithm outputs Yes with high probability.

No case: $\mathrm{opt}(\mathcal{S}) > k \cdot (1-\beta)\ln N$. By Lemma 2.6.17, any DNF for $\overline{\Gamma}_{\oplus 4}$ with size at most $2^{\mathrm{opt}(\mathcal{S})/8}$ has error at least $1/(4N)$. Using the assumption on $\mathrm{opt}(\mathcal{S})$:

$$\begin{aligned} 2^{\mathrm{opt}(\mathcal{S})/8} &> N^{k(1-\beta)/(8\log e)} \\ &\ge N^{5\lambda k} && ((1-\beta)/(40\log e) \ge \lambda) \end{aligned}$$

which shows that the DNF output by the algorithm must have error at least $1/(4N)$. It follows that our algorithm outputs No with high probability. □

As discussed in Section 2.6.3, this lower bound for DT-Construction implies a lower bound for PAC learning decision trees.

Corollary 2.8.1. *Assume Conjecture 1, then there is a constant $\lambda \in (0,1)$ such that decision trees cannot be distribution-free, properly PAC learned to accuracy $\varepsilon = 1/n$ in time $n^{\lambda \log s}$ where s is the size of the decision tree target. The same result also holds for properly learning DNFs and CNFs.*

The proof of this corollary is identical to that of Corollary 2.6.18.

2.9 Discussion and future work

Our work makes new progress on the longstanding open problem of determining the complexity of properly PAC learning decision trees. A natural avenue for future work is to close the remaining gap between our lower bound of $n^{\tilde{\Omega}(\log \log s)}$ and the $n^{O(\log s)}$ runtime of Ehrenfeucht and Haussler's algorithm. Our techniques point to an approach towards an $n^{\Omega(\log s)}$ lower bound via Conjecture 1, which adds further motivation to the study of parameterized Set-Cover.

More broadly, there is a growing and concerted effort within the machine learning community to design algorithms that produce *simple* hypotheses, such as decision trees, especially in the context of high-stakes applications where interpretability is paramount; see e.g. the position paper [Rud19]. Our lower bounds show that interpretability can come at the price of computational intractability, even under strong assumptions on the target function. There is substantial practical motivation for the development of a theoretical understanding of such tradeoffs and how they can be mitigated. For example, a concrete next step from our work is to identify reasonable assumptions under which our lower bounds can be circumvented.

Chapter 3

The complexity of learning decision trees with queries

3.1 Introduction

In this chapter, we consider the task of constructing *optimal* decision tree representations of data using queries. A standard formalization of this task is the problem of *properly PAC learning decision trees*: given query access to a target function f and a distribution $\mathcal{D}$, construct the optimal decision tree hypothesis for f under $\mathcal{D}$. Valiant's original definition of the PAC model [Val84] considered learners with both passive access to the target function in the form of random labeled examples as well as active access in the form of queries. Both settings have since been intensively studied. Valiant's motivation for the more powerful query setting came from modeling interactions with an expert ("[an] important aspect of the formulation is that the notion of oracles makes it possible to discuss a whole range of teacher-learner interactions beyond the mere identification of examples"). The query setting also models the task of converting an existing accurate but inscrutable hypothesis, for which one has query access to, into a more intelligible representation—once again, decision trees are a canonical sought-for representation for this task [CS95, BS96, VAB07, ZH16, BKB17, VLJ+17, FH17, VS20].

As we have seen, the NP-hardness of properly learning decision trees from *random examples* is a foundational result known since the early days of PAC learning [Ang, PV88]. The question of whether there exists an efficient *query* learner, on the other hand, has been raised repeatedly over the years, in research papers [Bsh93, GLR99, MR02] and surveys [Fel16]. We resolve this question by showing that properly learning decision trees is NP-hard even for query learners:

Theorem 15. *There is an absolute constant $\varepsilon > 0$ such that the following holds. Suppose there is an algorithm 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 $\mathcal{D}$, runs in time $t(n)$ and w.h.p. outputs a size-s decision tree h that is ε-close to f under $\mathcal{D}$. Then* $\mathrm{SAT} \in \mathrm{RTIME}(\mathrm{poly}(t(n^2 \mathrm{polylog} n)))$.

Theorem 15 addresses a stark gap in our understanding of the problem. The fastest known algorithm runs in exponential time, $2^{O(n)}$ for all values of s. There were no previous lower bounds, leaving open the possibility of a $\mathrm{poly}(n, s)$-time algorithm. Indeed, existing query learners for various relaxations of the problem had suggested that such an algorithm was within striking distance. Theorem 15 provides for the first time strong evidence that there are no polynomial-time, or indeed even subexponential-time, algorithms for the problem.

3.1.1 Background and Context

Hardness of properly learning decision trees from random examples. NP-hardness in the setting of random examples has been known since the seminal work of Pitt and Valiant [PV88]. Their paper, which initiated the study of the hardness of proper learning, attributed the result to an unpublished manuscript of Angluin [Ang]. Subsequently, Hancock, Jiang, Li, and Tromp [HJLT96] obtained hardness even of *weakly-proper* learning, where the algorithm is allowed to return a decision tree of size larger than that of the target. There have since been several works [ABF+09, KST23b, Bsh23] further improving [HJLT96]'s result.

These works build crucially on a simple reduction from SetCover, a reduction variously attributed to Levin [Lev73], Angluin [Ang], and Haussler [Hau88] which we used as the basis of Theorem 1. We describe how this technique is limited to the setting of random examples in Section 3.2.1.

Algorithms for properly learning decision trees. There is a simple Occam algorithm for properly learning decision trees from random examples: for a size-s decision tree target, draw $O(s \log(n))$ many labeled examples and use dynamic programming to find a size-s decision tree hypothesis that fits the dataset perfectly (see e.g. [GLR99, MR02]). Standard generalization bounds [BEHW89] show that this algorithm satisfies the PAC guarantee. Its runtime is $2^{O(n)}$, with the dynamic program being the bottleneck.

Ehrenfeucht and Haussler [EH89] gave a faster algorithm that runs in time $n^{O(\log s)}$, but their algorithm is only weakly proper: for a size-s target, its hypothesis can be as large as $n^{\Omega(\log s)}$. This large gap is a significant drawback—decision tree hypotheses are interpretable and fast to evaluate insofar as they are *small*—and [EH89] stated as the first open problem of their paper that

of designing algorithms that produce smaller hypotheses. There has been no progress on this problem in the setting of random examples since 1989.

The power of queries. In contrast, granting the learner queries enables the design of several polynomial-time algorithms that *almost* solve the problem of properly learning decision trees. Already in his original paper [Val84] (see also [Ang88]), Valiant gave a polynomial-time query algorithm for properly learning monotone DNFs; consequently, for size-s monotone decision tree targets Valiant's algorithm returns a size-s monotone DNF as its hypothesis. Other such results include polynomial-time query learners for general decision tree targets that output depth-3 formulas [Bsh93] and polynomials [KM12, SS93] as hypotheses. As further demonstration of the power of queries, a recent work of Blanc, Lange, Qiao, and Tan [BLQT22] gives an almost-polynomial-time ($\mathrm{poly}(n) \cdot s^{O(\log \log s)}$ time) query algorithm that properly learns decision trees under the *uniform distribution*. Finally, the query model opens the possibility of circumventing long-known SQ lower bounds for the problem [BFJ+94], which show that in the setting of random examples all SQ algorithms must take time $n^{\Omega(\log s)}$.

Taken together, this was all evidence in favor of a polynomial-time, or at least a mildly-super-polynomial time algorithm for properly learning decision trees with queries. In light of Theorem 15, even a subexponential-time algorithm is now unlikely.

3.1.2 Other related work

Scarcity of hardness results for PAC learning with queries. Theorem 15 adds to a dearth of NP-hardness results for the model of PAC learning with queries. Indeed, we are aware of only one other such result: in [Fel06] Feldman proved that DNFs are NP-hard to properly learn with queries, resolving a longstanding problem of Valiant [Val84, Val85]. As Feldman remarked in his paper, this was the first NP-hardness result, for *any* learning task, for the model of PAC learning with queries. (Our techniques are entirely different from [Fel06]'s.)

Related to the scarcity of hardness results, there are numerous query algorithms, for a variety of learning tasks, whose runtimes remain unmatched in the setting of random examples. It is also well known that under standard cryptographic assumptions, PAC learning with queries is strictly more powerful than from random examples only.

3.1.3 Technical remarks about Theorem 15

Hardness for constant error. A notable aspect of Theorem 15 is that it rules out learners that are allowed constant error. This was not known even in the setting of random examples, where existing hardness results only hold for inverse-polynomial error: prior to our work, there were no lower bounds ruling out algorithms for properly learning size-s DTs, from random examples only, in

time say $(ns)^{O(1/\varepsilon)}$, which is polynomial for constant ε. (Feldman's NP-hardness result for properly learning DNFs with queries also only holds for inverse-polynomial error.)

Implications for decision tree minimization. The actual result that we prove is stronger than as stated in Theorem 15: it holds even if the learner is given explicit descriptions of the target function f and the distribution $\mathcal{D}$ as inputs. Furthermore, the target function can even be given to the learner in the form of a decision tree. For this reason, our result also has implications for the problem of *decision tree minimization*: given a decision tree, find an equivalent one of minimum size. We recover the best known hardness of approximation result for this problem [ZB00, Sie08] via what is, in our opinion, a much simpler proof. Our proof also yields a stronger result: we show that the problem remains hard even if the resulting tree only has to agree with the original tree on a small given set of inputs.

Implications for testing decision trees. Another aspect in which the actual result we prove is stronger than as stated in Theorem 15 is that it even rules out distribution-free *testers* for decision trees. While there's a large body of work giving lower bounds for testing various classes of functions, the vast majority of these results are information-theoretic in nature, focusing on query complexity, with far fewer computational lower bounds. Our result does not rule out decision tree testers with low query complexity, but it shows that even if such a tester exists, it must nevertheless run in exponential time (unless SAT admits a subexponential time algorithm).

3.2 Technical Overview

3.2.1 Why the query setting necessitates new techniques

Before delving into our techniques, we describe the key construction [Lev73, Ang, Hau88] at the heart of all previous results on the hardness of properly learning decision trees from random examples [Ang, HJLT96, ABF+09, KST23b, Bsh23] and discuss why it is limited this setting. Consider the reduction from SetCover to the problem of properly learning disjunctions from Section 2.6. Let $\mathcal{S} = \{S_1, \ldots, S_n\}$ be a SetCover instance over the universe $[m]$ and define $u^{(1)}, \ldots, u^{(m)} \in \{0,1\}^n$ where

$$(u^{(j)})_i = \begin{cases} 1 & \text{if } j \in S_i \\ 0 & \text{otherwise.} \end{cases}$$

Let $C \subseteq [n]$ be the indices of an optimal set cover for $\mathcal{S}$ and consider the target disjunction $f : \{0,1\}^n \to \{0,1\}$,

$$f = \bigvee_{i \in C} x_i.$$

Let $\mathcal{D}$ be the uniform distribution over $\{u^{(1)}, \ldots, u^{(m)}, 0^n\}$. Note that given any disjunction hypothesis h for f that achieves error $< 1/(m+1)$ under $\mathcal{D}$, the variables in h must constitute a set cover for $\mathcal{S}$.

To see why this reduction, and reductions like it, do not extend to the setting of queries, we first observe that this specific target function can be easily learned with queries, simply by querying f on all strings of Hamming weight 1. More generally and crucially, we note that the target function is defined by the optimal solution to the SetCover instance. While this is a very natural strategy (and indeed many other hardness results for learning employ such a strategy), for any such reduction it seems challenging to provide query access to the target function without having to solve the SetCover instance, which would of course render the reduction inefficient. (Beyond the issue of queries, this reduction is also limited to the inverse-polynomial error regime and does not rule out learners that are allowed larger error.) While this reduction is for the hardness of properly learning disjunctions, all aforementioned hardness results for decision trees use it as their starting point and suffer from the same limitations.

How our approach differs. Departing from these works, we design a reduction where the *definition* of our target function does not depend on the solution to a computationally hard problem—which allows us to efficiently provide the learner query access to it—and only its *decision tree complexity* scales with the quality of the solution; see Remark 2. Our resulting reduction is quite a bit more elaborate than those for the setting of random examples.

3.2.2 Overview of our proof and techniques

We prove Theorem 15 by reducing from VertexCover: we design an efficient mapping from graphs G to functions f where the decision tree complexity of f reflects the vertex cover complexity of G. The properties of this mapping that we require our application to learning are somewhat subtle to state, so we describe and motivate them incrementally.

The core reduction

Our starting point is a reduction with the following basic properties:

The core reduction

- Yes case: If G has a small vertex cover, then f has small decision tree complexity.
- No case: If G requires a large vertex cover, then f has large decision tree complexity.

We sketch the main ideas behind this core reduction. For an n-vertex graph G, we consider its *edge indicator function* $\textsc{IsEdge}_G : \{0,1\}^n \to \{0,1\}$. An input $v = (v_1, \ldots, v_n) \in \{0,1\}^n$ to $\textsc{IsEdge}_G$

is viewed as specifying the presence or absence of each vertex $v_1, \ldots, v_n \in V$, and $\textsc{IsEdge}_G(v) = 1$ iff v specifies the presence of exactly the two endpoints of some edge of G. More formally:

Definition 6 ($\textsc{IsEdge}_G$)**.** *Let $G = (V, E)$ be an n-vertex graph. For an edge $e = \{v_i, v_j\} \in E$, we write $\mathrm{Ind}[e] \in \{0,1\}^n$ to denote its indicator string:*

$$\mathrm{Ind}[e]_k = \begin{cases} 1 & \text{if } k \in \{i, j\} \\ 0 & \text{otherwise.} \end{cases}$$

The edge indicator function of G is the function $\textsc{IsEdge}_G : \{0,1\}^n \to \{0,1\}$,

$$\textsc{IsEdge}_G(v_1, \ldots, v_n) = \begin{cases} 1 & (v_1, \ldots, v_n) = \mathrm{Ind}[e] \text{ for some } e \in E \\ 0 & \text{otherwise.} \end{cases}$$

When G is clear from context, we drop the subscript and simply write $\textsc{IsEdge}$.

Warmup. We first prove:

Claim 3.2.1 (Decision tree complexity of $\textsc{IsEdge}$)**.** *Let G be an n-vertex m-edge graph.*

- *Yes case: If G has a vertex cover of size $\leq k$, then there is a decision tree T for $\textsc{IsEdge}_G$ of size*
$$|T| \leq k + m + mn.$$
- *No case: If G requires a vertex cover of size $\geq k'$, then any decision tree T for $\textsc{IsEdge}_G$ must have size*
$$|T| \geq k' + m.$$

As stated, Claim 3.2.1 is not useful since the upper bound of the Yes case is much larger than the lower bound of the No case, owing to the additional additive factor of mn. More precisely, we need these bounds to satisfy:

$$\text{If } k' = (1+\delta)k \text{ then (Upper bound of Yes case)} < \text{(Lower bound of No case)} \qquad (\star)$$

in order to invoke the NP-hardness of $(1+\delta)$-approximating $\textsc{VertexCover}$.

Amplification. We therefore consider an "amplified" version of $\textsc{IsEdge}_G$,

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

and prove:

Theorem 16 (Decision tree complexity of ℓ-IsEdge)**.** *Let G be an n-vertex m-edge graph and $\ell \in \mathbb{N}$.*

- *Yes case: If G has a vertex cover of size $\leq k$, then there is a decision tree T for ℓ-IsEdge$_G$ of size*
$$|T| \leq (\ell + 1) \cdot (k + m) + mn.$$
- *No case: If G requires a vertex cover of size $\geq k'$, then any decision tree T for ℓ-IsEdge$_G$ must have size*
$$|T| \geq (\ell + 1) \cdot (k' + m).$$

We point out two properties of Theorem 16 that will be important for us:

Remark 1 (Asymmetric amplification in the Yes case)**.** Comparing Claim 3.2.1 and Theorem 16, we see that in No case, the entire lower bound of $k' + m$ is amplified by a factor of $\ell + 1$. On the other hand, in the Yes case only the $k + m$ factor—and crucially, *not* the mn factor—is amplified by a factor of $\ell + 1$. This is important as it allows us to choose ℓ to be sufficiently large to make the mn factor negligible, thereby having our bounds satisfy the sought-for property ($\star$).

Remark 2 (Efficiently providing query access to ℓ-IsEdge$_G$)**.** We defer the definition of ℓ-IsEdge$_G$ to Section 3.4.2 but mention here that (i) it will be the hard target function in our proof of Theorem 15; and (ii) just like the unamplified IsEdge$_G$ function—and *unlike* the SetCover-based target function described in Section 3.2.1— its definition will depend only on the edges in G and not its optimal vertex cover. This is crucial as it allows us to efficiently provide the learner query access to its values in our reduction without having to solve VertexCover. Circling back to our discussion in Section 3.2.1, this is a key qualitative difference between our reduction and previous reductions for the setting of random examples.

Hardness distillation

Theorem 16 already allows us to recover, with a markedly simpler proof, the best known hardness of approximation result [ZB00, Sie08] for decision tree minimization. However, it does not yet have any implications for learning since the No case only states that any decision tree that computes f *exactly* must have large size, and does not rule out the possibility that f can be well-approximated by a small decision tree.

We therefore strengthen the No case via a process that we call *hardness distillation*: we identify a small set of inputs $D \subseteq \{0,1\}^n$, which we call a *coreset*, that is responsible for f's large decision tree complexity.

The core reduction with hardness distillation

- Yes case: If G has a small vertex cover, then f has small decision tree complexity.
- No case: If G requires a large vertex cover, then there is a small set $D \subseteq \{0,1\}^n$ such that any decision tree that agrees with f on D must be large.

Such a reduction yields the NP-hardness of learning decision trees to error $< 1/|D|$, which motivates the problem of constructing coresets that are as small as possible. Our coreset will have size $\text{poly}(n)$, and therefore we get the hardness of learning to inverse-polynomial error. (In the next subsection we describe a further extension of this technique that establishes constant-error hardness.)

Hardness distillation via certificate complexity and relevant variables. We give a general method for identifying a small coreset that witnesses the large decision tree complexity of a function f. At a high level, there are two main components to this coreset:

1. A set of inputs D_1 that ensures that any decision tree T that agrees with f on D_1 must have a long path π, one of length at least s_1.
2. Another set of inputs D_2 that ensures that the at-least-s_1 many disjoint subtrees that branch off of π must have sizes that sum up to at least s_2.

See Figure 3.6 for an illustration. Together, D_1 and D_2 form a coreset witnessing the fact that f has decision tree complexity at least $s_1 + s_2$. To formalize this approach we rely on generalizations of two notions of boolean function complexity, namely certificate complexity and the relevance of variables, from the setting of total functions to partial functions. More formally, the two components of our method are as follows:

1. If there is an input $x^\star \in D_1$ such that the certificate complexity of f on $x^\star$ relative to D_1 is at least s_1, then any decision tree T that agrees with f on D_1 must have a long path π of length at least s_1.
2. This path π induces at least s_1 many subfunctions of f, corresponding to f restricted by paths that diverge from π at each of π's at-least-s_1 many nodes. If the number of variables of these subfunctions that are relevant relative to D_2 is at least s_2, then the at-least-s_1 many disjoint subtrees that branch off π must have sizes that sum up to at least s_2.

Hardness for constant error

To obtain hardness even against algorithms that are allowed constant error, we further improve the No case as follows:

The reduction for constant-error hardness

- Yes case: If G has a small vertex cover, then f has small decision tree complexity.
- No case: If G requires a large vertex cover, then there is a set $D \subseteq \{0,1\}^n$, a distributuon $\mathcal{D}$ over D, and a constant $\varepsilon > 0$ such that any decision tree that agrees with f with probability $\geq 1 - \varepsilon$ over $\boldsymbol{x} \sim \mathcal{D}$ must be large.

The key new ingredient in this final reduction is the hardness of α-*Partial* VertexCover, a relaxed version of VertexCover where the goal is to find a set of vertices that cover a $1 - \alpha$ fraction of vertices. We show that α-PartialVertexCover inherits its hardness of approximation from VertexCover itself:

Claim 3.2.2 (Hardness of α-PartialVertexCover)**.** *There are constants $\alpha \in (0,1)$ and $\delta > 0$ such that if α-PartialVertexCover on constant-degree, n-vertex graphs can be approximated to within a factor of $1+\delta$ in time $t(n)$, then* SAT *can be solved in time $t(n \cdot \mathrm{polylog}(n))$.*

This is thanks to the fact that VertexCover is hard the approximate even for constant-degree graphs, which in turn follows from the PCP Theorem.

With Claim 3.2.2 in hand, Theorem 15 then follows by appropriately robustifying the other machinery described in this section.

3.3 Preliminaries

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 $x^\star_i = 0$.

We identify a depth-d, non-terminal path π in a decision tree with a tuple 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. Paths naturally correspond to restrictions by forgetting their ordering. Therefore, we also write f_π to denote the restriction of f by $\{\ell_1, \ell_2, \ldots, \ell_d\}$.

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.

PCPs and Max-3Sat**.** For this work, we are interested in reductions from SAT. Our techniques will rely on the hardness of approximation and we therefore need a reduction from SAT to approximating Max-3Sat. The most efficient reduction exploits quasilinear PCPs:

Theorem 17 (Hardness of approximating Max-3Sat via quasilinear PCPs [Din06, BSS08])**.** *There is a constant $c \in (0,1)$ and a polynomial-time reduction that takes a 3CNF formula φ with m clauses and produces a 3CNF formula $\varphi^\star$ with $O(m \cdot \text{polylog}(m))$ clauses satisfying*

- *if φ is satisfiable then $\varphi^\star$ is satisfiable;*
- *if φ is unsatisfiable then no assignment satisfies a c-fraction of clauses of $\varphi^\star$.*

3.3.1 Hardness of 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 $\text{VC}(G) \in \mathbb{N}$ to denote the size of the smallest vertex cover. See Figure 3.1 for an example of a vertex cover. The VertexCover problem is to decide whether a graph contains a vertex cover of size-k, i.e. to decide if $\text{VC}(G) \leq k$. We consider the more general *gapped* vertex problem where the problem is to decide whether a graph has a small vertex cover or requires large vertex cover. Specifically we write (k, k')-VertexCover for the problem of deciding whether a graph contains a vertex cover of size-k or every vertex cover has size at least k'. This gapped problem is equivalent to the problem of *approximating* vertex cover. There is a polynomial-time greedy algorithm for vertex cover that approximates it within a factor of 2, i.e. solves $(k, 2k)$-VertexCover in polynomial-time.

Constant factor hardness of VertexCover is known, even for bounded degree graphs (graphs whose degree is bounded by some universal constant). Papadimitriou and Yannakakis in [PY91] give an approximation preserving reduction from Max-3Sat to VertexCover on constant-degree graphs. The PCP theorem [AS01, ALM+05] implies NP-hardness of approximating Max-3Sat and therefore, combined with the reduction in [PY91], implies hardness of approximating VertexCover on constant-degree graphs. (For a further discussion and history of these results, see the survey by Trevisan [Tre14].)

Theorem 18 (Hardness of approximating VertexCover)**.** *There are constants $\delta > 0$ and $d \in \mathbb{N}$ such that if $(k, (1+\delta) \cdot k)$-VertexCover on n-vertex degree-d graphs can be solved in time $t(n)$, then* SAT *can be solved in time $t(n \cdot \text{polylog}(n))$.*

This hardness follows from Theorem 17 and the reduction in [PY91]. The $n \cdot \text{polylog}(n)$ factor originates from Theorem 17.

The fact that Theorem 18 holds for constant degree graphs will be essential for our lower bound because it allows us to assume that k is large: $\text{VC}(G) = \Theta(m)$.

Fact 3.3.1 (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.

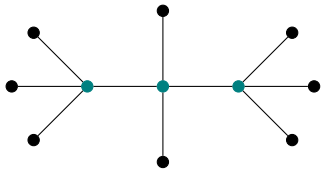


Figure 3.1: A graph $G = (V, E)$ with 10 edges having $\mathrm{VC}(G) = 3$. The unique vertex cover of size 3 is highlighted in teal.

3.4 A reduction from VertexCover to Decision Tree Minimization

3.4.1 Intuition and warmup: the $\textsc{IsEdge}_G$ function

In this section we prove Claim 3.2.1, which serves as a warmup for our core reduction, Theorem 16. We first introduce a few notions (and notation) that will be useful throughout the rest of the paper.

Useful notions and notation: edge partitions and divergent path prefixes

Edge partitions induced by decision trees for IsEdge. We will make use of the notion of a *restricted edge neighborhood* and a *restricted vertex neighborhood*. Specifically, we will be interested in the edges incident to a particular vertex which do *not* contain certain vertices.

Definition 7 (Restricted edge and vertex neighborhood)**.** *For a graph $G = (V, E)$, the edge neighborhood of $v_{i_\kappa} \in V$ restricted by $v_{i_1}, \ldots, v_{i_{\kappa-1}}$, denoted $E(v_{i_\kappa}; v_{i_1}, \ldots, v_{i_{\kappa-1}})$, is the set of edges containing v_{i_κ} but not any of $v_{i_1}, \ldots, v_{i_{\kappa-1}}$:*

$$E(v_{i_\kappa}; v_{i_1}, \ldots, v_{i_{\kappa-1}}) \coloneqq \{e \in E \mid v_{i_\kappa} \in e \text{ and } v_{i_1}, \ldots, v_{i_{\kappa-1}} \notin e\}.$$

The vertex neighborhood of v_{i_κ} restricted by $v_{i_1}, \ldots, v_{i_{\kappa-1}}$, denoted $V(v_{i_\kappa}; v_{i_1}, \ldots, v_{i_{\kappa-1}})$, is the set of neighbors of v_{i_κ} excluding the vertices $v_{i_1}, \ldots, v_{i_{\kappa-1}}$:

$$V(v_{i_\kappa}; v_{i_1}, \ldots, v_{i_{\kappa-1}}) \coloneqq \left\{v \in V \mid \{v_{i_\kappa}, v\} \in E \text{ and } v \neq v_{i_1}, \ldots, v_{i_{\kappa-1}}\right\}.$$

Often when a tuple of vertices $(v_{i_1}, \ldots, v_{i_k})$ is understood from context, we will use the shorthand notation $E_\kappa = E(v_{i_\kappa}; v_{i_1}, \ldots, v_{i_{\kappa-1}})$ for $\kappa = 1, \ldots, k$ and likewise for V_κ. Restricted edge and vertex

neighborhoods are closely related to each other, and each can be defined in terms of the other:

$$E_\kappa = \{\{v_{i_\kappa}, v\} \mid v \in V_\kappa\} \quad \text{and} \quad V_\kappa = \{v \mid \{v_{i_\kappa}, v\} \in E_\kappa\}.$$

Given a vertex cover $\{v_{i_1}, \ldots, v_{i_k}\}$, the sets $\{E_\kappa\}_{\kappa \in [k]}$ form a partition of the edge set E. Indeed,

$$\bigcup_{\kappa \in [k]} E_\kappa = E$$

since every edge in G is incident to some vertex v_{i_κ}. Also, the sets E_κ are disjoint since each E_κ excludes the edges already covered by the previous $E_1, \ldots, E_{\kappa-1}$ sets. In fact, the converse also holds. If $C = \{v_{i_1}, \ldots, v_{i_k}\}$ are vertices such that E_κ partition the edge set then C must form a vertex cover: every edge $e \in E$ is in some partition E_κ and so v_{i_κ} covers e.

Fact 3.4.1. *Let $C = \{v_{i_1}, \ldots, v_{i_k}\}$ be a subset of vertices of a graph G and $E_\kappa \coloneqq E(v_{i_\kappa}; v_{i_1}, \ldots, v_{i_{\kappa-1}})$ for $\kappa \in [k]$. Then C forms a vertex cover of G if and only if $\{E_\kappa\}_{\kappa \in [k]}$ form a partition of E.*

A key property of the IsEdge$_G$ function is that every decision tree for it induces such an edge partition in the following way. Every decision tree for IsEdge$_G$ has a path π in it whose path variables form a vertex cover. This vertex cover induces a partition of the edges of G. Each part of the partition corresponds to a unique variable in this decision tree path. This correspondence will be important for lower bounding the size of the decision tree in the case when G requires large vertex covers. To describe this correspondence, it will be useful for us to have the following notation for a path that diverges from from π at a particular point and then stops.

Definition 8 (Divergent path prefix; see Figure 3.2)**.** *For a path π, the path $\pi|_{\oplus\kappa}$ denotes the path which follows π for the first $\kappa - 1$ queries, flips the κth query, then terminates:*

$$\pi|_{\oplus\kappa} \coloneqq \left(\pi(1), \ldots, \pi(\kappa-1), \overline{\pi(\kappa)}\right).$$

If π is the path corresponding to a vertex cover, then $\pi|_{\oplus\kappa}$ corresponds to the path followed by edges in E_κ (here we are conflating edges and edge indicator strings).

Proof of Claim 3.2.1

Proof of the Yes case. Let $C = \{v_{i_1}, \ldots, v_{i_k}\}$ be a vertex cover for G. The leftmost branch π of our decision tree queries these vertices successively and terminates with a 0-leaf. These are the vertices colored blue in Figure 3.3.

We move on to describing each of the subtrees branching off of π. More formally, for each $\kappa \in [k]$, we describe the subtree rooted at the end of $\pi|_{\oplus\kappa}$ (i.e. the subtree that is the 1-successor of v_{i_k}). At this point T "knows" that v_{i_k} is set to 1. For IsEdge to output 1, exactly one of v_{i_κ}'s neighbors

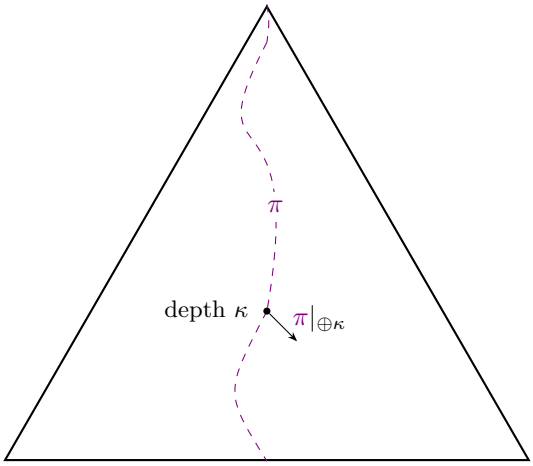


Figure 3.2: Illustration of a divergent path prefix. The root-to-leaf path π is illustrated in purple. At depth κ the path $\pi|_\kappa$ diverges and terminates.

must also be set to 1, and all $n-2$ other vertices must be set to 0 (i.e. these are precisely the inputs $\text{Ind}[e]$ for $e \in E_\kappa$). Therefore T queries all $v \in V_\kappa$ (i.e. the neighbors of v_{i_κ} that have not already been queried along π), testing to see whether any of them are 1, and terminates with a 0-leaf if they are all set to 0. These are the vertices colored teal in Figure 3.3.

Finally, we describe the subtree that is the 1-successor of each $v \in V_\kappa$. At this point T knows that v_{i_κ} and this neighbor v are both set to 1, and it remains only to check that all other vertices are set to 0 before outputting 1: it queries all $n-2$ vertices in V that are not v or v_{i_κ} and outputs 1 iff all of them are set to 0. These are the vertices colored orange in Figure 3.3.

We complete the proof by bounding the size of T. Its leftmost branch has size k (the blue vertices). By Fact 3.4.1, querying all $v \in V_\kappa$ for $\kappa \in [k]$ results in an additional $\sum_\kappa |V_\kappa| = \sum_\kappa |E_\kappa| = m$ internal nodes (the teal vertices). After each of these m internal nodes, we query $n-2$ more vertices, resulting in an additional $m(n-2) < mn$ internal nodes (the orange vertices). Thus, the total size of T is at most $k + m + mn$. □

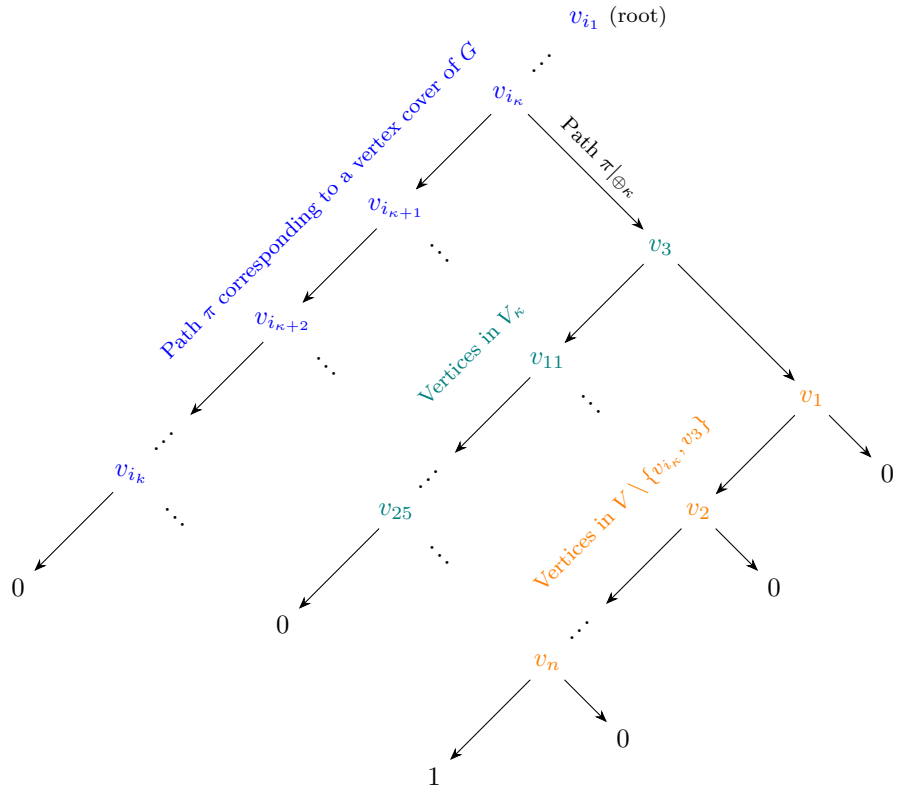


Figure 3.3: An illustration of the proof of the Yes case of Claim 3.2.1. Given a vertex cover $C = \{v_{i_1}, \ldots, v_{i_k}\}$ of G, our decision tree for IsEdge queries C among the leftmost branch (colored blue in the figure). If some vertex $v_{i_\kappa} \in C$ is set to 1 then the decision tree queries all vertices in V_κ (colored teal). Once some $v \in V_\kappa$ is set to 1, the decision tree queries the remaining unqueried vertices to ensure that they are set to 0 (colored orange) before outputting 1.

We proceed to a proof of the lower bound.

Proof of the No case. Our proof consists of two parts: (1) proving that the leftmost branch of T must be a vertex cover and therefore has size at least k' and (2) showing that the rest of the tree has size at least m. See Figure 3.4 for an illustration.

1. *Leftmost branch must be a vertex cover.* Let π be the leftmost branch of T and suppose for contradiction that the vertices queried along π do not form a vertex cover for G. This means that there is some edge $e \in E$ that is not queried along π, and hence both $\mathrm{Ind}[e]$ and 0^n will follow π and reach the same leaf. Since $\textsc{IsEdge}(0^n) = 0 \neq 1 = \textsc{IsEdge}(\mathrm{Ind}[e])$, this is a contradiction.

2. *Rest of the tree has at least m nodes.* Let us order the vertices of π from root downwards as $v_{i_1}, \ldots, v_{i_{|\pi|}}$. For each $\kappa \in [|\pi|]$, we consider the subtree T_κ that is the 1-successor of

v_κ. Consider $e \in E_\kappa$ and suppose $e = (v_{i_\kappa}, v)$. By the definition of E_κ, the endpoint v has not yet been queried when $\mathrm{Ind}[e]$ enters T_κ. Thus, T_κ must query v, since otherwise T cannot distinguish between $\mathrm{Ind}[e]$ and $\mathrm{Ind}[e]^{\oplus v}$ (note that $\textsc{IsEdge}(\mathrm{Ind}[e]) = 1 \neq 0 = \textsc{IsEdge}(\mathrm{Ind}[e]^{\oplus v})$). Further, all $e \in E_\kappa$ will have distinct second endpoints that T_κ must query (since if not, then they would share both their endpoints and be the exact same edge). In other words, we have argued that T_κ must query all the vertices in V_κ.

Since the sets E_κ for $\kappa \in [|\pi|]$ partition the edges (Fact 3.4.1), we have that all these disjoint subtrees $T_1, \ldots, T_{|\pi|}$ taken together must query at least $\sum_\kappa |V_\kappa| = |E_\kappa| = |E| = m$ additional vertices.

Combining the two claims above we show shown that $|T| \geq k' + m$ and the proof is complete. □

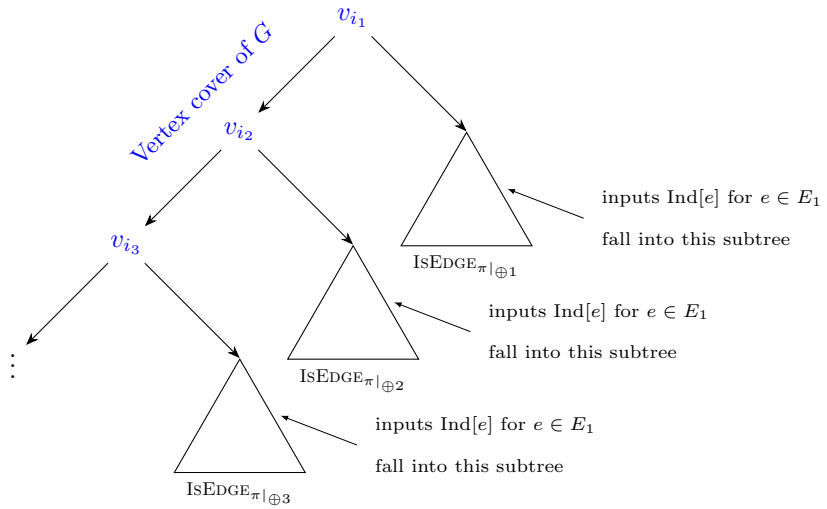


Figure 3.4: An illustration of the No case of Claim 3.2.1. Given any decision tree T computing IsEdge, the leftmost branch π must form a vertex cover of G. Furthermore, for each vertex v_{i_κ} queried along π, the subtree T_κ branching off of π at v_{i_κ} must query all the vertices in V_κ. The size of T is therefore at least $k' + \sum_\kappa |V_\kappa| = k' + m$.

3.4.2 ℓ-IsEdge: an amplified version of IsEdge

Definition 9 (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*

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

defined as follows: ℓ-$\textsc{IsEdge}_G(v^{(0)}, v^{(1)}, \ldots, v^{(\ell)}) = 1$ iff

1. $\textsc{IsEdge}_G(v^{(0)}) = 1$ *(i.e. $v^{(0)} = \mathrm{Ind}[e]$ for some $e \in E$), and*

2. $v_i^{(1)} = \cdots = v_i^{(\ell)} = 1$ *for all* $i \in [n]$ *such that* $v_i^{(0)} = 1$.

Notation and terminology. When G is clear from context, we drop the subscript and just write ℓ-IsEdge. We also use $N \coloneqq n + n\ell$ to denote the number of inputs to ℓ-IsEdge. We refer to $v_1^{(0)}, \ldots, v_n^{(0)}$ as the *original* variables. As in the nonamplified IsEdge function, there is a natural correspondence between these original variables and the vertices $V = \{v_1, \ldots, v_n\}$ of G. For each original variable $v_i^{(0)}$, we refer to $v_i^{(1)}, \ldots, v_i^{(\ell)}$ as its *duplicated* variables and write

$$\textsc{Dup}(v_i) \coloneqq \left\{v_i^{(1)}, \ldots, v_i^{(\ell)}\right\}.$$

We write ℓ-Ind$[e] \in (\{0,1\}^n)^{\ell+1}$ to denote the string $(\text{Ind}[e], \ldots, \text{Ind}[e])$. Note that we have ℓ-IsEdge$(\ell\text{-Ind}[e]) = 1$ and these are the 1-inputs of minimum Hamming weight.

Asymmetries in the definition of ℓ-IsEdge. We note two sources of asymmetry in the definition of ℓ-IsEdge, both of which are crucial for Theorem 16 (specifically, Remark 1) to hold. First, the original variables play a distinct role from the duplicated ones: for ℓ-IsEdge to output 1, the original variables have to agree with an edge indicator but the duplicated variables do not. Second, there is also an asymmetry between 1- and 0-coordinates: for ℓ-IsEdge to output 1, the duplicated variables have to be set to 1 whenever the original variables are set to 1, but the same is not true for the 0-coordinates.

Proof of Theorem 16

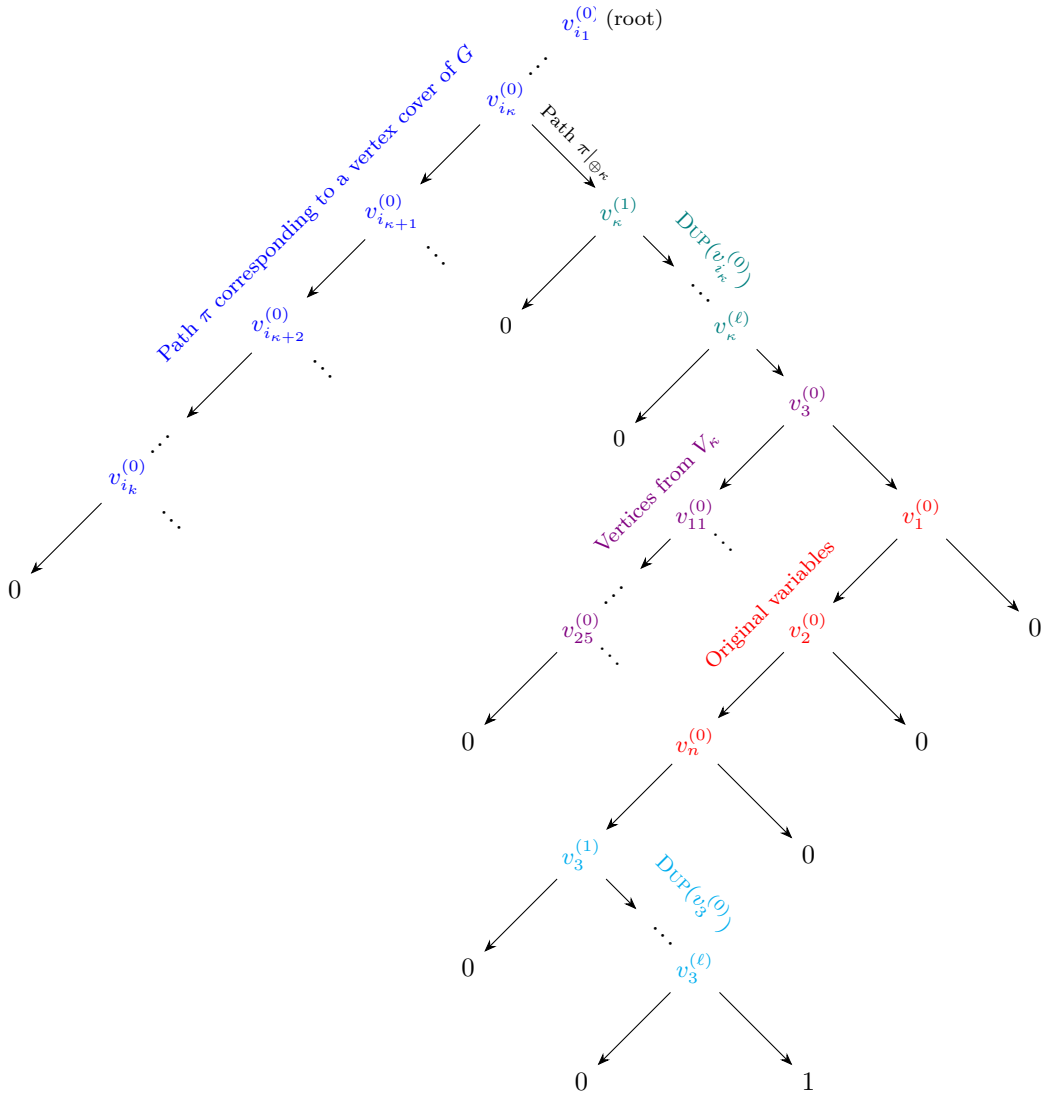


Figure 3.5: An illustration of the proof of the Yes case of Theorem 16. Given a vertex cover $C = \{v_{i_1}, \ldots, v_{i_k}\}$ of G, our decision tree for ℓ-IsEdge queries the original variables corresponding to C among the leftmost branch (colored blue in the figure). If some vertex $v_{i_\kappa}^{(0)} \in C$ is set to 1 then the decision tree queries all vertices in $\textsc{Dup}(v_{i_\kappa}^{(0)})$ (colored in teal). If all of these are 1, then it proceeds to compute the appropriate IsEdge subfunction on the remaining vertices.

Proof of the Yes case. The construction is a slight extension of our tree for IsEdge that we constructed for the Yes case of Claim 3.2.1. See Figure 3.5 for an illustration of this construction. Let $C = \{v_{i_1}, \ldots, v_{i_k}\}$ be a vertex cover of G. Similar to before, the leftmost branch π of our tree T

queries the original variables $v_{i_1}^{(0)}, \ldots, v_{i_k}^{(0)}$ corresponding to these vertices and terminates with a 0-leaf.

We now describe the subtree T_κ that is the 1-successor of $v_{i_\kappa}^{(0)}$ for $\kappa \in [k]$. It first checks if the duplicated variables of v_{i_κ} are all set to 1, since that is a necessary criterion for ℓ-IsEdge to output 1: it queries all ℓ variables in $\textsc{Dup}(v_{i_\kappa})$ and outputs 0 once any of them are set to 0. If all of them are indeed set to 1, then for T_κ to output 1 it must check that (i) there is a neighbor v of v_{i_k} whose original variable is set to 1, (ii) all the other original variables are set to 0, and (iii) the duplicated variables of v are set to 1.

For (i) and (ii), T_κ queries the remaining original variables in a manner identical to the tree for IsEdge. Briefly restating that construction, it verifies Condition (i) by querying the original variables of $v \in V_\kappa$, testing to see of any of them are set to 1. If none of them are set to 1, it outputs 0. Otherwise, once the original variable of some $v \in V_\kappa$ is set to 1, the tree T_κ moves on to verifying Condition (ii): it queries the original variables of all $n-2$ vertices in $V \setminus \{v_{i_\kappa}, v\}$ and outputs 0 once any of them are set to 1. If all of them are indeed set to 0, it moves on to verifying Condition (iii). It queries all ℓ variables in $\textsc{Dup}(v)$ and outputs 1 iff all of them are set to 1.

We now bound the size of this tree. The portion of it that is identical to the tree for IsEdge will have size at most $k + m + mn$, as proved in Claim 3.2.1. We incur an additional $k\ell$ nodes to query the duplicate variables for the vertex cover: ℓ duplicate variables for each vertex in the size-k vertex cover. Finally, since we additionally query the ℓ many duplicate variables for each $v \in V_\kappa$ in the subtree T_κ, we incur another additional $\ell \sum_\kappa |V_\kappa| = \ell \sum_\kappa |E_\kappa| = \ell m$ many nodes. In total, this results in a tree of size

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

We now prove the lower bound.

Proof of No case. Just as in the proof of Claim 3.2.1, we divide our proof into two parts. We show that (1) the leftmost branch of any decision tree for ℓ-IsEdge must correspond to a vertex cover and hence has size at least k', and (2) the rest of the tree must have size at least $\ell k' + (\ell + 1)m$.

1. *Leftmost branch must be a vertex cover.* Let π be the leftmost branch of T and $v_{i_1}^{(j_1)}, \ldots, v_{i_{|\pi|}}^{(j_{|\pi|})}$ be the variables queried along π. We claim that the corresponding vertices $v_{i_1}, \ldots, v_{i_{|\pi|}} \in V$ must form a vertex cover for G. Suppose for contradiction that they do not. This means that there is some edge $e \in E$ such that neither the original nor duplicated variables of e's endpoints are queried along π. Therefore both ℓ-Ind$[e]$ and 0^N will follow π and reach the same leaf. Since $\textsc{IsEdge}(0^N) = 0 \neq 1 = \ell\text{-}\textsc{IsEdge}(\ell\text{-Ind}[e])$, this is a contradiction.

2. *Rest of the tree has at least $\ell k' + (\ell+1)m$ nodes.* Order the vertices of π from root downwards as $v_{i_1}, \ldots, v_{i_{|\pi|}}$. We will consider only the indices $\kappa \in [|\pi|]$ such that E_κ is nonempty, noting

that the vertices corresponding to these indices still form a vertex cover, and hence there are at least k' such indices. Fix such a κ and consider the subtree T_κ that is the 1-successor of $v_{i_\kappa}^{(j_\kappa)}$. Let $e = (v_{i_\kappa}, v) \in E_\kappa$. We argue that v_{i_κ} is responsible for ℓ additional queries within T_κ, and v for $\ell + 1$ additional ones. For the former claim, let $j \in \{0, \ldots, \ell\} \setminus \{j_\kappa\}$. By the definition of E_κ, the variable $v_{i_\kappa}^{(j)}$ has not yet been queried when ℓ-Ind$[e]$ enters T_κ. Since

$$\ell\text{-IsEdge}(\ell\text{-Ind}[e]) = 1 \neq 0 = \ell\text{-IsEdge}(\ell\text{-Ind}[e]^{\oplus v_{i_\kappa}^{(j)}}),$$

it follows that $v_{i_\kappa}^{(j)}$ must be queried within T_κ. Similarly, the latter claim follows from the fact that T_κ must query the original and all the duplicated variables of v, a total of $\ell + 1$ many variables. This latter claim holds for all endpoints of edges $e \in E_\kappa$ (i.e. the vertices $v \in V_\kappa$), so in total T_κ must contain at least $\ell + (\ell+1)|V_\kappa|$ nodes. Summing over all $\kappa \in [|\pi|]$ such that E_κ is nonempty and applying Fact 3.4.1, we get that the disjoint subtrees $T_1, \ldots, T_{|\pi|}$ must query at least

$$\sum_{\kappa\colon E_\kappa \neq \varnothing} \ell + (\ell+1)|E_\kappa| = \ell k' + (\ell+1)m$$

many variables.

Combining the two claims above we have shown that

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

and the proof is complete. □

3.4.3 Hardness of decision tree minimization

DT-Min: Given a decision tree $T^\star : \{0,1\}^n \to \{0,1\}$, construct a minimum-size decision tree T such that $T \equiv T^\star$ (i.e. $T(x) = T^\star(x)$ for all $x \in \{0,1\}^n$).

This problem of decision tree minimization was first shown to be NP-hard by Zantema and Bodlaender [ZB00]. That result was subsequently improved by Sieling [Sie08] who showed that the problem is even NP-hard to approximate. Using Theorem 16 we recover this hardness of approximation. We begin by observing that our proofs of the Yes and No cases of Theorem 16 are algorithmic in the following sense:

- In the Yes case, we showed that given a graph G and a size-k vertex cover, the tree T for ℓ-IsEdge of size $(\ell+1)\cdot(k+m) + mn$ can be constructed in poly(ℓ, n) time.
- In the No case, we showed that given a size-s' tree T for ℓ-IsEdge, a size-k' vertex cover for G satisfying $(\ell+1)\cdot(k'+m) \leq s'$ can be constructed in poly(ℓ, n) time.

With these observations in hand, we are now ready to recover [Sie08]'s result.

Lemma 3.4.2 (A reduction from VertexCover to DT-Min)**.** *There is a polynomial-time reduction that takes a degree-d, n-vertex, m-edge graph G and produces a decision tree $T^\star$ such that the following holds. Given any tree T such that $T \equiv T^\star$ and whose size is within a $(1+\delta)$ factor of the optimal for $T^\star$, one can construct in polynomial time a size-k' vertex cover of G satisfying $k' \leq (1+\delta') \cdot \mathrm{VC}(G)$ where $\delta' = O(d\delta)$.*

Proof. Let $\ell := 2mn$. We begin by applying the Yes case of Theorem 16 to G with the trivial vertex cover of all n vertices to obtain a decision tree $T^\star$ for ℓ-IsEdge of size

$$(\ell+1)\cdot(n+m)+mn = (2mn+1)\cdot(n+m)+mn.$$

As observed above, our proof of Theorem 16 shows that $T^\star$ can be constructed from G in $\mathrm{poly}(n)$ time. This tree $T^\star$ will be the input to DT-Min in our reduction.

By the Yes case of Theorem 16 again, if $\mathrm{VC}(G) =: k$ then

$$\mathrm{DT}(\ell\text{-IsEdge}) \leq (\ell+1)(k+m)+mn =: s.$$

Suppose an algorithm for DT-Min returns a tree T for ℓ-IsEdge of size s' where $s' \leq (1+\delta)\cdot s$. We claim that we can then efficiently construct a vertex cover for G of size k' where $k' \leq (1+\delta')\cdot k$ and $\delta' = O(d\delta)$, thereby completing the reduction. Our proof of Theorem 16 shows that we can efficiently construct from T, in $\mathrm{poly}(n)$ time, a size-k' vertex cover satisfying:

$$(\ell+1)(k'+m) \leq s'.$$

The claim that $s' \leq (1+\delta)\cdot s$ is therefore equivalent to

$$(\ell+1)\cdot(k'+m) \leq (1+\delta)\cdot\big[(\ell+1)(k+m)+mn\big].$$

Rearranging the above, we get that

$$\begin{aligned}
k' &\leq (1+\delta)\cdot k + \delta m + \frac{(1+\delta)\cdot mn}{\ell+1} \\
&\leq (1+\delta)\cdot k + \delta kd + \frac{(1+\delta)\cdot mn}{\ell+1} && (m \leq kd \text{ by Fact 3.3.1}) \\
&< (1+\delta)\cdot k + \delta kd + 1 && (\text{Our choice of } \ell) \\
&< \big[1+\delta(d+2)\big]\cdot k
\end{aligned}$$

and the proof is complete. □

[Sie08]'s result now follows as an immediate consequence of Lemma 3.4.2 and the fact that

VertexCover is hard to approximate even for constant-degree graphs (Theorem 18):

Theorem 19 (Hardness of approximation for DT-Min [Sie08])**.** *There is a constant $\delta \in (0,1)$ such that if* DT-Min *can be approximated to within a factor of $1+\delta$ in polynomial-time, then* $\mathrm{P} = \mathrm{NP}$*.*

([Sie08] then amplifies this constant-factor inapproximability to a superconstant factor using an XOR lemma from [HJLT96]. We refer the interested reader to [Sie08] for the details of this step.)

In the next section, we strengthen [Sie08]'s result by showing that the same hardness holds even if the algorithm need only minimize T over a small set of input points rather than all of $\{0,1\}^n$.

3.5 Hardness distillation and learning consequence for small error

3.5.1 A general method for hardness distillation

For a function $f : \{0,1\}^n \to \{0,1\}$, the quantity $\mathrm{DT}(f)$ captures the complexity of computing f on *all* of its inputs. If $\mathrm{DT}(f)$ is large, then any small decision tree that tries to compute f must err on at least one point in $\{0,1\}^n$. For some f, it may be the case that, more specifically, there is a fixed set $D \subseteq \{0,1\}^n$ such that all small decision trees err on some point in D. The set D then captures or "distills" the hardness of f since any function g which agrees with f over the set D must also have large decision tree complexity. We call this set D a *coreset*.[1] Ultimately, our goal will be to identify explicitly coresets D which distill the hardness of the target function f. This way, any learner that learns f over the distribution $\mathrm{Uniform}(D)$ to error $< \frac{1}{|D|}$ has to output a decision tree whose size captures $\mathrm{DT}(f)$. Since the error scales with $\frac{1}{|D|}$, we have a vested interest in making D has small as possible so that we can tolerate large learning errors. In this section, we identify a general method for distilling the hardness of a function f into a coreset D. We start by generalizing certificate complexity and relevant variables with respect to fixed subsets D.

Certificate complexity with respect to a set of inputs. A certificate for $f : \{0,1\}^n \to \{0,1\}$ over a set of inputs $D \subseteq \{0,1\}^n$ on $x \in \{0,1\}^n$ is a restriction ρ consistent with x such that $f_\rho(y) = f_\rho(x)$ for all $y \in D$. The certificate complexity of x on f over D is the size of the smallest certificate of x on f over D.

One useful fact is that a decision tree path forms a certificates for the inputs that follow it.

Fact 3.5.1 (Decision tree paths are certificates)**.** *If a decision tree T computes $f : \{0,1\}^n \to \{0,1\}$ over $D \subseteq \{0,1\}^n$, then the path that an input $x \in D$ follows in T forms a certificate for f over D on x.*

[1] This naming convention is inspired by, though not formally related to, the notion of a coreset from the clustering literature.

Indeed, in the above, any root-to-leaf path π terminates in a leaf which implies f_π is a constant function over D. Any input $x \in D$ that follows π is consistent with it and so $f(x) = f_\pi(x) = f_\pi(y)$ for all $y \in D$.

Relevant variables with respect to a set of inputs. A variable $i \in [n]$ is said to be *relevant* for $f : \{0,1\}^n \to \{0,1\}$ over a set of inputs $D \subseteq \{0,1\}^n$ if there is some $x \in D$ such that $x^{\oplus i} \in D$ and $f(x) \neq f(x^{\oplus i})$. We write $\mathrm{Rel}(f; D) \in [n]$ for the number of relevant variables of f with respect to D. When referring to the number of relevant variables over the entire domain $D = \{0,1\}^n$, we drop D and simply write $\mathrm{Rel}(f)$. If $D \subseteq D'$, then every relevant variable for f over D is also relevant for f over D'. Therefore, $\mathrm{Rel}(f, D) \leq \mathrm{Rel}(f, D')$ and in particular $\mathrm{Rel}(f, D) \leq \mathrm{Rel}(f)$ for all D.

Decision tree complexity with respect to a set of inputs. The decision tree complexity of $f : \{0,1\}^n \to \{0,1\}$ over $D \subseteq \{0,1\}^n$ is the size of the smallest decision tree that computes f over D and is denoted $\mathrm{DT}(f, D)$. Any decision tree that computes f also computes f over D and so $\mathrm{DT}(f, D) \leq \mathrm{DT}(f)$.

We now state and prove the main result for this section.

Theorem 20 (Hardness distillation)**.** *Let $f : \{0,1\}^n \to \{0,1\}$ and $D \subseteq \{0,1\}^n$ be a set of inputs. Let $s_1 \in \mathbb{N}$ lower bound the certificate complexity of x on f over D. Let $s_2 \in \mathbb{N}$ satisfy*

$$\sum_{i=1}^{|\rho|} \mathrm{Rel}(f_{\pi|_{\oplus i}}; D) \geq s_2$$

for every certificate ρ for x on f over D and $\pi \in \mathrm{Perm}(\rho)$, a permutation of ρ. Then,

$$\mathrm{DT}(f, D) \geq s_1 + s_2.$$

If one can show for some D that the quantity $s_1 + s_2$ captures the decision tree complexity of f, then D is a good candidate for hardness distillation. Figure 3.6 illustrates some intuition for the quantity $\sum_{i=1}^{|\rho|} \mathrm{Rel}(f_{\pi|_{\oplus i}}; D)$ in Theorem 20. If $x \in D$, then any decision tree for f over D contains a certificate, ρ, for f on x. The depth-$|\rho|$ path followed by x induces an ordering over ρ and naturally yields $|\rho|$ disjoint subtrees, each of which hangs off the main path. The size of the main decision tree is lower bounded by the sizes of these subtrees plus the length of the path followed by x. The sizes of these subtrees can be lower bounded by the number of relevant variables of the corresponding subfunctions which then yields the desired lower bound.

Before proving Theorem 20, we establish a lemma stating that the number of relevant variables of disjoint subtrees of a decision tree lower bounds its size.

Lemma 3.5.2 (Relevant variables of disjoint subtrees lower bound decision tree size)**.** *Let T be a*

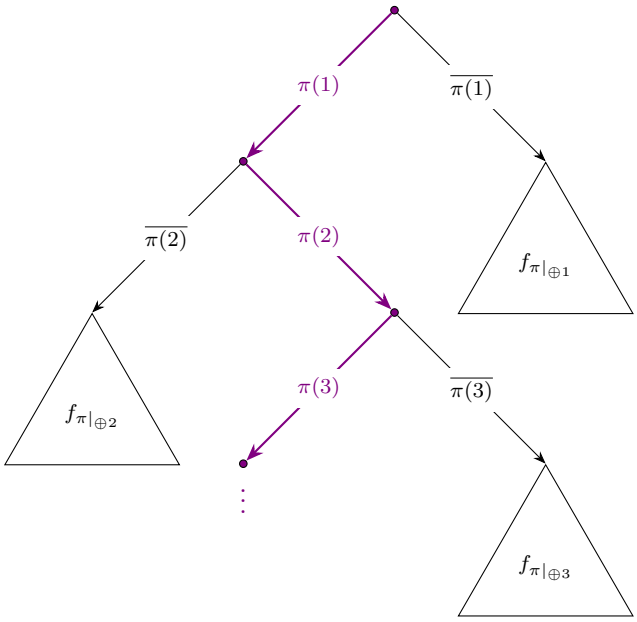


Figure 3.6: An illustration of hardness distillation for a function f. A path π through the decision tree is highlighted in purple. This path corresponds to an ordering of a certificate for some input x that follows this path. The subtrees hanging off the main path π compute the subfunctions $f_{\pi|_{\oplus i}}$ where $\pi|_{\oplus i}$ corresponds to the path leading to the root of the subtree. The sum of the number of relevant variables of these subfunctions plus the length of the path π lower bounds the overall size of the decision tree.

decision tree, and let $T_1, \ldots, T_d$ be disjoint subtrees of T. Then,

$$|T| \geq \sum_{i=1}^{d} \mathrm{Rel}(T_i).$$

Proof. If a variable x_j is *not* queried in the subtree T_i, then x_j cannot be relevant for the function T_i. Indeed, in this case, the leaf in T_i that any input x reaches is the same as the leaf that $x^{\oplus j}$ reaches. Therefore, every relevant variable of T_i is queried in the subtree. Since the subtrees $T_1, \ldots, T_d$ are disjoint, each relevant variable of T_i can be mapped to a *unique* internal node of T. It follows that

$$|T| \geq \sum_{i=1}^{d} |T_i| \geq \sum_{i=1}^{d} \mathrm{Rel}(T_i). \qquad \square$$

With this lemma in hand, we are able to prove Theorem 20.

Proof of Theorem 20. Let T be any decision tree computing f over D. We will show that $|T| \geq s_1 + s_2$. Let π be the path followed by $x \in D$ in T. By Fact 3.5.1, π is a certificate for f over D on x. Therefore $|\pi| \geq s_1$. Recall from Definition 8 that $\pi|_{\oplus i} = \left\{\pi(1), \ldots, \pi(i-1), \overline{\pi(i)}\right\}$ corresponds

to the depth i path in T that follows x to depth $i-1$ and then diverges from x on the ith variable queried. Let $T_{\pi|_{\oplus i}}$ denote the subfunction of T computed by the subtree rooted at the last variable queried in $\pi|_{\oplus i}$. Each $T_{\pi|_{\oplus i}}$ contributes $\mathrm{Rel}(T_{\pi|_{\oplus i}})$ many variables to the size of T by Lemma 3.5.2 and the path ρ itself contributes at least s_1 many variables since π is also disjoint from the subtrees. It follows that

$$\begin{aligned}
|T| &\geq |\pi| + \sum_{i=1}^{|\pi|} \mathrm{Rel}(T_{\pi|_{\oplus i}}) && \text{(Lemma 3.5.2)} \\
&\geq s_1 + \sum_{i=1}^{|\pi|} \mathrm{Rel}(T_{\pi|_{\oplus i}}; D) && \text{(Definition of Rel)} \\
&= s_1 + \sum_{i=1}^{|\pi|} \mathrm{Rel}(f_{\pi|_{\oplus i}}; D) && (T \text{ computes } f \text{ over } D) \\
&\geq s_1 + s_2. && \text{(Assumption from theorem statement)}
\end{aligned}$$

□

3.5.2 Warmup: hardness distillation for IsEdge

We start by applying the framework from Section 3.5.1 to the function IsEdge. The first step is to identify a small coreset D which captures decision tree size.

Definition 10 (Decision tree coreset of the IsEdge function)**.** *For an n-vertex graph G, the set $D_G \subseteq \{0,1\}^n$ consists of the points*

- *all edge indicators:* $\mathrm{Ind}[e] \in \{0,1\}^n$ *such that* $e \in E$*;*
- *all 1-coordinate perturbations of edge indicators:* $\mathrm{Ind}[e]^{\oplus i}$ *and* $\mathrm{Ind}[e]^{\oplus j}$ *for all* $e = \{v_i, v_j\} \in E$*;*
- *the all 0s inputs:* 0^n*.*

Example. See Figure 3.7 for an example of a graph G and the associated set of inputs D_G.

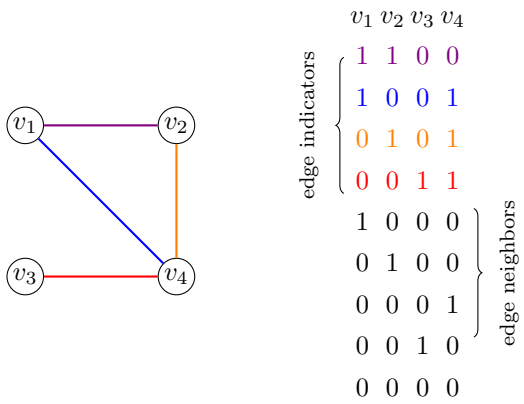


Figure 3.7: Example of a graph G on four vertices and the associated set of inputs $D_G \subseteq \{0,1\}^4$. Each row in the table corresponds to a data point in D_G. The first 4 rows correspond to the edges in G and are color coded to highlight which row corresponds to which edge. The next 4 rows correspond to 1-coordinate perturbations of the edge indicators, all of which are Hamming neighbors of edge indicators.

Recall from Claim 3.2.1 that $\mathrm{DT}(\textsc{IsEdge}_G) \geq k' + m$ where k' is the size of a vertex cover for G. The main claim of this section is that D_G "distills" this hardness factor of $\textsc{IsEdge}_G$. The upper bound from Claim 3.2.1 immediately applies to D_G. That is, $k + m + mn \geq \mathrm{DT}(\textsc{IsEdge}_G) \geq \mathrm{DT}(\textsc{IsEdge}_G, D_G)$. Therefore, the lower bound is all that remains for showing D_G is a good coreset.

Claim 3.5.3 (D_G is a decision tree coreset for IsEdge)**.** *Let G be an m-vertex graph. Then*

$$\mathrm{DT}(\textsc{IsEdge}, D_G) \geq k' + m$$

where k' is the size of a vertex cover for G.

Ultimately, we would like to prove Claim 3.5.3 by applying Theorem 20 where f is the function IsEdge, D is the set of inputs D_G, and x is the input 0^n. To this end, we first show that k' is a lower bound on the certificate complexity of 0^n on IsEdge over D_G. Then we prove a lemma showing that the number of edges in G lower bounds the number of relevant variables of subfunctions of IsEdge induced by certificates of 0^n.

Proposition 3.5.4 (Any certificate of 0^n contains a vertex cover)**.** *Let G be an n-vertex graph and let ρ be a certificate for* IsEdge *over D_G on 0^n. Then the variables in ρ form a vertex cover of G.*

Proof. If the variables in ρ do not cover some edge $e \in E$, then $\mathrm{Ind}[e] \in \{0,1\}^n$ is consistent with ρ and $\textsc{IsEdge}_G(0^n) = 0 \neq 1 = \textsc{IsEdge}_G(\mathrm{Ind}[e])$ implies ρ is not a certificate. Therefore, any certificate ρ must contain a vertex cover. □

Lemma 3.5.5 (Lower bounding the number of relevant variables of IsEdge subfunctions)**.** *Let G be an n-vertex graph, ρ a certificate for* $\textsc{IsEdge} : \{0,1\}^n \to \{0,1\}$ *over D_G, and $\pi = (\overline{v}_{i_1}, \dots, \overline{v}_{i_k}) \in \mathrm{Perm}(\rho)$ a permutation of ρ. Then*

$$\mathrm{Rel}(\textsc{IsEdge}_{\pi|_{\oplus\kappa}}; D_G) \geq |E(v_{i_\kappa}; v_{i_1}, \dots, v_{i_{\kappa-1}})|$$

for all $\kappa \in [k]$.

Proof. Let $\pi|_{\oplus\kappa}$ be as in the lemma statement and let $v \in V_\kappa = V(v_{i_\kappa}; v_{i_1}, \dots, v_{i_{\kappa-1}})$ be arbitrary (recall the definition of these quantities from Definitions 7 and 8). Let $e = (v_{i_\kappa}, v) \in E_\kappa = E(v_{i_\kappa}; v_{i_1}, \dots, v_{i_{\kappa-1}})$ be the edge containing v. The input $\mathrm{Ind}[e] \in D_G$ has a 1 for the coordinates corresponding to v and v_{i_κ} and 0s elsewhere. Therefore, it is consistent with $\pi|_{\oplus\kappa} = \{\overline{v}_{i_1}, \dots, \overline{v}_{i_{\kappa-1}}, v_{i_\kappa}\}$ (since $v \notin \{v_{i_1}, \dots, v_{i_{\kappa-1}}\}$ by the definition of V_κ). The input $\mathrm{Ind}[e]^{\oplus v} \in D_G$ is similarly consistent with $\pi|_{\oplus\kappa}$. Therefore, each $v \in V_\kappa$ is a distinct relevant variable for $\textsc{IsEdge}_{\pi|_{\oplus\kappa}}$ over D_G:

$$\textsc{IsEdge}_{\pi|_{\oplus\kappa}}(\mathrm{Ind}[e]) = 1 \quad \text{and} \quad \textsc{IsEdge}_{\pi|_{\oplus\kappa}}\left(\mathrm{Ind}[e]^{\oplus v}\right) = 0.$$

It follows that $\mathrm{Rel}(\textsc{IsEdge}_{\pi|_{\oplus\kappa}}; D_G) \geq |V_\kappa| = |E_\kappa|$ as desired. □

Proof of Claim 3.5.3. Let ρ be a certificate for IsEdge over D_G on 0^n. By Proposition 3.5.4, the variables of ρ form a vertex cover and so $|\rho| \geq k'$ where k' is the size of a vertex cover of G. Let $\pi = (\overline{v}_{i_1}, \dots, \overline{v}_{i'_k}) \in \mathrm{Perm}(\rho)$ be an arbitrary permutation of ρ. Then:

$$\begin{aligned}
\sum_{\kappa=1}^{|\pi|} \mathrm{Rel}(\textsc{IsEdge}_{\pi|_{\oplus j}}; D_G) &\geq \sum_{\kappa=1}^{|\pi|} |E(v_{i_\kappa}; v_{i_1}, \dots, v_{i_{\kappa-1}})| && \text{(Lemma 3.5.5)} \\
&= m. && \text{(Fact 3.4.1)}
\end{aligned}$$

It follows from Theorem 20 that $\mathrm{DT}(\textsc{IsEdge}, D_G) \geq k' + m$. □

3.5.3 Hardness distillation for ℓ-IsEdge

Following the ideas from Section 3.5.2, we show that the following set of inputs forms a coreset of ℓ-IsEdge.

Definition 11 (Coreset for ℓ-IsEdge)**.** *For an n-vertex, m-edge graph G and $\ell \in \mathbb{N}$, the set ℓ-$D_G \subseteq \{0,1\}^n \times (\{0,1\}^\ell)^n$ consists of the $m + m(2\ell+2) + 1$ many points*

- *all generalized edge indicators: ℓ-$\mathrm{Ind}[e] \in (\{0,1\}^n)^{\ell+1}$ for each edge $e \in E$ where ℓ-$\mathrm{Ind}[e] \coloneqq (\mathrm{Ind}[e])^{\ell+1}$;*
- *1-coordinate perturbations of edge indicators: $2\ell+2$ many points for each $e \in E$ obtained by flipping one of the 1-coordinates in ℓ-$\mathrm{Ind}[e]$; and*

- *the all* 0*s input:* $0^{n\ell+n}$.

Example. See Figure 3.8 for an example of a graph G and the associated set of inputs ℓ-D_G.

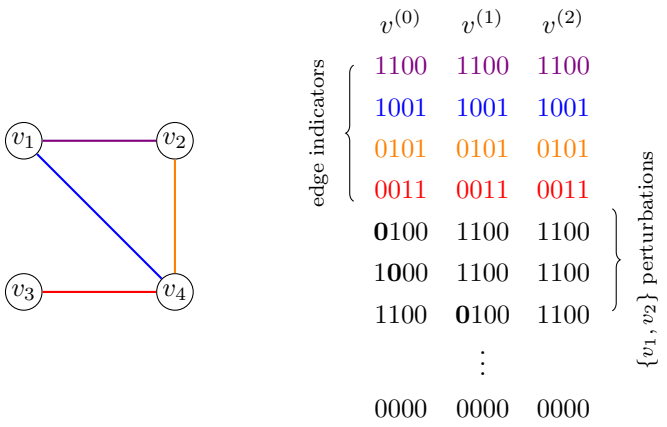


Figure 3.8: Example of a graph G on four vertices and the associated set of inputs ℓ-D_G with $\ell = 2$. The colored collection of points correspond to edge indicators. The next collection of points correspond to 1-coordinate perturbations of the duplicated variables of the edge indicator for the edge $e = \{v_1, v_2\}$. The perturbed coordinates are bold.

Recall from Theorem 16 that $\mathrm{DT}(\ell\text{-IsEdge}) \geq (\ell+1)\cdot(k'+m)$ where k' is the size of a vertex cover for G. The main claim of this section is that ℓ-D_G distills this hardness factor of ℓ-IsEdge.

Claim 3.5.6 (ℓ-D_G is a coreset for ℓ-IsEdge)**.** *Let G be an m-vertex graph and $\ell \in \mathbb{N}$ be arbitrary. Then,*

$$\mathrm{DT}(\ell\text{-IsEdge}, \ell\text{-}D_G) \geq (\ell+1)(k'+m)$$

where k' is the size of a vertex cover for G.

This claim is analgous to Claim 3.5.3 and the proof similarly proceeds in two steps. Ultimately, we will apply Theorem 20 where f is ℓ-IsEdge : $\{0,1\}^N \to \{0,1\}$, D is ℓ-D_G, and x is 0^N. As such, the first step extends Proposition 3.5.4 to ℓ-IsEdge and shows that certificates for 0^N contain vertex covers. The second step extends Lemma 3.5.5 and lower bounds the number of relevant variables of subfunctions of ℓ-IsEdge induced by certificates of 0^N.

Proposition 3.5.7 (Any certificate of 0^N contains a vertex cover)**.** *Let G be a graph and let $\{\overline{v}_{i_1}^{(j_1)}, \ldots, \overline{v}_{i_k}^{(j_k)}\}$ be a certificate for ℓ-IsEdge over ℓ-D_G on 0^N. Then, the vertices $\{v_{i_1}, \ldots, v_{i_k}\}$ form a vertex cover of G.*

Proof. If an edge e is not covered by the vertices $\{v_{i_1}, \ldots, v_{i_k}\}$, then the 1-input ℓ-Ind$[e]$ is consistent with any restriction of the form $\rho = \{\overline{v}_{i_1}^{(j_1)}, \ldots, \overline{v}_{i_k}^{(j_k)}\}$. Therefore, any such ρ cannot be a certificate. □

Lemma 3.5.8 (Lower bounding the number of relevant variables of ℓ-IsEdge subfunctions)**.** *Let ρ be a certificate for ℓ-IsEdge over ℓ-D_G on 0^N and $\pi = (\overline{v}_{i_1}^{(j_1)}, \dots, \overline{v}_{i_k}^{(j_k)}) \in \mathrm{Perm}(\rho)$, a permutation of ρ. Then*

$$\mathrm{Rel}(\ell\text{-IsEdge}_{\pi|_{\oplus\kappa}}, \ell\text{-}D_G) \geq \ell + (\ell+1) \cdot |E(v_{i_\kappa}; v_{i_1}, \dots, v_{i_{\kappa-1}})|$$

for all $\kappa \in [k]$ such that $E(v_{i_\kappa}; v_{i_1}, \dots, v_{i_{\kappa-1}}) \neq \varnothing$.

Proof. Let $\pi|_{\oplus\kappa}$ be as in the lemma statement and let E_κ and V_κ denote $E(v_{i_\kappa}; v_{i_1}, \dots, v_{i_{\kappa-1}})$ and $V(v_{i_\kappa}; v_{i_1}, \dots, v_{i_{\kappa-1}})$, respectively (recall these quantities from Definitions 7 and 8). If $E_\kappa \neq \varnothing$, then we will show that v_{i_κ} contributes ℓ relevant variables to $\mathrm{Rel}(\ell\text{-IsEdge}_{\pi|_{\oplus\kappa}}, \ell\text{-}D_G)$ and that each $v \in V_\kappa$ contributes $\ell + 1$.

The vertex v_{i_κ} contributes ℓ relevant variables. By assumption, E_κ is nonempty so there is some edge $e = (v_{i_\kappa}, v) \in E_\kappa$. The restriction $\pi|_{\oplus\kappa}$ sets one coordinate, $v_{i_\kappa}^{j_\kappa}$, to 1 and the other coordinates: $\{v_{i_1}, \dots, v_{i_{\kappa-1}}\}$ are set to 0. Since $v \notin \{v_{i_1}, \dots, v_{i_{\kappa-1}}\}$, the input $\ell\text{-Ind}[e] \in \{0,1\}^N$ is consistent with $\pi|_{\oplus\kappa}$. All of the coordinates in $\textsc{Dup}(v_{i_\kappa}) \cup \{v_{i_\kappa}^{(0)}\}$ are set to 1 in the input $\ell\text{-Ind}[e]$. Hence, for any $v' \in \textsc{Dup}(v_{i_\kappa}) \cup \{v_{i_k}^{(0)}\} \setminus \{v_{i_\kappa}^{(j_\kappa)}\}$, the input $\ell\text{-Ind}[e]^{\oplus v'}$ is consistent with $\pi|_{\oplus\kappa}$ since $v' \neq v_{i_\kappa}^{(j_\kappa)}$. Therefore,

$$\ell\text{-IsEdge}_{\pi|_{\oplus\kappa}}(\ell\text{-Ind}[e]) = 1 \quad \text{and} \quad \ell\text{-IsEdge}_{\pi|_{\oplus\kappa}}(\ell\text{-Ind}[e]^{\oplus v'}) = 0$$

and $\ell\text{-Ind}[e], \ell\text{-Ind}[e]^{\oplus v'} \in \ell\text{-}D_G$. Since v' was arbitrary this shows that each of the ℓ variables in $\textsc{Dup}(v_{i_\kappa}) \cup \{v_{i_\kappa}^{(0)}\} \setminus \{v_{i_\kappa}^{(j_\kappa)}\}$ is relevant for $\ell\text{-IsEdge}_{\pi|_{\oplus\kappa}}$ over $\ell\text{-}D_G$.

Each vertex $v \in V_\kappa$ contributes $\ell + 1$ relevant variables. Let $v \in V_\kappa$ be an arbitrary vertex and let $e = (v_{i_\kappa}, v) \in E_\kappa$ be the edge incident to v_{i_κ} that contains v. Let $v' \in \textsc{Dup}(v) \cup \{v^{(0)}\}$ be a coordinate of ℓ-IsEdge. As above, the inputs $\ell\text{-Ind}[e]$ and $\ell\text{-Ind}[e]^{\oplus v'}$ are both consistent with the restriction $\pi|_{\oplus\kappa}$. Moreover,

$$\ell\text{-IsEdge}_{\pi|_{\oplus\kappa}}(\ell\text{-Ind}[e]) = 1 \quad \text{and} \quad \ell\text{-IsEdge}_{\pi|_{\oplus\kappa}}(\ell\text{-Ind}[e]^{\oplus v'}) = 0$$

and $\ell\text{-Ind}[e], \ell\text{-Ind}[e]^{\oplus v'} \in \ell\text{-}D_G$. This shows that all $\ell + 1$ variables in $\textsc{Dup}(v) \cup \{v^{(0)}\}$ for $v \in V_\kappa$ is relevant. All of these relevant variables are unique and so the total number of relevant variables of $\ell\text{-IsEdge}_{\pi|_{\oplus\kappa}}$ is at least $\ell + (\ell+1)|V_\kappa| = \ell + (\ell+1)|E_\kappa|$ as desired. □

Proof of Claim 3.5.6. Let ρ be a certificate for ℓ-IsEdge over ℓ-D_G on 0^N. By Proposition 3.5.7, the variables in ρ form a vertex cover and so $|\rho| \geq$ the size of a vertex cover of G. Let $\pi = (\overline{v}_{i_1}^{(j_1)}, \dots, \overline{v}_{i_k}^{(j_k)}) \in \mathrm{Perm}(\rho)$ be a permutation of ρ. In order to apply hardness distillation (Theorem 20), we need to lower bound the number of relevant variables of $\ell\text{-IsEdge}_{\pi|_{\oplus\kappa}}$ for $\kappa =$

$1, 2, \ldots, |\pi|$. However, the lower bound from Lemma 3.5.8 only applies if the corresponding restricted edge neighborhood $E_\kappa = E(v_{i_\kappa}; v_{i_1}, \ldots, v_{i_{\kappa-1}})$ is nonempty. To this end, we consider the restriction $\rho' = \{\overline{v}_{i_\kappa}^{(j_\kappa)} \mid E_\kappa \neq \varnothing\} \subseteq \rho$. This restriction is still a certificate for ℓ-IsEdge over ℓ-D_G on 0^N and therefore must still contain a vertex cover by Proposition 3.5.7. Therefore, $|\rho'| \geq k'$ where k' is the size of a vertex cover of G. We can now write

$$\begin{aligned}
\sum_{\kappa=1}^{|\pi|} \mathrm{Rel}(\ell\text{-IsEdge}; \ell\text{-}D_G) &\geq \sum_{\substack{\kappa \in [|\pi|] \\ E_\kappa \neq \varnothing}} \mathrm{Rel}(\ell\text{-IsEdge}_{\pi|_{\oplus\kappa}}; \ell\text{-}D_G) \\
&\geq \sum_{\substack{\kappa \in [|\pi|] \\ E_\kappa \neq \varnothing}} \ell + (\ell+1)|E_\kappa| && \text{(Lemma 3.5.8)} \\
&= |\rho'|\ell + (\ell+1)m && \text{(Fact 3.4.1: } \{E_\kappa\} \text{ partition } E) \\
&\geq \ell k' + (\ell+1)m. && (\rho' \text{ contains a vertex cover})
\end{aligned}$$

We have satisfied the conditions of Theorem 20 with f being ℓ-IsEdge, D being ℓ-D_G, and x being 0^N. We conclude

$$\mathrm{DT}(\ell\text{-IsEdge}, \ell\text{-}D_G) \geq k' + k'\ell + (\ell+1)m = (\ell+1)(k'+m). \qquad \square$$

3.5.4 Learning consequence for inverse polynomial error

In this section, we use Claim 3.5.6 to obtain hardness of learning decision trees with membership queries. We recall, formally, the learning problem we are interested in.

> DT-Learn(n, s, s', ε): Given random examples from an unknown distribution $\mathcal{D}$ and membership queries to a size-s target decision tree, output a size-s' decision tree which ε-approximates the target over $\mathcal{D}$.

Theorem 21 (Hardness learning DTs with inverse polynomial error)**.** *For all constants $\delta' > 0, d \in \mathbb{N}$, there is a sufficiently small constant $\delta > 0$ such that the following holds. If* DT-Learn$(n, s, (1+\delta)\cdot s, \varepsilon)$ *with $s = O(n)$, $\varepsilon = O(1/n)$ can be solved in randomized time $t(n)$, then* VertexCover$(k, (1+\delta')\cdot k)$ *on degree-d, n-vertex graphs can be solved in randomized time $O(n^2 \cdot t(n^2))$.*

Proof. Given $\delta' > 1$ and $d \in \mathbb{N}$, let $\lambda < 1$ be any large enough constant so that $\lambda(1+\delta') > 1$ and let $\delta > 0$ be any constant satisfying $1 < (1+\delta) < \min\{\lambda(1+\delta'), 1 + \frac{1-\lambda}{d}\}$. Then we will use a learner for DT-Learn$(n, s, (1+\delta)\cdot s, \varepsilon)$ to solve VertexCover$(k, (1+\delta')k)$.

The reduction. Fix $\ell = \Theta(n)$ large enough so that $1 + \frac{1-\lambda}{d} > (1+\delta) + \frac{2(1+\delta)n}{\ell}$. Such an ℓ exists since $1 + \frac{1-\lambda}{d} > 1 + \delta$ by assumption. Consider the function ℓ-IsEdge : $\{0,1\}^N \to \{0,1\}$ and the set of inputs ℓ-D_G for $N = n + \ell n = \Theta(n^2)$. Let $\mathcal{D}$ be the distribution which is uniform over the set ℓ-D_G and fix $\varepsilon < 1/|\mathrm{supp}(\ell\text{-}D_G)| = O(1/m^2)$ (which is $O(1/N)$ since $n = \Theta(m)$ for constant degree graphs) and $s = \ell(k+m) + 2mn = O(N)$. Run the procedure in Figure 3.9.

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

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

Run: DT-Learn$(N, s, (1+\delta) \cdot s, \varepsilon)$ for $t(N)$ 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 $(\boldsymbol{v}^{(0)}, \ldots, \boldsymbol{v}^{(\ell)}) \sim \mathcal{D}$ for a random sample.

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

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

Output: Yes if and only if $|T_{\mathrm{hyp}}| \le (1+\delta) \cdot [\ell(k+m) + 2mn]$ and $\varepsilon_{\mathrm{hyp}} \le \varepsilon$

Figure 3.9: Using an algorithm for DT-Learn to solve VertexCover.

Runtime. Any query $(v^{(0)}, \ldots, v^{(\ell)}) \in \{0,1\}^N$ to ℓ-IsEdge can be answered in $O(N)$ time by looking at G and computing IsEdge$(v^{(0)})$ in time $O(m)$ then checking that the appropriate vertices are set to 1. Similarly, a random sample from $\mathcal{D}$ can be obtained in time $O(N)$ by picking a uniform random element of ℓ-D_G. This algorithm for VertexCover requires $O(N \cdot t(N))$ time to run the learner plus time $O(N^2)$ to compute $\mathrm{dist}_{\mathcal{D}}(T_{\mathrm{hyp}}, \ell\text{-IsEdge})$. Since $t(N) \ge N$, this implies an overall runtime of $O(N \cdot t(N))$ which is $O(n^2 \cdot t(n^2))$.

Correctness. For correctness, we analyze the yes and no cases separately.

Yes case: $\mathrm{VC}(G) \le k$. In this case, Theorem 16 ensures that

$$\mathrm{DT}(\ell\text{-IsEdge}) \le (\ell+1)(k+m) + mn \le \ell(k+m) + 2mn.$$

Therefore after $t(N)$ time steps, with high probability, the learner outputs a decision tree T_{hyp} satisfying

$$\mathrm{dist}_{\mathcal{D}}(T_{\mathrm{hyp}}, \ell\text{-IsEdge}) \le \varepsilon$$

and

$$\begin{aligned} |T_{\text{hyp}}| &\leq (1+\delta) \cdot \text{DT}(\ell\text{-IsEdge}) && \text{(learner assumption)} \\ &\leq (1+\delta) \cdot [\ell(k+m) + 2mn] && \text{(Theorem 16)} \end{aligned}$$

which ensures that our algorithm correctly outputs Yes.

No case: $\text{VC}(G) > (1+\delta') \cdot k$**.** Assume that $\text{dist}_{\mathcal{D}}(T_{\text{hyp}}, \ell\text{-IsEdge}) \leq \varepsilon < 1/|\text{supp}(\ell\text{-}D_G)|$ (otherwise the algorithm correctly outputs No). In particular, $\text{dist}_{\mathcal{D}}(T_{\text{hyp}}, \ell\text{-IsEdge}) = 0$. We would like to show that, under our assumption on $\text{VC}(G)$, $|T_{\text{hyp}}| > (1+\delta) \cdot [\ell(k+m) + 2mn]$. We start by bounding the vertex cover size of G:

$$\begin{aligned} \text{VC}(G) &= \lambda \text{VC}(G) + \frac{1-\lambda}{d} d\text{VC}(G) \\ &\geq \lambda \text{VC}(G) + \frac{1-\lambda}{d} m && \text{(Fact 3.3.1)} \\ &\geq \lambda \text{VC}(G) + \left(\delta + \frac{2(1+\delta)n}{\ell}\right) m && \left(\tfrac{1-\lambda}{d} > \delta + \tfrac{2(1+\delta)n}{\ell}\right) \\ &> (1+\delta)k + \left(\delta + \frac{2(1+\delta)n}{\ell}\right) m. && (\lambda \text{VC}(G) > \lambda(1+\delta')k > (1+\delta)k) \end{aligned}$$

This implies that

$$\ell \text{VC}(G) > (1+\delta)\ell k + \delta \ell m + 2(1+\delta)mn. \tag{3.1}$$

We can now write

$$\begin{aligned} |T_{\text{hyp}}| &\geq \text{DT}(\ell\text{-IsEdge}, \ell\text{-}D_G) && (T_{\text{hyp}} \text{ computes } \ell\text{-IsEdge over } \ell\text{-}D_G) \\ &> \ell(\text{VC}(G) + m) && \text{(Claim 3.5.6)} \\ &> (1+\delta)\ell k + (1+\delta)\ell m + 2(1+\delta)mn && \text{(Equation (3.1))} \\ &= (1+\delta) \cdot [\ell(k+m) + 2mn] \end{aligned}$$

which ensures that our algorithm correctly outputs No. □

3.6 Hardness for constant error

3.6.1 Hardness of partial vertex cover

Partial vertex cover. For a graph $G = (V, E)$ and $\alpha \in [0, 1)$, an α-partial vertex cover is subset of the vertices $C \subseteq V$ such that C covers at least a $(1-\alpha)$-fraction of the edges. The problem 0-partial vertex cover is the ordinary vertex cover problem. We write $\text{VC}_\alpha(G) \in \mathbb{N}$ to denote the

size of the smallest α-partial vertex cover of G. See Figure 3.10 for an example of a partial vertex cover. The problem α-partial (k, k')-VertexCover is to distinguish whether there exists an α-partial vertex cover of size $\leq k$ or every α-partial vertex cover requires size $> k'$. As with ordinary vertex cover, solving this gapped problem is equivalent to approximating α-partial vertex cover. Theorem 18 implies hardness of approximating α-partial vertex cover. It is possible to upgrade an α-partial vertex cover to an ordinary vertex cover by augmenting it with the vertices of uncovered edges.

Fact 3.6.1 (Upgrading α-partial vertex covers)**.** *Any α-partial vertex cover C for a graph G with m-edges can be transformed into a vertex cover C' for G satisfying $|C'| \leq |C| + 2\alpha m$.*

(a) A vertex cover and its covered edges highlighted in teal

(b) A $\frac{1}{5}$-partial vertex cover and its covered edges highlighted in purple

Figure 3.10: A graph $G = (V, E))$ with 10 edges having $\mathrm{VC}(G) = 3$ and $\mathrm{VC}_{1/5}(G) = 2$.

By definition, if C is an α-partial vertex cover, then C leaves at most αm edges uncovered. Augmenting C with the $\leq 2\alpha m$ vertices of these uncovered edges yields a vertex cover of G. The size of the resulting vertex cover is $\Theta(m)$ which would be problematic if G has small vertex covers. Fortunately, for constant degree graphs, $\mathrm{VC}(G) = \Theta(m)$ (Fact 3.3.1), so α-partial vertex covers for these graphs are close to optimal vertex covers. This enables us to show that α-partial vertex cover on constant degree graphs is just as hard to approximate as vertex cover. Claim 3.2.2 follows by combining Claim 3.6.2 with Theorem 18.

Claim 3.6.2 (Hardness of approximating α-partial vertex cover)**.** *For every constant $c' > 1$ and $d \in \mathbb{N}$, there are constants $\alpha \in (0, 1)$ and $c > 1$ such that if there is an algorithm solving α-partial $(k, c \cdot k)$-VertexCover on n-vertex, degree-d graphs in time $t(n)$, then there is an algorithm for solving $(k, c' \cdot k)$-VertexCover on n-vertex degree-d graphs in time $t(n)$. One can assume that $\alpha < \frac{1}{d+1}$.*

Proof. Given $c' > 1$, let $\alpha \in (0, 1)$ be small enough so that $1 < (1 - 2\alpha d)c'$ (and also small enough so that $\alpha < \frac{1}{d+1}$ for the second part of the claim) and let c be any constant satisfying $1 < c < (1 - 2\alpha d)c'$. We will solve $(k, c' \cdot k)$-VertexCover using an algorithm for α-partial $(k, c \cdot k)$-VertexCover.

Given a graph G and a parameter k, run the algorithm for α-partial $(k, c \cdot k)$-VertexCover on G and k. Output Yes if and only if the algorithm returns Yes. We claim that this procedure solves $(k, c' \cdot k)$-VertexCover on degree-d graphs. For correctness, we analyze the yes and no cases separately.

Yes case: $\mathrm{VC}(G) \leq k$**.** In this case, we have

$$\mathrm{VC}_\alpha(G) \leq \mathrm{VC}(G) \leq k$$

and so the algorithm correctly outputs YES.

No case: $\mathrm{VC}(G) > c'k$**.** Let m denote the number of edges of G and let C be the smallest α-partial vertex cover of G. Fact 3.6.1 implies that $|C'| - 2\alpha m \leq |C| = \mathrm{VC}_\alpha(G)$ where C' is a (possibly suboptimal) vertex cover for G. Therefore,

$$\begin{aligned}
\mathrm{VC}_\alpha(G) &\geq \mathrm{VC}(G) - 2\alpha m && \text{(Fact 3.6.1)} \\
&\geq \mathrm{VC}(G) - 2\alpha d \cdot \mathrm{VC}(G) && \text{(Fact 3.3.1)} \\
&> (1 - 2\alpha d)c'k && (\mathrm{VC}(G) > ck \text{ by assumption}) \\
&> ck && (c < (1 - 2\alpha d)c')
\end{aligned}$$

which means the algorithm correctly outputs NO. □

3.6.2 Definition of the hard distribution

For Theorem 21, we used the distribution which was uniform over the set ℓ-D_G. This distribution has the property that the target function ℓ-ISEDGE *can* be approximated with subconstant error by a small decision tree. In fact, the constant function $f(x) = 0$ obtains error $\leq \Pr[\ell\text{-ISEDGE} = 1] \leq 1/m$ in approximating ℓ-ISEDGE. Therefore, to obtain hardness in the constant-error regime, we need to define a new distribution, one over which the target function ℓ-ISEDGE is close to balanced. To this end, we define the following distribution.

Definition 12 (Constant-error hard distribution)**.** *For a graph G and $\ell \in \mathbb{N}$, the distribution ℓ-$\mathcal{D}_G$ over $\{0,1\}^n \times (\{0,1\}^\ell)^n$ is obtained via the following experiment*

- *with probability* $1/2$ *sample the all* 0*s input;*
- *with probability* $1/4$ *sample a generalized edge indicator,* ℓ-$\mathrm{Ind}[e]$ *for* $e \in E$ *uniformly at random;*
- *with probability* $1/4$ *sample a* 1*-coordinate perturbation of an edge indicator uniformly at random.*

We prove the following analogue of Claim 3.5.6 which shows that constant error decision trees must have large size.

Claim 3.6.3. *Let G be an m-edge graph, $\ell \in \mathbb{N}$, and $\alpha \in (0,1)$. If T is a decision tree satisfying*

$$\mathrm{dist}_{\ell\text{-}\mathcal{D}_G}(T, \ell\text{-IsEdge}) \leq \frac{1}{16} \cdot \alpha$$

then

$$|T| \geq (\ell+1) \cdot [\mathrm{VC}_\alpha(G) + (1-\alpha)m] .$$

We first need the following lemma showing how to extract an α-partial vertex cover from a decision tree for ℓ-IsEdge.

Lemma 3.6.4 (Obtaining an α-partial vertex cover from a constant-error decision tree for ℓ-IsEdge)**.** *Let T be a decision tree satisfying*

$$\mathrm{dist}_{\ell\text{-}\mathcal{D}_G}(T, \ell\text{-IsEdge}) < \frac{1}{4}\alpha$$

for any constant $\alpha \in (0,1)$. Then:

1. *the set $E' = \{e \in E \mid T(\ell\text{-Ind}[e]) = 1\}$ satisfies $|E'| \geq (1-a)m$; and*
2. *if $\pi = (\overline{v}_{i_1}^{(j_1)}, \ldots, \overline{v}_{i_k}^{(j_k)})$ is the path followed by 0^N in T, then for $E_\kappa = E(v_{i_\kappa}; v_{i_1}, \ldots, v_{i_{\kappa-1}})$, the set of vertices*

$$C = \{v_{i_\kappa} \mid E' \cap E_\kappa \neq \varnothing\}$$

covers all edges in E'. In particular, C is an α-partial vertex cover.

Proof. We prove the two points separately.

First point: $|E'| \geq (1-\alpha)m$. The set of edges $E \setminus E'$ correspond to inputs ℓ-Ind$[e]$ such that $T(\ell\text{-Ind}[e]) = 0$ but ℓ-IsEdge$(\ell\text{-Ind}[e]) = 1$. Since each input ℓ-Ind$[e]$ has mass $\frac{1}{4m}$ over ℓ-$\mathcal{D}_G$, we have

$$\begin{aligned}
\frac{1}{4}\alpha &> \mathrm{dist}_{\ell\text{-}\mathcal{D}_G}(T, \ell\text{-IsEdge}) && \text{(Assumption)} \\
&\geq |E \setminus E'| \cdot \frac{1}{4m} && \text{(Definition of } E') \\
&= (m - |E'|) \cdot \frac{1}{4m}.
\end{aligned}$$

Second point: C is an α-partial vertex cover. Since $\mathrm{dist}(T, \ell\text{-IsEdge}) < 1/2$, we know that $T(0^N) = 0$ and therefore the path π terminates in a 0-leaf. For every edge $e \in E'$, the input ℓ-Ind$[e]$ must diverge from π at some point $v_{i_\kappa}^{j_\kappa}$. This κ then satisfies $e \in E_\kappa$ so that $e \in E' \cap E_\kappa \neq \varnothing$. It follows that C covers the edge e. Since E' constitutes at least a $(1-\alpha)$-fraction of the edges, C is an α-partial vertex cover. □

Proof of Claim 3.6.3. Let T be any decision tree such that

$$\mathrm{dist}_{\ell\text{-}\mathcal{D}_G}(T, \ell\text{-IsEdge}) \leq \frac{1}{16}\alpha.$$

In particular, T satisfies the conditions of Lemma 3.6.4. Let E', π, E_κ, and C be as in the statement of Lemma 3.6.4. For each $v_{i_\kappa} \in C$, we define

$$R(v_{i_\kappa}) \coloneqq \textsc{Dup}(v_{i_\kappa}) \cup \left\{v_{i_\kappa}^{(0)}\right\} \setminus \left\{v_{i_\kappa}^{(j_\kappa)}\right\} \cup \bigcup_{\{v_{i_\kappa}, v\} \in E' \cap E_\kappa} \textsc{Dup}(v) \cup \left\{v^{(0)}\right\}.$$

Furthermore, let T_κ be the subtree which is the right child of $\pi(\kappa)$. That is T_κ is the subtree of T which catches all of the inputs $\ell\text{-Ind}[e]$ for $e \in E' \cap E_\kappa$. Recall from the proof of Lemma 3.5.8 that the variables in $R(v_{i_\kappa})$ are all relevant for the subfunction $\ell\text{-IsEdge}_{\pi|_{\oplus\kappa}}$. Each such relevant variable which is *not* queried in the subtree T_κ results in an error. For example, if T_κ does not query a variable $v' \in \textsc{Dup}(v_{i_\kappa}) \cup \left\{v_{i_\kappa}^{(0)}\right\} \setminus \left\{v_{i_\kappa}^{(j_\kappa)}\right\}$, then the string $\ell\text{-Ind}[e]^{\oplus v'}$ where $e \in E' \cap E_\kappa$ is classified as 1 by T:

$$T(\ell\text{-Ind}[e]^{\oplus v'}) = T_\kappa(\ell\text{-Ind}[e]^{\oplus v'}) = T_\kappa(\ell\text{-Ind}[e]) = 1$$

whereas $\ell\text{-IsEdge}(\ell\text{-Ind}[e]^{\oplus v'}) = 0$. Thus, each relevant variable which is *not* queried in the subtree T_κ results in a 0-input being misclassified as 1. Each such misclassification contributes $\Pr_{\ell\text{-}\mathcal{D}_G}[\ell\text{-Ind}[e]^{\oplus v'}] \geq \frac{1}{4} \cdot \frac{1}{m(2\ell+2)}$ to $\mathrm{dist}_{\ell\text{-}\mathcal{D}_G}(T, \ell\text{-IsEdge})$. Therefore, we can write

$$\begin{aligned}
\frac{1}{16}\alpha &\geq \mathrm{dist}_{\ell\text{-}\mathcal{D}_G}(T, \ell\text{-IsEdge}) \\
&\geq \left(\sum_{v_{i_\kappa} \in C} |R(v_{i_\kappa})| - |T_\kappa|\right) \cdot \frac{1}{8(m\ell + m)} + (m - |E'|) \cdot \frac{1}{4m} \\
&\geq \left(\sum_{v_{i_\kappa} \in C} |R(v_{i_\kappa})| - |T_\kappa|\right) \cdot \frac{1}{16\ell m} + (m - |E'|) \cdot \frac{1}{4m} && (m \leq \ell m)
\end{aligned}$$

where the quantity $\sum_{v_{i_\kappa} \in C} |R(v_{i_\kappa})| - |T_\kappa|$ counts how many 0-inputs are misclassified as 1 by T and $m - |E'|$ counts how many 1-inputs are misclassified as 0. These quantities are weighted by the

respective masses of each type of input over ℓ-$\mathcal{D}_G$. Rearranging gives the lower bound:

$$\begin{aligned}
\sum_{v_{i_\kappa} \in C} |T_\kappa| &\geq 4\ell(m - |E'|) - \alpha\ell m + \sum_{v_{i_\kappa} \in C} |R(v_{i_\kappa})| \\
&\geq 4\ell(m - |E'|) - \alpha\ell m + \ell|C| + \sum_{v_{i_\kappa} \in C} (\ell+1)|E' \cap E_\kappa| && (|\textsc{Dup}(v) = \ell|) \\
&= 4\ell(m - |E'|) - \alpha\ell m + \ell|C| + (\ell+1)|E'| && (\{E' \cap E_\kappa\} \text{ partitions } E') \\
&\geq 4\ell(m - |E'|) - \alpha\ell m + \mathrm{VC}_\alpha(G)\ell + (\ell+1)|E'| && (C \text{ is an } \alpha\text{-partial vertex cover}) \\
&\geq \ell\mathrm{VC}_\alpha(G) + (1-\alpha)(\ell+1)m. && (|E'| \leq m)
\end{aligned}$$

Therefore, since the T_κ and π are all disjoint parts of T:

$$\begin{aligned}
|T| &\geq |\pi| + \sum_{v_{i_\kappa} \in C} |T_\kappa| \\
&\geq |C| + \sum_{v_{i_\kappa} \in C} |T_\kappa| && (\text{Definition of } C) \\
&\geq \mathrm{VC}_\alpha(G) + \sum_{v_{i_\kappa} \in C} |T_\kappa| && (\text{C is an } \alpha\text{-partial vertex cover}) \\
&\geq (\ell+1)\left[\mathrm{VC}_\alpha(G) + (1-\alpha)m\right]
\end{aligned}$$

which completes the proof. □

3.6.3 Learning consequence for constant-error: Proof of Theorem 15

Theorem 22 (Hardness of learning DTs with constant error)**.** *For all constants $\delta' > 0$, $d \in \mathbb{N}$, and $\alpha < \frac{1}{d+1}$, there is a sufficiently small constant $\delta > 0$ such that the following holds. If* DT-Learn$(n, s, (1+\delta)\cdot s, \varepsilon)$ *with $s = O(n)$ and $\varepsilon = \Theta(1)$ can be solved in randomized time $t(n)$, then* α-PartialVertexCover$(k, (1+\delta')k)$ *on degree-d graphs can be solved in time $O(n^2 t(n^2))$.*

The proof of this theorem is similar to that of Theorem 21. The main difference is that our lower bound on the decision tree size of ℓ-IsEdge in the constant-error regime is quantitatively weaker than that of Claim 3.5.6. We will need to make the appropriate adjustments to the approximation factor of the DT-Learner in order to tolerate the weaker lower bound.

Proof. Let $\delta' > 0$, $d \in \mathbb{N}$, and $\alpha < \frac{1}{d+1}$ be given. The assumption that $\alpha < \frac{1}{d+1}$ implies $\alpha < \frac{1-\alpha}{d}$. Therefore, we can fix some $\lambda < 1$ large enough so that $\lambda(1+\delta') > 1$ *and* $\alpha < \frac{(1-\lambda)(1-\alpha)}{d}$. Let $\delta > 0$ be any constant satisfying $\frac{(1-\lambda)(1-\alpha)}{d} > \delta + \alpha$ and $(1+\delta) < \lambda(1+\delta')$. We will solve α-PartialVertexCover$(k, (1+\delta')k)$ using an algorithm for DT-Learn$(n, s, (1+\delta)\cdot s, \varepsilon)$.

The reduction. Fix $\ell = \Theta(n)$ large enough so that

$$\frac{(1-\lambda)(1-\alpha)}{d} > \delta + \alpha + \frac{2(1+\delta)n}{\ell}. \tag{3.2}$$

Such an ℓ exists by our assumption that $\frac{1-\lambda}{d} > \delta + \alpha$. As in Theorem 21 our target function will be ℓ-IsEdge : $\{0,1\}^N \to \{0,1\}$ for $N = n + \ell n = \Theta(n^2)$. Our distribution will be ℓ-$\mathcal{D}_G$ and we fix $\varepsilon < \frac{1}{16}\alpha = \Theta(1)$ and $s = \ell(k+m) + 2mn = O(N)$. Run the same procedure as in Figure 3.9 where the distribution $\mathcal{D}$ is ℓ-$\mathcal{D}_G$.

Runtime. As in the proof of Theorem 21, queries and random samples for ℓ-IsEdge can be handled in $O(N)$ time. Thus running the learner requires $O(N \cdot t(N))$ time. Computing the error $\mathrm{dist}_{\ell\text{-}\mathcal{D}_G}(T_{\mathrm{hyp}}, \ell\text{-IsEdge})$ takes $O(N^2)$ time. The overall runtime is therefore $O(N \cdot t(N))$ which is $O(n^2 \cdot t(n^2))$.

Correctness. We analyze the Yes case and No case separately.

Yes case: $\mathrm{VC}_\alpha(G) \le k$. This case is identical to the Yes case in Theorem 21. So our algorithm correctly outputs Yes.

No case: $\mathrm{VC}_\alpha(G) > (1+\delta')k$. Assume that $\mathrm{dist}_{\ell\text{-}\mathcal{D}_G}(T_{\mathrm{hyp}}, \ell\text{-IsEdge}) \le \varepsilon < \frac{1}{16}\alpha$ (otherwise our algorithm correctly outptus No). We would like to show that $|T_{\mathrm{hyp}}| > (1+\delta)\cdot[\ell(k+m)+2mn]$. We start by bounding α-partial vertex cover size of G:

$$\begin{aligned}
\mathrm{VC}_\alpha(G) &= \lambda \mathrm{VC}_\alpha(G) + \frac{1-\lambda}{d} d\mathrm{VC}_\alpha(G) \\
&\ge \lambda \mathrm{VC}_\alpha(G) + \frac{(1-\lambda)(1-\alpha)}{d} m && (d\mathrm{VC}_\alpha(G) \ge (1-\alpha)m \text{ for degree } d \text{ graphs}) \\
&\ge \lambda \mathrm{VC}_\alpha(G) + \left(\delta + \alpha + \frac{2(1+\delta)n}{\ell}\right) m && (\text{Equation (3.2)}) \\
&\ge (1+\delta)k + \left(\delta + \alpha + \frac{2(1+\delta)n}{\ell}\right) m. && (\lambda \mathrm{VC}_\alpha(G) > \lambda(1+\delta')k > (1+\delta)k)
\end{aligned}$$

Rearranging gives

$$\ell \mathrm{VC}_\alpha(G) \ge (1+\delta)k\ell + (\delta+\alpha)m\ell + 2(1+\delta)mn. \tag{3.3}$$

Therefore,

$$\begin{aligned}
|T_{\mathrm{hyp}}| &> \ell(\mathrm{VC}_\alpha(G) + (1-\alpha)m) && (\text{Claim 3.6.3}) \\
&> (1+\delta)k\ell + (\delta+\alpha)m\ell + 2(1+\delta)mn + (1-\alpha)m\ell && (\text{Equation (3.3)}) \\
&= (1+\delta)\cdot[\ell(k+m) + 2mn]
\end{aligned}$$

which ensures that our algorithm correctly outputs No. □

Remark 3 (Implications for testing decision trees)**.** The above proof of Theorem 22 and the proof of Theorem 21 actually prove hardness of *testing* decision tree size. Specifically, the proof of Theorem 22 shows that any tester which can distinguish whether a target function f is a size-s decision tree or is $\Omega(1)$-far from every size-s decision tree over a distribution $\mathcal{D}$ can also approximate PartialVertexCover. Therefore, the problem of distribution-free testing decision tree size is also NP-hard.

Proof of Theorem 15. If there were an algorithm for learning decision trees which satisfies the constraints of Theorem 15, then Theorem 22 shows that α-PartialVertexCover can be solved in $\mathrm{RTIME}(n^2 t(n^2))$. Theorem 18 and Claim 3.6.2 then imply that SAT can be solved in randomized time $O(n^2 \mathrm{polylog} n \cdot t(n^2 \mathrm{polylog} n))$. □

3.7 Obtaining stronger inapproximability via an XOR lemma for decision trees

In the PAC model with queries, 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. Theorem 15 shows 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 [Bsh93, GLR99, MR02, Fel16], but still left open the possibility of efficient algorithms achieving s' that is slightly larger than s.

In this section, we show that the problem remains NP-hard even for $s' = Cs$ where C is an arbitrarily large constant:

Theorem 23. *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 23 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 24. *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 24, 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 [ZB00], and subsequently [Sie08] showed that it is NP-hard even to approximate. We recover [Sie08]'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 [Sie08]). See Section 3.17 for details.

3.8 Background and Context

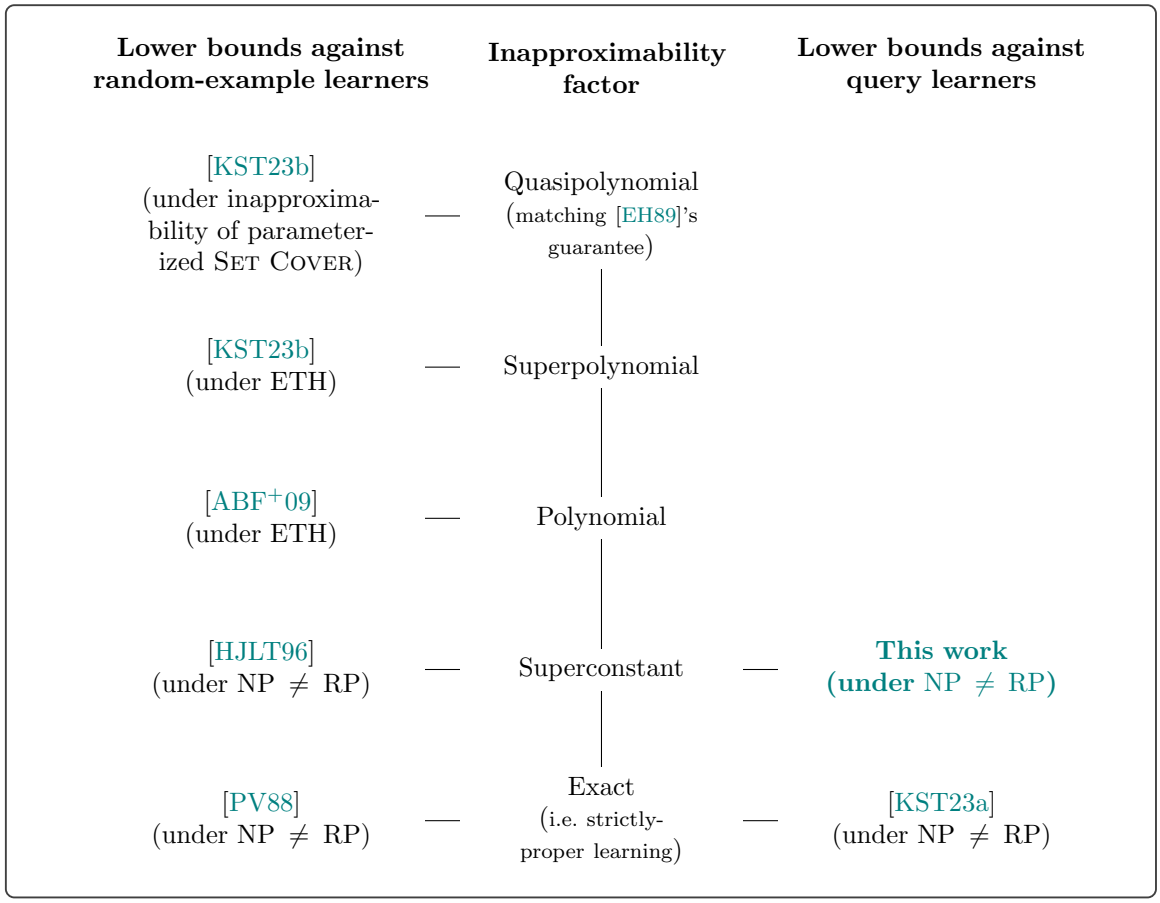
3.8.1 Lower bounds for random example learners

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 [EH89]'s weakly-proper algorithm under standard complexity-theoretic assumptions.

[PV88], 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 [Ang]. [HJLT96] then established a superconstant inapproximability factor (assuming NP $\neq$ RP), which was subsequently improved to polynomial by [ABF+09] (assuming the Exponential Time Hypothesis (ETH)). Recent work of [KST23b] further improves the inapproximability factor to superpolynomial (assuming ETH) and quasipolynomial (assuming the inapproximability of parameterized Set Cover), the latter of which exactly matches [EH89]'s performance guarantee.

It is reasonable to conjecture that [EH89]'s algorithm is optimal even for query learners. If so,

Figure 3.11: Summary of lower bounds for decision tree learning.



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 [KST23a] can be viewed as establishing the query-learner analogue of [PV88]'s result (i.e. the hardness of strictly-proper learning), this work can be viewed as establishing the query-learner analogue of [HJLT96]'s result (i.e. superconstant inapproximability of weakly-proper learning). Figure 3.11 summarizes the current landscape of decision tree lower bounds and shows how our work fits into it.

3.8.2 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. [Riv87, Blu92, KM12, Han93, Bsh93, BFJ+94, HJLT96, JS06, OS07, KS06, GKK08, KST09, HKY18, CM19]).

Examples of hypotheses that are constructed by existing algorithms include the sign of low-degree polynomials and small-depth boolean circuits.

3.9 Technical Overview

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

- **Step 1: Slight inapproximaibility.** We first give a new proof of Theorem 15. 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 $f^{\oplus r}(x^{(1)}, \ldots, x^{(r)}) \coloneqq f(x^{(1)}) \oplus \cdots \oplus f(x^{(r)})$.

There is a large body of work on XOR lemma for decision tree complexity [IRW94, NRS94, Sav02, Sha04, KvdW07, JKS10, Dru12, BDK18, BB19, BKLS20] 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 [Dru12], 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 [Dru12]'s proof technique, and we explain in Remark 4 why we cannot afford even a tiny constant factor loss in the exponent.

3.9.1 Step 1: Slight inapproximability

As in the proof of Theorem 15, we reduce from the NP-complete problem Vertex Cover. Recall that for every graph G there is an associated *edge indicator function* IsEdge (see Definition 6).

For technical reasons, we work with a generalization of IsEdge called ℓ-IsEdge where $\ell \in \mathbb{N}$ is a tuneable "padding parameter" (recall the definition of ℓ-IsEdge from Definition 9). We prove that the decision tree complexity of ℓ-IsEdge$_G$ scales with the vertex cover complexity of G with fairly tight quantitative parameters:

Claim 3.9.1. *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 ℓ-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 ℓ-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 [PY91, Hå07, DS05]. With an appropriate choice of parameters, Claim 3.9.1 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 3.9.1: Patch up and hard distribution lemmas. As is often the case in reductions such as Claim 3.9.1, 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. The analysis of the 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. (Our proof of Theorem 15 does not appear to be amenable to such a generalization, despite our best efforts at obtaining it.[2])

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.

Lemma 3.9.2 (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 $\pi(x)$.

[2] On a more technical level, the technique we use in Theorem 15 requires us to reason about the complexity of Partial Vertex Cover, a generalization of Vertex Cover, whereas the simpler approach here 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 3.9.3 (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*

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

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 3.9.1 follows by applying Lemmas 3.9.2 and 3.9.3 to the ℓ-IsEdge function and reasoning about its certificate complexity and sensitivity.

3.9.2 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 3.9.4 (XOR-ed version of Patch Up Lemma, see Lemma 3.14.1 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 $r \geq 1$,*

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

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

Lemma 3.9.5 (XOR-ed version of Hard Distribution Lemma, see Lemma 3.15.1 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 $r \geq 1$,*

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

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 3.9.2 and 3.9.3 combine to yield Claim 3.9.1, combining their XOR-ed generalizations Lemmas 3.9.2 and 3.15.1 yields the following amplified version of of Claim 3.9.1:

Claim 3.9.6. *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*

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

- *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-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 3.9.6 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-IsEdge$_G^{\oplus r}$, where δ' is a constant that depends only on δ. Theorem 24 follows as a corollary. See Figure 3.12 for an illustration of this amplification and how it fits into our overall reduction from Vertex Cover.

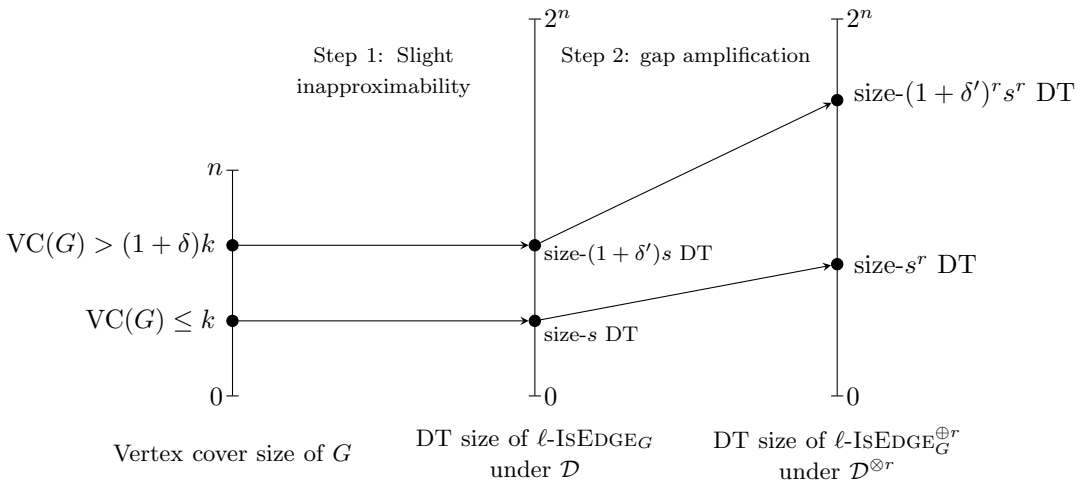


Figure 3.12: An illustration of main reduction from Vertex Cover in two steps. The first step, which establishes slight inapproximability of decision tree learning, is proved in Claim 3.9.1. The second step amplifies this slight inapproximability gap using Claim 3.9.6.

3.10 Preliminaries: sensitivity and certificate complexity

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 $\mathrm{Sens}(f) \coloneqq \max_{x\in D} |\mathrm{Sens}(f,x)|$.

3.11 Patching up a decision tree: Proof of Lemma 3.9.2

We start with the following claim about building a decision tree from scratch using certificates.

Claim 3.11.1 (Building a decision tree out of certificates)**.** *Let $f : D \to \{0,1\}$ be a function with $D \subseteq \{0,1\}^n$, then*

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

Proof. 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. □

The next lemma shows that we can patch up a decision tree by querying certificates. This recovers Lemma 3.9.2 in the setting where $D = \{0,1\}^n$.

Lemma 3.11.2 (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,

$$\begin{aligned}
\mathrm{DT}(f) &\leq \sum_{\pi \in \Pi} \mathrm{DT}(f_\pi) \\
&\leq \sum_{\pi \in \Pi} \left(1 + \sum_{x \in f_\pi^{-1}(1)} \mathrm{Cert}(f_\pi, x)\right) && \text{(Claim 3.11.1)} \\
&= |T| + \sum_{\pi \in \Pi} \sum_{x \in f_\pi^{-1}(1)} \mathrm{Cert}(f_\pi, x) && (|\Pi| = |T|) \\
&= |T| + \sum_{x \in f^{-1}(1)} \mathrm{Cert}(f_{\pi(x)}, x).
\end{aligned}$$

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

3.12 Hard distribution lemma: Proof of Lemma 3.9.3

Lemma 3.9.3 is proved for the *canonical hard distribution* for a partial function $f : D \to \{0,1\}$.

Definition 13 (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 3.9.3 in the setting where $D = \{0,1\}^n$.

Lemma 3.12.1 (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*

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

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

Claim 3.12.2 (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 of Lemma 3.12.1. 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 13 and notice that the support of this distribution is C and all of its sensitive neighbors. We have

$$\begin{aligned}
\mathrm{error}_{\mathcal{D}}(T, f) &= \Pr_{\boldsymbol{z}\sim\mathcal{D}}[T(\boldsymbol{z}) \neq f(\boldsymbol{z})] \\
&= \sum_{x\in C} \frac{1}{|C|} \cdot \Pr_{\boldsymbol{z}\sim\mathcal{D}}[T(\boldsymbol{z}) \neq f(\boldsymbol{z}) \mid \text{ first sampling } x \text{ from } f^{-1}(1)] \\
&\geq \frac{1}{2|C|} \sum_{x\in C} \frac{|\mathrm{Sens}(f_{\pi(x)}, x)|}{\max\{1, |\mathrm{Sens}(f, x)|\}} && \text{(Claim 3.12.2)} \\
&\geq \frac{1}{2|C| \cdot \mathrm{Sens}(f)} \sum_{x\in C} |\mathrm{Sens}(f_{\pi(x)}, x)| && (\max\{1, |\mathrm{Sens}(f, x)|\} \leq \mathrm{Sens}(f) \text{ for all } x)
\end{aligned}$$

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. □

3.13 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 from Theorem 15, while also achieving a better dependence on the Vertex Cover approximation factor. Later, we build on this reduction to prove our main result Theorem 23.

Theorem 25 (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 25 works by taking a graph G, and defining an "amplified" version of the edge indicator function for G (recall Definition 6). This function is called ℓ-IsEdge and is formally defined in Definition 9.

3.13.1 DT size upper bound for ℓ-IsEdge$_G$: First part of Claim 3.9.1

The upper bound in Claim 3.9.1 follows from Theorem 16.

Theorem 26 (Upper bound on decision tree size of ℓ-IsEdge)**.** *Let G be an n-vertex, m-edge graph with a vertex cover of size k. Then, there is a decision tree T computing ℓ-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 26, the constructivity of T, is implicit in the proof of Theorem 16, but is made explicit when proving Lemma 3.4.2.

3.13.2 DT size lower bound for ℓ-IsEdge$_G$: Second part of Claim 3.9.1

The lower bound in Claim 3.9.1 is proved with respect to the canonical hard distribution for ℓ-IsEdge$_G$:

Definition 14 (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 13, 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 3.13.1 (Decision tree size lower bound for computing ℓ-IsEdge)**.** *Let T be a decision tree for ℓ-IsEdge$_G$ satisfying* $\mathrm{error}_{\ell\text{-}\mathcal{D}_G}(T, \ell\text{-IsEdge}) \leq \varepsilon$*, then*

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

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

We obtain Lemma 3.13.1 using an application of the general size lower bound from Lemmas 3.11.2 and 3.12.1.

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 3.13.2 and propositions 3.13.3 and 3.13.4, respectively.

Lemma 3.13.2 (Zero-error lower bound for ℓ-IsEdge$_G$; see Claim 3.5.6)**.** *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-IsEdge$_G$: ℓ-$D_G \to \{0,1\}$ 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 3.5.6 under a slightly different subset of inputs. Specifically, they prove $\mathrm{DT}(\ell\text{-IsEdge}_G) \geq (\ell+1)(k'+m)$ where ℓ-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 $T(x) = \ell\text{-IsEdge}_G(x)$ 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 3.13.3 (Sensitivity of ℓ-IsEdge$_G$)**.** *For a graph G, $\ell \geq 1$, and ℓ-IsEdge$_G$: ℓ-$D_G \to \{0,1\}$, we have*

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

Proof. Let ℓ-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-Ind$[e]$. By definition, there are $2(\ell+1)$ many such i and each i is sensitive: ℓ-IsEdge$_G(\ell\text{-Ind}[e]^{\oplus i}) = 0$. Therefore, $|\text{Sens}(\ell\text{-IsEdge}_G, \ell\text{-Ind}[e])| = 2(\ell+1)$. Conversely, for every sensitive neighbor ℓ-Ind$[e]^{\oplus i}$, we have $\text{Sens}(\ell\text{-IsEdge}_G, \ell\text{-Ind}[e]^{\oplus i}) = \{\ell\text{-Ind}[e]\}$ and so $|\text{Sens}(\ell\text{-IsEdge}_G, \ell\text{-Ind}[e]^{\oplus i})| = 1$. Thus the overall sensitivity is $\text{Sens}(\ell\text{-IsEdge}_G) = 2(\ell+1)$. □

Proposition 3.13.4 (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 $x = \ell$-Ind$[e]$ and for all restrictions π, we have*

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

Proof. By definition $|\text{Sens}(\ell\text{-IsEdge}_\pi, x)|$ 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 ℓ-IsEdge$_\pi$ since fixing these coordinates forces ℓ-IsEdge to be the constant 1-function. It follows that $\text{Cert}(\ell\text{-IsEdge}_\pi, x) = |\text{Sens}(\ell\text{-IsEdge}_\pi, x)|$. □

Proof of Lemma 3.13.1. For a graph consisting of m edges, the number of 1-inputs to ℓ-IsEdge : ℓ-$D_G \to \{0,1\}$ is m. Therefore,

$$\begin{aligned}
\varepsilon &\geq \frac{1}{2m \cdot \text{Sens}(\ell\text{-IsEdge})} \sum_{x \in \ell\text{-IsEdge}^{-1}(1)} |\text{Sens}(\ell\text{-IsEdge}_{\pi(x)}, x)| && \text{(Lemma 3.12.1)} \\
&= \frac{1}{2m \cdot \text{Sens}(\ell\text{-IsEdge})} \sum_{x \in \ell\text{-IsEdge}^{-1}(1)} \text{Cert}(\ell\text{-IsEdge}_{\pi(x)}, x) && \text{(Proposition 3.13.4)} \\
&\geq \frac{1}{2m \cdot \text{Sens}(\ell\text{-IsEdge})} \left(\text{DT}(\ell\text{-IsEdge}, \ell\text{-}D_G) - |T|\right). && \text{(Lemma 3.11.2)}
\end{aligned}$$

Rearranging the above, we obtain

$$\begin{aligned}
|T| &\geq \text{DT}(\ell\text{-IsEdge}, \ell\text{-}D_G) - 2\varepsilon m \cdot \text{Sens}(\ell\text{-IsEdge}) \\
&\geq (\ell+1)(k'+m) - 2\varepsilon m \cdot \text{Sens}(\ell\text{-IsEdge}) && \text{(Lemma 3.13.2)} \\
&= (\ell+1)(k'+m) - 4\varepsilon m(\ell+1) && \text{(Proposition 3.13.3)}
\end{aligned}$$

which completes the proof. □

3.13.3 Putting things together to prove Theorem 25

The following key lemma is used in the analysis of correctness for the reduction in Theorem 25. It follows from Claim 3.9.1 and a careful choice of parameters.

Lemma 3.13.5 (Main technical lemma)**.** *For all $\delta, \delta', \varepsilon > 0$ and $d, k \geq 1$, 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 $dk \geq m$ 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 $\mathrm{error}_{\ell\text{-}\mathcal{D}_G}(T, \ell\text{-IsEdge}) \leq \varepsilon$.*

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 3.13.5. We state it on its own, since we will reuse the calculation later when proving Theorem 24.

Proposition 3.13.6. *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 of Lemma 3.13.5. 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 26 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 $\text{error}_{\ell\text{-}\mathcal{D}_G}(T, \ell\text{-IsEdge}) \leq \varepsilon$. Then, using our assumptions on the parameters:

$$\begin{aligned}
(1+\delta)s &= (1+\delta)\left[(\ell+1)(k+m) + mn\right] && \text{(Definition of } s\text{)} \\
&< (\ell+1)\left[(1+\delta')k + (1-4\varepsilon)m\right] && \text{(Proposition 3.13.6 with } \alpha = 4\varepsilon\text{)} \\
&\leq |T|. && \text{(Lemma 3.13.1 with } k' = (1+\delta')k\text{)}
\end{aligned}$$

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

Theorem 25 follows in a straightforward way from Lemma 3.13.5.

Proof of Theorem 25. 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 3.13.5. We will run $\mathcal{A}$ over the distribution $\ell\text{-}\mathcal{D}_G$ and on the function ℓ-IsEdge$_G$: $\{0,1\}^N \to \{0,1\}$ where ℓ is as in Lemma 3.13.5. Note that $N = n\ell + n = O(n^2)$ and $s = O(n^2) = O(N)$. See Figure 3.13 for the exact procedure we run.

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 $t(N, 1/\varepsilon) \geq N$, the overall runtime is $O(N \cdot t(N, 1/\varepsilon)) = O(n^2 t(n^2, 1/\varepsilon))$.

Correctness. Correctness follows from Lemma 3.13.5. 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 $|T_{\text{hyp}}| \leq (1+\delta) \cdot s$ and $\varepsilon_{\text{hyp}} \leq \varepsilon$ 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 $\varepsilon_{\text{hyp}} \leq \varepsilon$. Then, Lemma 3.13.5 ensures that $(1+\delta)s < |T_{\text{hyp}}|$ and so our algorithm correctly outputs "No" in this case as well. This completes the proof. □

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$(v^{(0)}, \dots, v^{(\ell)})$ for a query $(v^{(0)}, \dots, v^{(\ell)}) \in \{0,1\}^N$; and
- *random samples*: return $(\boldsymbol{v}^{(0)}, \dots, \boldsymbol{v}^{(\ell)}) \sim \ell\text{-}\mathcal{D}_G$ 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 3.13: Using an algorithm for DT-LEARN to solve VERTEX COVER.

3.14 Patching up a decision tree for $f^{\oplus r}$: Proof of Lemma 3.9.4

The following lemma recovers Lemma 3.9.4 by setting $D = \{0,1\}^n$.

Lemma 3.14.1 (XOR-ed version of patchup lemma, formal statement of Lemma 3.9.4)**.** *Let T be a decision tree and $f : D \to \{0,1\}$ be a nonconstant function for $D \subseteq \{0,1\}^n$, then*

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

For the lemma, we require the following generalization of a result from [Sav02]. 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 $\mathrm{DT}(f_1 \oplus f_2) \ge \mathrm{DT}(f_1) \cdot \mathrm{DT}(f_2)$ [Sav02, Lemma 2.1]. We will use the following analogous statement for partial functions.

Theorem 27 (Generalization of Savický [Sav02])**.** *Let $f^{(1)}, \dots, f^{(r)}$ be functions, $f^{(i)} : D^{(i)} \to \{0,1\}$ with $D^{(i)} \subseteq \{0,1\}^{n^{(i)}}$ for each $i = 1, \dots, r$. Then,*

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

Proof. First, the upper bound $\mathrm{DT}(f^{(1)} \oplus \cdots \oplus f^{(r)}) \le \prod_{i=1}^{r} \mathrm{DT}(f^{(i)})$ follows by considering the decision tree for $f^{(1)} \oplus \cdots \oplus f^{(r)}$ which sequentially computes $f^{(i)}(x)$ for each $i = 1, \dots, r$ using a decision tree of size $\mathrm{DT}(f^{(i)})$. See Figure 3.14 for an illustration of this decision tree.

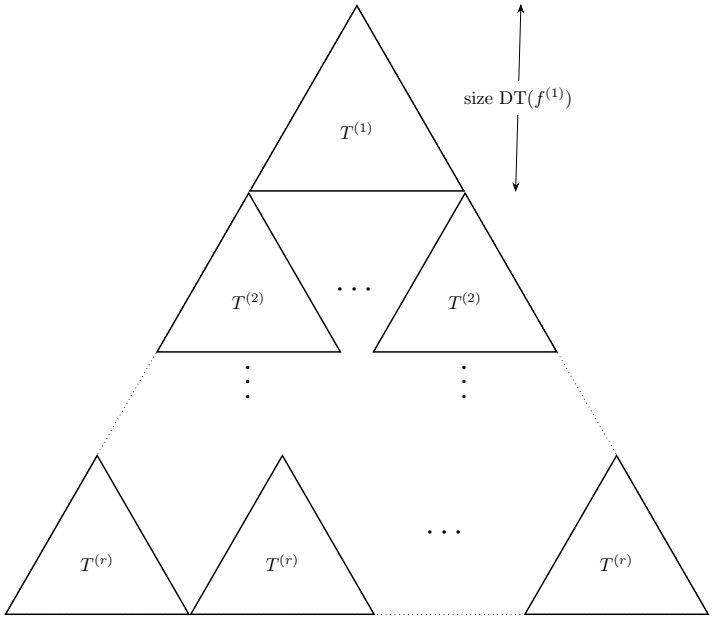


Figure 3.14: Illustration of a stacked decision tree for a function $f^{(1)} \oplus \cdots \oplus f^{(r)}$. For an input $x = (x^{(1)}, \ldots, x^{(r)})$, the decision tree sequentially computes $f^{(i)}(x^{(i)})$ for each $i = 1, \ldots, r$ using a decision tree $T^{(i)}$ of size $\mathrm{DT}(f^{(i)})$ 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 $\prod_{i=1}^{r} \mathrm{DT}(f^{(i)})$.

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:

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

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 of Lemma 3.14.1. 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 $f^{\oplus r}_{\pi}$ corresponds to the function $f_{\pi^{(1)}} \oplus \cdots \oplus f_{\pi^{(r)}}$ Then we have

$$
\begin{aligned}
\mathrm{DT}(f^{\oplus r}_{\pi}) &\leq \sum_{\pi \in \Pi} \mathrm{DT}(f^{\oplus r}_{\pi}) \\
&= \sum_{\substack{\pi \in \Pi \\ f^{\oplus r}_{\pi} \text{ is constant}}} \mathrm{DT}(f^{\oplus r}_{\pi}) + \sum_{\substack{\pi \in \Pi \\ f^{\oplus r}_{\pi} \text{ is nonconstant}}} \mathrm{DT}(f^{\oplus r}_{\pi}) \\
&= |T| + \sum_{\substack{\pi \in \Pi \\ f^{\oplus r}_{\pi} \text{ is nonconstant}}} \mathrm{DT}(f^{\oplus r}_{\pi}) && (\mathrm{DT}(f^{\oplus r}_{\pi}) = 1 \text{ when } f^{\oplus r}_{\pi} \text{ is constant}) \\
&= |T| + \sum_{\substack{\pi \in \Pi \\ f^{\oplus r}_{\pi} \text{ is nonconstant}}} \prod_{i=1}^{r} \mathrm{DT}(f_{\pi^{(i)}}) && (\text{Theorem 27}) \\
&\leq |T| + \sum_{\substack{\pi \in \Pi \\ f^{\oplus r}_{\pi} \text{ is nonconstant}}} \prod_{i=1}^{r} \left(1 + \sum_{x : f_{\pi^{(i)}}(x) = 1} \mathrm{Cert}(f_{\pi^{(i)}}, x) \right) && (\text{Claim 3.11.1})
\end{aligned}
$$

We use the fact that $1 + a \leq 2\max\{1, a\}$ for all $a \in \mathbb{R}$ to rewrite the above in a simpler form,

while suffering a factor of 2^r:

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

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 $i \in [r]$. □

3.15 Hard distribution lemma for $f^{\oplus r}$: Proof of Lemma 3.9.5

The following lemma recovers Lemma 3.9.5 by setting $D = \{0,1\}^n$.

Lemma 3.15.1 (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 $r \geq 1$,*

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

First, we prove the following claim.

Claim 3.15.2 (Error of $f^{\oplus r}$ conditioned on an input)**.** *Let T be a decision tree, $r \geq 1$, $f : D \to \{0,1\}$ 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*

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

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 of Lemma 3.15.1. As in the case of Lemma 3.12.1, we assume without loss of generality that $C = f^{-1}(1)$. We lower bound the error as follows

$$
\begin{aligned}
\varepsilon &\geq \Pr_{\boldsymbol{z}\sim\mathcal{D}^{\otimes r}}[T(\boldsymbol{z}) \neq f^{\oplus r}(\boldsymbol{z})] \\
&= \sum_{\substack{x\in C^r \\ \mathrm{Sens}(f_{\pi(x)}^{\oplus r}, x)\neq\varnothing}} \frac{1}{|C|^r} \cdot \Pr_{\boldsymbol{z}\sim\mathcal{D}^{\otimes r}}[T(\boldsymbol{z}) \neq f^{\oplus r}(\boldsymbol{z}) \mid \text{ first sampling } x \text{ from } C^r] \\
&\geq (2|C|)^{-r} \sum_{\substack{x\in C^r \\ \mathrm{Sens}(f_{\pi(x)}^{\oplus r}, x)\neq\varnothing}} \prod_{i=1}^{r}\frac{\max\{1, |\mathrm{Sens}(f_\pi^{(i)}, x^{(i)})|\}}{\max\{1, |\mathrm{Sens}(f^{(i)}, x^{(i)})|\}} \qquad \text{(Claim 3.15.2)} \\
&\geq (2|C| \cdot \mathrm{Sens}(f))^{-r} \sum_{\substack{x\in C^r \\ \mathrm{Sens}(f_{\pi(x)}^{\oplus r}, x)\neq\varnothing}} \prod_{i=1}^{r}\max\{1, |\mathrm{Sens}(f_\pi^{(i)}, x^{(i)})|\} \qquad (|\mathrm{Sens}(f^{(i)}, x^{(i)})| \leq \mathrm{Sens}(f))
\end{aligned}
$$

which completes the proof. □

3.16 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 23 and 24.

Theorem 28 (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 25 where the lower bound is obtained using a combination of Lemmas 3.14.1 and 3.15.1.

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

Theorem 29 (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 $\operatorname{error}_{\ell\text{-}\mathcal{D}_G^{\otimes r}}(T, \ell\text{-IsEdge}^{\oplus r}) \leq \varepsilon$ 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 $\operatorname{Sens}(\ell\text{-IsEdge}) = 2(\ell+1)$, we have

$$\begin{aligned}
\varepsilon &\geq \left(\frac{1}{4m(\ell+1)}\right)^r \sum_{\substack{x \in \ell\text{-}D_G^r \\ \operatorname{Sens}(\ell\text{-IsEdge}^{\oplus r}_{\pi(x)}, x) \neq \varnothing}} \prod_{i=1}^{r} \max\{1, \operatorname{Sens}(\ell\text{-IsEdge}^{(i)}_{\pi(x)}, x^{(i)})\} && \text{(Lemma 3.15.1)} \\
&= \left(\frac{1}{4m(\ell+1)}\right)^r \sum_{\substack{x \in \ell\text{-}D_G^r \\ \operatorname{Sens}(\ell\text{-IsEdge}^{\oplus r}_{\pi(x)}, x) \neq \varnothing}} \prod_{i=1}^{r} \max\{1, \operatorname{Cert}(\ell\text{-IsEdge}^{(i)}_{\pi(x)}, x^{(i)})\} && \text{(Proposition 3.13.4)} \\
&\geq \left(\frac{1}{4m(\ell+1)}\right)^r \sum_{\substack{x \in \ell\text{-}D_G^r \\ \ell\text{-IsEdge}^{\oplus r}_{\pi(x)} \text{ is nonconstant}}} \prod_{i=1}^{r} \max\{1, \operatorname{Cert}(\ell\text{-IsEdge}^{(i)}_{\pi(x)}, x^{(i)})\} \\
&\geq \left(\frac{1}{8m(\ell+1)}\right)^r (\operatorname{DT}(\ell\text{-IsEdge}^{\oplus r}) - |T|). && \text{(Lemma 3.14.1)}
\end{aligned}$$

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 non-constant, 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:

$$\begin{aligned}
|T| &\geq \mathrm{DT}(\ell\text{-IsEdge}^{\oplus r}) - \varepsilon\big[8m(\ell+1)\big]^r \\
&= \mathrm{DT}(\ell\text{-IsEdge})^r - \varepsilon\big[8m(\ell+1)\big]^r && \text{(Theorem 27)} \\
&\geq \big[(\ell+1)(k+m)\big]^r - \varepsilon\big[8m(\ell+1)\big]^r && \text{(Lemma 3.13.2)}
\end{aligned}$$

which completes the proof. □

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

Proposition 3.16.1. *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 Theorem 29 and proposition 3.16.1, we are able to prove the main technical lemma used for our reduction.

Lemma 3.16.2 (Main technical lemma for Theorem 28)**.** *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 $\ell\text{-IsEdge}^{\oplus r} : \{0,1\}^{r(n\ell+n)} \to \{0,1\}$; 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

$$\begin{aligned}
\mathrm{DT}(\ell\text{-IsEdge}^{\oplus r}) &= \mathrm{DT}(\ell\text{-IsEdge})^r && \text{(Theorem 27)} \\
&\leq [(\ell+1)(k+m) + mn]^r = s. && \text{(Theorem 26)}
\end{aligned}$$

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

$$\begin{aligned}
(1+\delta)^r s &= \Big[(1+\delta)[(\ell+1)(k+m)+mn]\Big]^r \\
&< \Big[(\ell+1)(1+\delta')k + (\ell+1)(1-8\varepsilon^{1/r})m\Big]^r && \text{(Proposition 3.13.6 with } \alpha = 8\varepsilon^{1/r}) \\
&\le \big[(\ell+1)(k+m)\big]^r - \varepsilon\big[8m(\ell+1)\big]^r && \text{(Proposition 3.16.1)} \\
&\le |T|. && \text{(Theorem 29)}
\end{aligned}$$

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

Proof of Theorem 28. 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 $dk \ge m$. If $dk < m$, our algorithm outputs "No" as G cannot have a vertex cover of size at most k (see Fact 3.3.1). Otherwise, we proceed under the assumption that $dk \ge m$. Let $s \in \mathbb{N}$ be the quantity from Lemma 3.16.2. 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 Lemma 3.16.2. Note that $N = r \cdot (n\ell + n) = O(rn^2)$ and $s = O(n^{2r})$. See Figure 3.15 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 $\ell\text{-IsEdge}^{\oplus r}(x)$ 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 3.15: Using an algorithm for DT-Learn on $\ell\text{-IsEdge}^{\oplus r}$ to solve Vertex Cover.

Runtime. Any query to $\ell\text{-IsEdge}_G^{\oplus r}$ 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 $\mathrm{dist}_{\mathcal{D}^{\otimes r}}(T, \ell\text{-IsEdge}_G^{\oplus r})$. 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 Lemma 3.16.2 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 $\varepsilon_{\mathrm{hyp}} > \varepsilon$ then our algorithm for Vertex Cover correctly outputs "No". Otherwise, assume that $\varepsilon_{\mathrm{hyp}} \le \varepsilon$. Then, Lemma 3.16.2 ensures that $(1+\delta)^r s < |T_{\mathrm{hyp}}|$ and so our algorithm correctly outputs "No" in this case as well.

□

Remark 4 (Why we require such sharp lower bounds in the proof of Theorem 28)**.** A key step in the analysis of the correctness of our reduction is Lemma 3.16.2. 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 Lemma 3.16.2.

3.16.1 Proof of Theorem 24

By Theorem 18, 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 $t(n\mathrm{polylog}n)$. Given an n-vertex graph with constant degree d, the reduction in Theorem 28 produces an instance of DT-Learn in time $n^{O(r)}$. We choose $A = 2^{\Theta(r)}$ in Theorem 28 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 28, 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. ■

3.16.2 Proof of Theorem 23

Let $C > 1$ be the constant from the theorem statement. Let $r \ge 1$ be a large enough constant so that the $2^{O(r)}$ term in Theorem 24 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 $\tilde{O}(rn^2) \cdot \mathrm{poly}(n^{O(r)}, 2^{O(r)})$ which is polynomial when r is constant. Therefore, the task is NP-hard. ■

3.17 Decision tree minimization given a subset of inputs

In this section, we show how our results yield new lower bounds for minimizing decision trees. First, we recall the problem of decision tree minimization [ZB00, Sie08].

Definition 15 (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'*.*

[Sie08] proves the following hardness results for DT-MIN:

Theorem 30 (Hardness of approximating DT-MIN [Sie08])**.** *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* NP $\subseteq$ DTIME$(n^{\mathrm{polylog}(n)})$

We observe that our proof of Theorem 24 recovers Theorem 30 and also strengthens the hardness results to hold even when the no case in Definition 15 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 31 (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* NP $\subseteq$ DTIME$(n^{\mathrm{polylog}(n)})$

Proof. These hardness results follow from the reduction in Theorem 28. 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 26 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 27 (see Figure 3.14) to get a decision tree for ℓ-IsEdge$^{\oplus r}$ whose size is $[(\ell+1)(n+m)+mn]^r$. We then choose ℓ-$D_G^r = \mathrm{supp}(\ell\text{-}\mathcal{D}_G^{\otimes r})$ to be the subset of inputs for the minimization instance. Moreover, it is straightforward to compute the pmf of the distribution ℓ-$\mathcal{D}_G^{\otimes r}$ and provide this to the algorithm for DT-Dataset-Min.

Therefore, as in the proof of Theorem 23, $(1+\delta')$-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 Lemma 3.16.2. Since $s = O(n^{2r})$, any $r = \mathrm{polylog} n$ 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. □

3.18 Discussion and future work

Assuming SAT requires exponential time, Theorem 15 shows that the inherent time complexity of properly learning decision trees with queries is also exponential: the simple dynamic-programming-based Occam algorithm is essentially optimal, despite evidence to the contrary in the form of fast algorithms for various relaxations of the problem.

A concrete problem left open by our work is that of optimizing the efficiency of our reduction, which takes an instance of SAT over n variables and produces an instance of properly learning decision trees over $\tilde{O}(n^2)$ variables. Can this be improved to linear or quasilinear in n?

More broadly, there is still work to do to fully understand the complexity of *weakly-proper* learning. As mentioned in the introduction, the landscape changes dramatically for this easier setting, and we have known since the 1980s of an algorithm that runs in quasipolynomial time [EH89]. This algorithm of Ehrenfeucht and Haussler has resisted improvement for over three decades and it is reasonable to conjecture that it is in fact optimal, even for query learners:

Conjecture 2. *There is no algorithm that, given queries to a size-s decision tree target and access to random labeled examples, runs in time $n^{o(\log s)}$ and returns an accurate decision tree hypothesis—one of any size, not necessarily s.*

Table 4.1 places Theorem 15 and Conjecture 2 within the context of prior work:

	Random Examples	**Queries**
Proper Learning	[Ang, PV88]: Exponential lower bound. Assumption: SAT requires exponential time	Theorem 15: Exponential lower bound. Assumption: ETH
Weakly-proper Learning	Theorem 5 ([KST23b]): Quasipoly lower bound. Assumption: Inapproximability of parameterized SetCover	Conjecture 2: Quasipoly lower bound.

Table 3.1: Lower bounds for proper and weakly-proper learning of decision trees. In terms of upper bounds, the fastest known proper algorithm (dynamic-programming-based Occam algorithm) runs in exponential time, and the fastest known weakly-proper (Ehreunfeucht–Haussler) runs in quasipolynomial time.

Weakly-proper learning algorithms are akin to approximation algorithms, and the hardness of weakly-proper learning is akin to the hardness of approximation. An immediate, but not necessarily insurmountable obstacle in extending our techniques to the setting of weakly-proper learning is the fact that VertexCover, whose hardness of approximation we rely on in our proof, is not *that* hard to approximate: a simple greedy algorithm achieves a 2-approximation.

There is also more to be understood for (strongly-)proper learning of decision trees. Our work taken together with the recent query learner of [BLQT22] highlights, quite dramatically, the effect of distributional assumptions on the problem: our work gives an exponential lower bound in the distribution-free setting, whereas [BLQT22] gives an almost-polynomial time query algorithm for the uniform distribution. In the spirit of beyond worst-case analysis, an ambitious direction for future work is to understand the tractability of the problem vis-à-vis the complexity of the underlying distribution. An ultimate goal is to design efficient algorithms that circumvent the lower bounds established in this work, but nonetheless enjoy performance guarantees for the broadest possible class of distributions.

Finally, we believe that the notions of hardness distillation and coresets introduced in this work merit further study and could lead to more connections between the hardness of minimization problems and the hardness of learning.

Chapter 4

The complexity of weakly learning decision trees

4.1 Introduction

This chapter connects two fundamental problems from two different areas, learning theory and coding theory.

> **Properly PAC Learning Decision Trees (DT-Learn).** Given random examples generated according to a distribution $\mathcal{D}$ and labeled by a function f, find a small decision tree that well-approximates f.

The fastest known algorithm for this problem is due to Ehrenfeucht and Haussler from 1989 and runs in quasipolynomial time:

Theorem ([EH89])**.** *There is an algorithm that, given random examples $(\boldsymbol{x}, f(\boldsymbol{x}))$ where $f : \{0,1\}^n \to \{0,1\}$ is a size-s decision tree and $\boldsymbol{x}$ is drawn from a distribution $\mathcal{D}$ over $\{0,1\}^n$, runs in $\mathrm{poly}(n^{\log s}, 1/\varepsilon)$ time and returns a decision tree T that is ε-close to f under $\mathcal{D}$.*

There are no known improvements to [EH89]'s algorithm even in the setting of *weak* learning where T only has to be mildly correlated with f (i.e. for values of ε close to $\frac{1}{2}$).

> **Parameterized Nearest Codeword Problem (k-NCP).** Given the generator matrix of a linear code of dimension n, a received word z, and a parameter k, decide if there is a codeword within Hamming distance k of z.

This problem is W[1]-hard [DFVW99], so it is natural to seek approximation algorithms. The current best algorithms achieve an approximation ratio of $O(n/\log n)$:

Theorem ([BK02, APY09])**.** *There is an algorithm that, given the generator matrix of a linear code $\mathcal{C}$ of dimension n, a received word z, a parameter k, and the promise that there is a codeword of $\mathcal{C}$ within distance k of z, runs in polynomial time and returns a codeword within distance αk of z where $\alpha = O(n/\log n)$.*

Berman and Karpinsky's algorithm is randomized whereas Alon, Panigrahy, and Yekhanin's is deterministic. Note that k-NCP can be solved exactly (i.e. with $\alpha = 1$) in time $n^{O(k)}$. There are no known algorithms that run in time $n^{o(k)}$ and achieve an approximation ratio of $\alpha = o(n/\log n)$.

4.1.1 Motivation for both problems

Both problems are central and well-studied in their respective fields of learning theory and coding theory. Part of the theoretical interest in DT-Learn—specifically, *proper* learning of decision trees—stems from the role that decision trees play in machine learning practice. They are the prime example of an interpretable hypothesis, and a recent survey of interpretable machine learning [RCC+22] lists the problem of constructing optimal decision tree representations of data as the very first of the field's "10 grand challenges".

[EH89]'s algorithm was one of the earliest PAC learning algorithms, coming on the heels of Valiant's introduction of the model [Val84]. Numerous works have since obtained faster algorithms for variants of the problem [Bsh93, KM12, SS93, JS05, OS07, GKK08, KST09, BLT20, BLQT22, BA24], but [EH89]'s algorithm for the original problem has resisted improvement. Indeed, faster algorithms for DT-Learn are known to have significant consequences within learning theory. Even just under the uniform distribution, DT-Learn contains as a special case the *junta problem* [Blu94, BL97], which itself has been called "the most important problem in uniform distribution learning" [MOS04]. Since every k-junta is a decision tree of size $s \leq 2^k$, an $n^{o(\log s)}$ time algorithm for DT-Learn gives an $n^{o(k)}$ time algorithm for learning k-juntas—this would be a breakthrough, as the current fastest algorithms run in n^{ck} time for some constant $c < 1$ [MOS04, Val05]. Far less is known about connections between DT-Learn and problems *outside* of learning theory.

The Nearest Codeword Problem (NCP), also known as Maximum Likelihood Decoding, has been called "probably the most fundamental computational problem on linear codes" [Mic]. While specific codes are often designed in tandem with fast decoding algorithms, results on the general problem have skewed heavily towards the side of hardness. NCP was proved to be NP-complete by Berlekamp, McEliece, and van Tilborg in 1978 [BMvT78]. Various aspects of its complexity has since been further studied in multiple lines of work, including the hardness of approximation [ABSS97, DKS98b, DKRS03, DMS03, Ale11]; hardness with preprocessing [BN90, Lob90, Reg03, FM04, GV05, AKKV11, KPV14]; hardness under ETH and SETH [BIWX11, SDV19]; and

most relevant to this work, hardness in the parameterized setting [DFVW99, ALW14, BELM16, BGKM18, Man20, LRSW22, BCGR23, LLL24, GLR+24]. On the other hand, the only known algorithms are those of [BK02, APY09].

4.2 Our results

We show how algorithms for DT-Learn yield approximation algorithms for k-NCP. Before stating our result in its full generality (Theorem 32 below), we first list a couple of its consequences. One instantiation of parameters shows that *any* improvement of [EH89]'s runtime, even in the setting of weak learning, will give new approximation algorithms for k-NCP with exponentially-improved approximation ratios:

Corollary 4.2.1. *Suppose there is an algorithm that given random examples generated according to a distribution $\mathcal{D}$ over $\{0,1\}^n$ and labeled by a size-s decision tree runs in time $n^{o(\log s)}$ and w.h.p. outputs a decision tree with accuracy $\frac{1}{2} + \frac{1}{\mathrm{poly}(n)}$ under $\mathcal{D}$. Then for $k = \Theta(\log s)$ there is a randomized algorithm running in time $n^{o(k)}$ which solves $O(\log n)$-approximate k-NCP.*

A different instantiation of parameters shows that a *polynomial-time* algorithm for properly learning decision trees, again even in the setting of weak learning, will give *constant-factor* approximation algorithms for k-NCP. Since the latter has been ruled out under standard complexity-theoretic assumptions [BELM16, Man20, LLL24], we get:

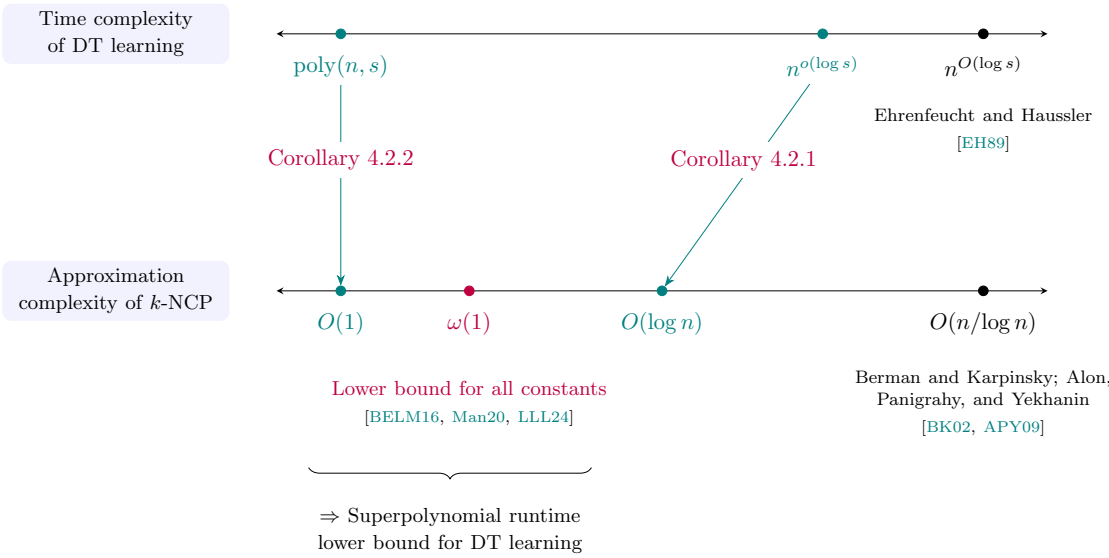
Corollary 4.2.2. *Assuming* W[1] $\neq$ FPT*, there is no polynomial-time algorithm for properly learning decision trees, even in the setting of weak learning.*

That is, there is no algorithm that given random examples generated according to a distribution $\mathcal{D}$ over $\{0,1\}^n$ and labeled by a size-n decision tree, runs in $\mathrm{poly}(n)$ time and w.h.p. outputs a decision tree hypothesis that achieves accuracy $\frac{1}{2} + \frac{1}{\mathrm{poly}(n)}$ under $\mathcal{D}$. Prior to our work, there were no results ruling out polynomial-time algorithms achieving error $\varepsilon = 0.01$, much less $\varepsilon = \frac{1}{2} - o(1)$. See Figure 4.1 for an illustration of our results.

4.2.1 Statement of our reduction

Corollaries 4.2.1 and 4.2.2 place no restrictions on the size s' of the decision tree hypothesis that the algorithm is allowed to output, other than the obvious one of $s' \leq t$ where t is the algorithm's runtime. The most general statement of our reduction decouples these two quantities. Algorithms that achieve small s' (ideally, close to the size s of the target decision tree) are of interest even if t is not comparably small. We show:

Figure 4.1: An illustration of the implications of our main result. The top axis denotes different runtimes for (weak) learning n-variable size-s decision trees. The bottom axis denotes approximation factors for k-NCP. The right hand side of each axis plots the best known algorithms for each respective problem. Each arrow indicates how a decision tree learning algorithm with a particular runtime yields an algorithm for k-NCP with a corresponding approximation ratio.



Theorem 32 (Our reduction)**.** *Suppose there is an algorithm that given random examples generated according to a distribution $\mathcal{D}$ over $\{0,1\}^n$ and labeled by a size-s decision tree, runs in time $t(n, s, s', \varepsilon)$ and w.h.p. outputs a size-s' decision tree hypothesis that achieves accuracy $1-\varepsilon$ under $\mathcal{D}$. Then, for all $\ell \in \mathbb{N}$ there is a randomized algorithm which solves α-approximate k-NCP running in time*

$$O(\ell n^2) \cdot t(\ell n, 2^{\ell k}, 2^{O(\alpha \ell k)}, \varepsilon) + \mathrm{poly}(n, \ell, 2^{\alpha \ell k}) \quad \textit{where } \varepsilon = \tfrac{1}{2} - 2^{-\Omega(\alpha \ell k)}.$$

(The parameter ℓ will be used to pad instances of k-NCP for small k to get instances of learning size-s decision trees for large s.)

Decoupling s' and t allows us to show variants of Corollary 4.2.2 where we obtain stronger time lower bounds at the price of stronger complexity-theoretic assumptions:

Corollary 4.2.3. *Suppose there is an algorithm which given random examples generated according to a distribution $\mathcal{D}$ and labeled by a size-n decision tree w.h.p. outputs a decision tree hypothesis of* $\mathrm{poly}(n)$ *size that achieves accuracy $\frac{1}{2} + \frac{1}{\mathrm{poly}(n)}$ under $\mathcal{D}$. Then:*

1. *(Corollary 4.2.2 restated) If the algorithm runs in* $\mathrm{poly}(n)$ *time, then* $\mathrm{W}[1] = \mathrm{FPT}$.

2. *If the algorithm runs in time $n^{(\log n)^{\delta}}$ for a sufficiently small constant δ, then* ETH *is false.*

3. *If the algorithm runs in time $n^{o(\log n)}$, then* Gap-ETH *is false.*

Addressing the main open problem from [EH89]. Paraphrasing the very first open problem of [EH89], the authors ask:

> *For the concept class of polynomial-size decision trees (i.e. $s = \text{poly}(n)$), can one design algorithms that run in polynomial time (i.e. achieve $t \leq \text{poly}(n)$)? Failing that, can one at least design algorithms that take superpolynomial time as those given here, but return polynomial-size decision tree hypotheses (i.e. achieve $s' \leq \text{poly}(n)$)?"*

Corollary 4.2.2 provides a negative answer to the first question and Corollary 4.2.3 provides negative answers to the second, with both holding even in the setting of weak learning.

4.2.2 Comparison with prior work

While we view the connection between DT-Learn and k-NCP as our main contribution, the new lower bounds that we obtain (i.e. Corollaries 4.2.2 and 4.2.3) also compare favorably with existing ones.

Inverse-polynomial error. There is a long line of work on the hardness of DT-Learn in the regime of inverse-polynomial error, $\varepsilon = 1/\text{poly}(n)$. Pitt and Valiant [PV88] first showed, via a simple reduction from Set Cover, that properly learning decision size-s decision trees (where $s = n$) to such an accuracy is NP-hard—*if* the algorithm is additionally required to output a hypothesis whose size s' exactly matches that of the target (i.e. $s' = s$). Hancock, Jiang, Li, and Tromp [HJLT96] subsequently ruled out polynomial-time algorithms that are required to return a hypothesis of size $s' \leq s^{1+o(1)}$, under the assumption that SAT cannot be solved in randomized quasipolynomial time. Alekhnovich, Braverman, Feldman, Klivans, and Pitassi [ABF+09] then ruled out polynomial-time algorithms, now with no restrictions on hypothesis size, under the randomized Exponential Time Hypothesis (ETH). Koch, Strassle, and Tan [KST23b] improved [ABF+09]'s lower bound to $n^{\Omega(\log \log n)}$ under the randomized ETH, and to $n^{\Omega(\log n)}$ under a plausible conjecture on the complexity of Set Cover.

Constant error. The above line of work is built successively on [PV88]'s reduction from Set Cover, which appears limited to the setting where $\varepsilon = 1/\text{poly}(n)$. Our work in [KST23a] showed, via a new reduction from Vertex Cover, that the problem is NP-hard even for ε being a small absolute constant ($\varepsilon = 0.01$). However, this result again only holds if the algorithm is required to output a hypothesis of size $s' = s$, like in the original result of [PV88].

Reference	Restriction on hypothesis size s'	Error ε	Runtime lower bound	Hardness assumption
[PV88]	$s' = s$	$1/\mathrm{poly}(n)$	$n^{\omega(1)}$	SAT $\notin$ RP
[HJLT96]	$s' \leq s^{1+o(1)}$	$1/\mathrm{poly}(n)$	$n^{\omega(1)}$	SAT $\notin$ RTIME$(n^{\mathrm{polylog}(n)})$
[ABF+09]	None	$1/\mathrm{poly}(n)$	$n^{\omega(1)}$	ETH
[KST23b]	None	$1/\mathrm{poly}(n)$	$n^{\Omega(\log \log n)}$	ETH
[KST23a]	$s' = s$	0.01	$n^{\omega(1)}$	SAT $\notin$ RP
Corollary 4.2.2	None	$\frac{1}{2} - \frac{1}{\mathrm{poly}(n)}$	$n^{\omega(1)}$	W[1] $\neq$ FPT
Corollary 4.2.3	$s' \leq \mathrm{poly}(s)$	$\frac{1}{2} - \frac{1}{\mathrm{poly}(n)}$	$n^{(\log n)^{\Omega(1)}}$	ETH
Corollary 4.2.3	$s' \leq \mathrm{poly}(s)$	$\frac{1}{2} - \frac{1}{\mathrm{poly}(n)}$	$n^{\Omega(\log n)}$	Gap-ETH

Table 4.1: Lower bounds for properly learning n-variable size-s decision trees under standard complexity-theoretic assumptions. All of them hold for $s = n$.

Summary. Prior lower bounds either held for $\varepsilon = 1/\mathrm{poly}(n)$, or for $\varepsilon = 0.01$ under the restriction that $s' = s$. For constant ε there were no lower bounds for general polynomial-time algorithms (i.e. ones without any restriction on their hypothesis size), and for $\varepsilon = \frac{1}{2} - o(1)$, there were no lower bounds even under the strictest possible restriction that $s' = s$. See Table 4.1.

As we will soon discuss, the linear-algebraic nature of k-NCP is crucial to our being able achieve hardness in the regime of $\varepsilon = \frac{1}{2} - o(1)$. While we cannot rule out the possibility that the previous reductions from Set Cover and Vertex Cover can be extended to this regime, we were unable to obtain such an extension despite our own best efforts—it seems that a fundamentally different approach is necessary.

In general, results basing the hardness of weak learning (of any learning task) on *worst-case* complexity-theoretic assumptions remain relatively rare. One reason is because the setting of weak learning corresponds to that of average-case complexity, and so any such result will have to amplify worst-case hardness into average-case hardness within the confines of the learning task at hand.[1]

[1] While boosting establishes an equivalence of weak and strong learning, boosting algorithms do not preserve the structure of the hypothesis. For example, boosting a weak learner that returns a decision tree hypothesis yields a strong learner that returns a hypothesis that is the majority of decision trees. Therefore, the hardness of properly learning decision trees in the setting of strong learning does not immediately yield hardness in the setting of weak learning.

4.2.3 Byproduct of our techniques: new lower bounds for testing juntas

Along the way to proving our main theorem, we give new results for the problem of *junta testing*. In this problem, the algorithm is given access to a function $f : \{0,1\}^n \to \{0,1\}$ and a distribution $\mathcal{D}$ over $\{0,1\}^n$, and the goal is to output a number $\hat{r}$ such that (i) $\hat{r}$ is as small as possible and (ii) f is well-approximated by an $\hat{r}$-junta under $\mathcal{D}$.

We prove a new NP-hardness result for this task:

Theorem 33. *Suppose there is an algorithm that, given queries to a function $f : \{0,1\}^n \to \{0,1\}$, i.i.d. draws from a distribution $\mathcal{D}$ over $\{0,1\}^n$, a parameter $r \in \mathbb{N}$, runs in time $t(n)$ and w.h.p. distinguishes between:*

- Yes*: f is an r-junta under $\mathcal{D}$.*

  $$\text{There is an } r\text{-junta } g \text{ such that } \Pr_{\boldsymbol{x}\sim\mathcal{D}}[f(\boldsymbol{x}) = g(\boldsymbol{x})] = 1.$$

- No*: f is $\frac{1}{2}$-far from all $\hat{r} := r \cdot 2^{\Theta(\log n/\log\log n)}$-juntas under $\mathcal{D}$.*

  $$\text{For every } \hat{r}\text{-junta } g, \text{ we have } \Pr_{\boldsymbol{x}\sim\mathcal{D}}[f(\boldsymbol{x}) = g(\boldsymbol{x})] = \tfrac{1}{2}.$$

Then $\textsc{Sat} \in \mathrm{RTIME}(t(\mathrm{poly}(n)))$.

In the language of property testing, this is the problem of *testing juntas* in the distribution-free model of Goldreich, Goldwasser, and Ron [GGR98]. In a survey on junta testing [Bla16], Blais concludes with "particularly appealing open problems related to the junta testing problem that are motivated by its application to the feature selection problem", the first of which is to design *distance approximators* for juntas. Theorem 33 sheds new light on this problem, showing that without distributional assumptions, it is intractable even for extremely coarse approximations, both in terms of the distance parameter and the size of the junta (i.e. 1 vs. $\frac{1}{2}$ and r vs. $\hat{r}$ in the Yes and No cases respectively).

Previous NP-hardness results for junta testing . Alekhnovich et al. [ABF+09], building on ideas in [PV88, HJLT96], gave an approximation-preserving reduction from Set Cover to junta testing. Given a Set Cover instance I of size n, they gave a polynomial-time reduction that produces a set $D \subseteq \{0,1\}^n$ of size n and a partial function $f : D \to \{0,1\}$ such that any algorithm that distinguishes between:

- Yes: $f : D \to \{0,1\}$ is computed by a k-junta
- No: Any function that computes $f : D \to \{0,1\}$ has more than αk many relevant variables

also distinguishes between the optimal set cover for I having size at most k versus greater than αk. Since Set Cover is NP-hard to approximate to within a factor of $\alpha = \Theta(\log n)$ [LY94, RS97, Fei98, DS14, Mos15], this implies that it is NP-hard to distinguish between:

- Yes: $f : \{0,1\}^n \to \{0,1\}$ is a k-junta under $\mathrm{Unif}(D)$
- No: $f : \{0,1\}^n \to \{0,1\}$ is $\frac{1}{n}$-far from all $\Theta(k \log n)$-juntas under $\mathrm{Unif}(D)$

where $\mathrm{Unif}(D)$ denotes the uniform distribution over D.

4.3 Discussion

Two interpretations of our results. The existing literature on properly learning decision trees is split roughly evenly between algorithms and hardness, and there is no consensus as to whether [EH89]'s algorithm is optimal. As for the approximability of k-NCP, there is a huge gap between the $O(n/\log n)$ ratio achieved by the algorithms of [BK02, APY09] and the constant-factor inapproximability results of [BELM16, Man20, LLL24], and there is likewise no consensus as to what the optimal ratio is within this range.

Corollary 4.2.1 can be viewed either as a new avenue for designing approximation algorithms for k-NCP or as one for showing that [EH89]'s algorithm is optimal. With regards to the former perspective, as already mentioned [EH89]'s quasipolynomial-time algorithm has been improved for variants of the problem—for example, we have polynomial-time algorithms that return hypotheses that are slightly more complicated than decision trees [Bsh93] and almost-polynomial-time query algorithms for the uniform distribution [BLQT22]. A natural avenue for future work is to see if the ideas in these works can now be useful for k-NCP or its variants. As for the latter perspective, the $O(n/\log n)$-versus-constant gap in our understanding of the approximability of k-NCP is especially stark when compared to the *unparameterized* setting, where NCP has long been known to be NP-hard to approximate to almost-polynomial ($n^{\Omega(1/\log\log n)}$) factors [DKS98b, DKRS03]. We hope that our work provides additional motivation for getting lower bounds in the parameterized setting "caught up" with those in the unparameterized setting.

More broadly, recent years have seen a surge of progress on parameterized inapproximability; see [FKLM20] for a survey. Notably, for example, a recent breakthrough of Guruswami, Lin, Ren, Sun, Wu [GLR+24] establishes the parameterized analogue of the PCP Theorem.[2] The framework of parameterized inapproximability syncs up especially nicely with the setup of learning theory: the parameterized setting is relevant because it allows us to control the size of the target function, and the inapproximability ratio corresponds to the gap in sizes between the target and hypothesis. We believe that there is much more to be gained, both in terms of algorithms and hardness, by further exploring connections between these two fields.

[2] Their work also carries new implications for k-NCP, though the parameters achieved by [BELM16, Man20, LLL24] are quantitatively stronger for our purposes.

Decision trees and weak learning in practice. Our interest in the setting of weak learning is motivated in part by a specific use case of decision trees in practice. *Tree ensemble methods* such as XGBoost [CG16] have emerged as powerful general-purpose algorithms that achieve state-of-the-art performance across a number of settings (especially on tabular data where they often outperform deep neural nets [SZA22, GOV22]). Roughly speaking, these methods first construct an ensemble of decision trees, each of which is mildly correlated with the data, and then aggregate the predictions of these trees into an overall prediction.

Our results provide a theoretical counterpoint to the empirical success of these methods. We show that the task of finding even a single small single decision tree that is mildly correlated with the data—the task that is at the very heart of these ensemble methods—is intractable. Indeed, Corollaries 4.2.2 and 4.2.3 show that this is the case even if the data is *perfectly* labeled by a small decision tree—a strong stylized assumption that real-world datasets almost certainly do not satisfy.

LPN hardness of uniform-distribution learning? A criticism that can be levied against all existing lower bounds for properly learning decision trees, including ours, is that they only hold if the examples are distributed according to a worst-case distribution. It would therefore be interesting to establish the hardness of learning under "nice" distributions, the most canonical one being the uniform distribution. Our work points to the possibility of basing such hardness on the well-studied *Learning Parities with Noise* problem [BFKL93, BKW03] (LPN), a distributional variant of NCP where the input is a random linear code instead of a worst-case code. Unfortunately, our reduction does not preserve the uniformity of distributions—i.e. it translates the hardness of LPN into the hardness of learning under a non-uniform distribution—but perhaps a modification of it can.

4.4 Technical Overview for Theorem 32

4.4.1 Warmup: DT-Learn solves decisional approximate k-NCP

We first show, as a warmup, how algorithms for DT-Learn can be used to solve the *decision version* of approximate k-NCP:

Definition 16 (Decisional α-approximate k-NCP)**.** *Given as input the generator matrix $G \in \mathbb{F}_2^{n \times d}$ of a code $\mathcal{C}$, a received word $z \in \mathbb{F}_2^n$, a distance parameter $k \in \mathbb{N}$, and an approximation parameter $\alpha \geq 1$, distinguish between:*

- Yes*: there is a codeword $y \in \mathcal{C}$ within Hamming distance k of z;*
- No*: the Hamming distance between z and every codeword $y \in \mathcal{C}$ is greater than αk.*

Theorem 34 (Theorem 32 for *decisional* approximate k-NCP)**.** *Suppose there is an algorithm that given random examples distributed according to a distribution $\mathcal{D}$ over $\{0,1\}^n$ and labeled by a size-s*

decision tree, runs in time $t(n, s, s', \varepsilon)$ *and outputs a size-*s' *decision tree hypothesis that achieves accuracy* $1 - \varepsilon$ *under* $\mathcal{D}$*. Then, for all* $\ell \in \mathbb{N}$ *there is an algorithm which solves* decisional α*-approximate* k-NCP *running in time*

$$O(\ell n^2) \cdot t(\ell n, 2^{\ell k}, 2^{O(\alpha \ell k)}, \varepsilon) + \mathrm{poly}(n, \ell, 2^{\alpha \ell k}) \text{ where } \varepsilon = \tfrac{1}{2} - 2^{-\Omega(\alpha \ell k)}.$$

There are no known search-to-decision reductions for approximate k-NCP, but in Section 4.4.2 we will explain how our proof of Theorem 34 can be upgraded to show that algorithms for DT-Learn in fact be used to solve the actual *search* version of approximate k-NCP, thereby yielding Theorem 32.

Dual formulation. We begin by transforming Definition 16 into its dual formulation where the algorithm is given as input the *parity check matrix* of a code instead of its generator matrix:

Definition 17 (Parity check view of decisional α-approximate k-NCP)**.** *Given as input the parity check matrix* $H \in \mathbb{F}_2^{m \times n}$ *of a linear code and a target vector* $t \in \mathbb{F}_2^m$*, distinguish between:*

- Yes*: there is a* k*-sparse vector* $x \in \mathbb{F}_2^n$ *such that* $Hx = t$
- No*: there does not exist a* αk*-sparse* $x \in \mathbb{F}_2^n$ *such that* $Hx = t$*.*

This view of NCP is also known as *syndrome decoding* in coding theory. The fact that one can efficiently switch between the two views of NCP is standard and follows by elementary linear algebra. The parity check view aligns especially well with the task of testing and learning an unknown function $f : \mathbb{F}_2^n \to \mathbb{F}_2$[3] since it can be equivalently stated as follows.

Definition 18. *Given as input a set* $D = \{x^{(1)}, \ldots, x^{(m)}\} \subseteq \mathbb{F}_2^n$ *and a partial function* $f : D \to \mathbb{F}_2$*, distinguish between:*

- Yes*:* f *is a* k*-parity*
- No*:* f *disagrees with every* αk*-parity on at least one input* $x \in D$*.*

We have reformulated decisional α-approximate k-NCP as the problem of distinguishing between $f : \mathbb{F}_2^n \to \mathbb{F}_2$ being a k-parity under $\mathrm{Unif}(D)$ versus $\frac{1}{m}$-far from all αk-parities under $\mathrm{Unif}(D)$.

Our strategy

Proving Theorem 34 therefore amounts to amplifying the gap between the Yes and No cases in such a way that f remains a sparse parity in the Yes case, and yet becomes $(\frac{1}{2} - 2^{-\Omega(\alpha k)})$-far from all decision trees of size $2^{\Omega(\alpha k)}$ in the No case. We do so incrementally in three steps. See Figure 4.2 for an illustration of these steps and Figure 1.2 for an illustration of the inclusions of the different function classes we consider.

[3] For the rest of the paper, we switch to viewing Boolean functions as mapping vectors in $\mathbb{F}_2^n$ to $\mathbb{F}_2$ since this aligns well with the linear-algebraic nature of NCP and our proofs.

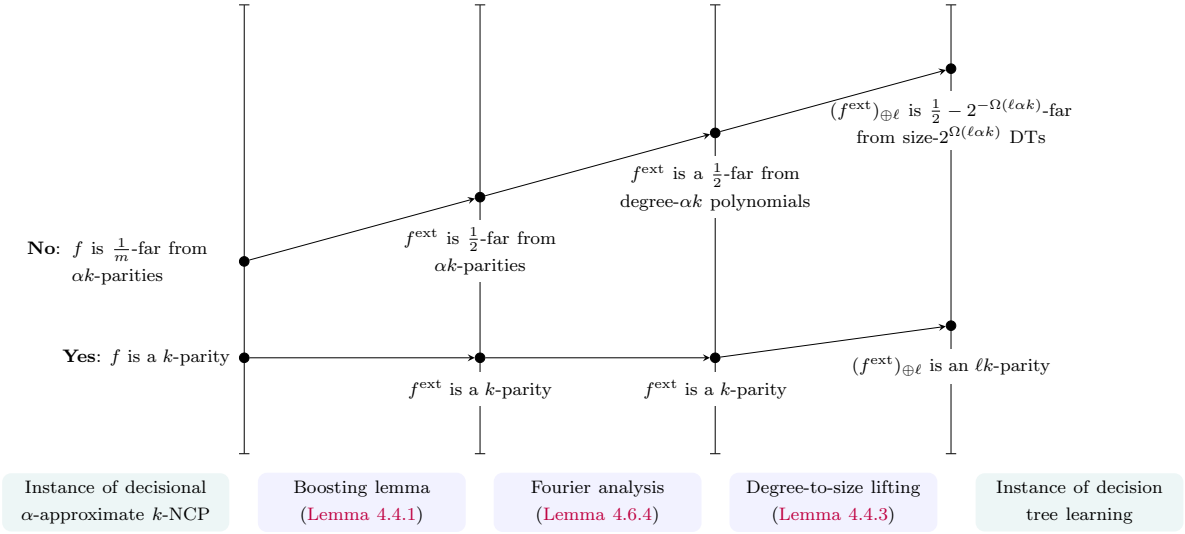


Figure 4.2: An illustration of Theorem 34 as a series of gap amplification steps. Starting with an instance of k-NCP on the left, we perform a series of transformations to obtain an instance of the distinguishing problem on the right. Due to space constraints we have omitted descriptions of the corresponding distributions from the figure. These distributions also go through a series of transformations, from $\mathrm{Unif}(D)$ on the left to $\mathrm{Unif}(\mathrm{Span}(D))_{\oplus \ell}$ on the right.

Step 1. For the first step, we consider the linear span of D:

$$\mathrm{Span}(D) := \left\{ \sum_{i \in S} x^{(i)} \mid S \subseteq [m] \right\},$$

where we have assumed for simplicity that the vectors in D are linearly independent. (Otherwise, the span is defined to be all possible linear combinations of the basis vectors of D.) We analogously consider f's linear extension $f^{\text{ext}} : \mathrm{Span}(D) \to \mathbb{F}_2$: for all $S \subseteq [m]$,

$$f^{\text{ext}}\left(\sum_{i \in S} x^{(i)} \right) = \sum_{i \in S} f\big(x^{(i)}\big)$$

and we prove the following "boosting lemma":

Lemma 4.4.1. *For every set $D \subseteq \mathbb{F}_2^n$ and function $f : D \to \mathbb{F}_2$, we have:*

- *Preservation of the* Yes *case: if f is a parity χ_S, then f^{ext} is also the parity χ_S.*
- *Amplification of the* No *case: if f disagrees with every αk-parity on at least one input in D, then f^{ext} disagrees with every αk-parity on exactly $\frac{1}{2}$ of the inputs in* $\mathrm{Span}(D)$.

Note that the domain of our function has been increased exponentially in size, since $|\mathrm{Span}(D)| =$

$2^{|\dim(D)|}$. Thankfully, this is not an issue since we will still be able to efficiently provide the learner with random examples sampled from this exponentially large set.

Step 2. The second step follows by Fourier analysis: if a function is uncorrelated with any small parity under $\mathcal{D}$, then by linearity of expectation, it is also uncorrelated with any low-degree Fourier polynomial under $\mathcal{D}$.

Step 3. Finally, we give a generic way to lift lower bounds against low-degree polynomials to lower bounds against small-size decision trees. For intuition about this step, we briefly sketch an elementary proof for the case when $\mathcal{D}$ is the *uniform* distribution. We claim that every small-size decision tree is well-approximated by a low-degree polynomial under the uniform distribution. To see this, note that truncating a size-s tree T at depth d yields a tree T_{trunc} that is $(2^{-d}s)$-close to T w.r.t. the uniform distribution. This is because the fraction of inputs that follow any path of length d is precisely 2^{-d} and we take a union bound over at most s truncated paths. Finally, the fact that depth-d decision trees have Fourier degree d completes the proof.

This proof fails for an arbitrary distribution $\mathcal{D}$ since the probability that a random $\boldsymbol{x} \sim \mathcal{D}$ follows a path of length d can now be much larger than 2^{-d}. To overcome this, we show that by composing $\mathcal{D}$ with a parity gadget, it becomes "uniform enough" for this fact to hold. The parity gadget is defined as follows.

Notation. For a vector $y \in (\mathbb{F}_2^\ell)^n$, we write $y^{(i)} \in \mathbb{F}_2^\ell$ to denote the ith block of y. We define the function $\text{BlockwisePar} : (\mathbb{F}_2^\ell)^n \to \mathbb{F}_2^n$:

$$\text{BlockwisePar}(y) \coloneqq (\oplus y^{(1)}, \ldots, \oplus y^{(n)}),$$

where $\oplus y^{(i)}$ denotes the parity of the bits in $y^{(i)}$.

Definition 19 (Parity substitution in functions and distributions)**.** *For a function $g : \mathbb{F}_2^n \to \mathbb{F}_2$, the function $g_{\oplus\ell} : (\mathbb{F}_2^\ell)^n \to \mathbb{F}_2$ is defined as*

$$g_{\oplus\ell}(y) = g(\text{BlockwisePar}(y)).$$

For a distribution $\mathcal{D}$ over $\mathbb{F}_2^n$, the distribution $\mathcal{D}_{\oplus\ell}$ is defined via the following experiment:

1. *First sample $\boldsymbol{x} \sim \mathcal{D}$.*
2. *For each $i \in [n]$, sample $\boldsymbol{y}^{(i)} \sim \mathbb{F}_2^\ell$ u.a.r. among all strings satisfying $\oplus \boldsymbol{y}^{(i)} = \boldsymbol{x}_i$. Equivalently, sample $\boldsymbol{y} \sim \mathcal{D}_{\oplus\ell}(\boldsymbol{x})$ where $\mathcal{D}_{\oplus\ell}(\boldsymbol{x})$ is the uniform distribution over all $y \in (\mathbb{F}_2^\ell)^n$ satisfying $\text{BlockwisePar}(y) = \boldsymbol{x}$.*

A key property of the parity substitution operation that for *any* initial distribution $\mathcal{D}$, the parity-substituted distribution $\mathcal{D}_{\oplus\ell}$ becomes "uniform-like" in the sense that the probability a random $\boldsymbol{y} \sim \mathcal{D}_{\oplus\ell}$ is consistent with a fixed restriction decays exponentially in the length of the restriction.

Proposition 4.4.2 ($\mathcal{D}_{\oplus\ell}$ is uniform-like)**.** *For any $\ell \geq 2$, let $R \subseteq [n\ell]$ be a subset of coordinates and $r \in \mathbb{F}_2^{|R|}$. Then,*

$$\Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[\boldsymbol{y}_R = r] \leq 2^{-\Omega(|R|)}.$$

Proposition 4.4.2 together with a couple of additional observations yields:

Lemma 4.4.3 (Degree-to-size lifting)**.** *Let $\mathcal{D}$ be any distribution over $\mathbb{F}_2^n$ and suppose $g : \mathbb{F}_2^n \to \mathbb{F}_2$ is $\frac{1}{2}$-far from all polynomials of Fourier degree αk under $\mathcal{D}$. Then for all $\ell \geq 2$, we have that $g_{\oplus\ell} : (\mathbb{F}_2^\ell)^n \to \mathbb{F}_2$ is $(\frac{1}{2} - 2^{-\Omega(\ell\alpha k)})$-far from all decision trees of size $2^{O(\ell\alpha k)}$ under $\mathcal{D}_{\oplus\ell}$.*

4.4.2 Proof of Theorem 32: DT-Learn solves the search version of k-NCP

As in the proof of Theorem 34, we first move from the generator matrix formulation of k-NCP to the parity check formulation (Definition 17). We therefore assume that our input is of the form $(H, t) \in \mathbb{F}_2^{m\times n} \times \mathbb{F}_2^m$ where there is a k-sparse vector $x \in \mathbb{F}_2^n$ such that $Hx = t$. Our goal, in the search version of approximate k-NCP, is to *find* a k'-sparse vector $x' \in F_2^n$ such that $Hx' = t$, where k' is as close to k as possible. By the equivalence between Definitions 17 and 18, this instance (H, t) can be viewed as a set $D \subseteq \mathbb{F}_2^n$ and a k-parity $f : D \to \mathbb{F}_2$, and our goal can be equivalently stated as that of finding a k'-parity $h : D \to \mathbb{F}_2$ that agrees with f, where k' is as close to k possible.

Running through the 3-step transformation of the Yes case outlined in the previous section, we can efficiently provide the learner with random examples distributed according to $\mathrm{Unif}(\mathrm{Span}(D))_{\oplus\ell}$ and labeled by $(f^{\mathrm{ext}})_{\oplus\ell}$. Suppose the learner returns a size-s' tree T that is γ-correlated with $(f^{\mathrm{ext}})_{\oplus\ell}$ under $\mathrm{Unif}(\mathrm{Span}(D))_{\oplus\ell}$. We will show how the desired k'-parity $h : D \to \mathbb{F}_2$ can be extracted from T. Roughly speaking, this amounts to showing that the proof we sketched in the previous section can be "unwound" to give an efficient algorithm for extracting such a parity. There are 4 steps to our analysis:

Step 1. By the contrapositive of Claim 4.6.10, truncating T at depth $\Theta(\log s') =: k'$ yields a tree T_{trunc} that is $(\gamma - \Theta(\frac{1}{s'}))$-correlated with $(f^{\mathrm{ext}})_{\oplus\ell}$ under $\mathrm{Unif}(\mathrm{Span}(D))_{\oplus\ell}$.

Step 2. Using basic Fourier-analytic properties of small-depth decision trees, we show that there exists a k'-parity χ_S in the Fourier support of T_{trunc} that is $((\gamma-\Theta(\frac{1}{s'}))4^{-k'})$-correlated with $(f^{\mathrm{ext}})_{\oplus\ell}$ under $\mathrm{Unif}(\mathrm{Span}(D))_{\oplus\ell}$.

Step 3. Implicit in the proof of Corollary 4.6.8 is that fact that we can undo the parity substitution operation and obtain from the aforementioned k'-parity χ_S a (k'/ℓ)-parity $\chi_{S^\star}$ whose correlation

with f^{ext} is the same as the correlation between χ_S and $(f^{\text{ext}})_{\oplus \ell}$:

$$\mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{D}}\left[f^{\text{ext}}(\boldsymbol{x})\chi_{S^\star}(\boldsymbol{x})\right] = \mathop{\mathbb{E}}_{\boldsymbol{y} \sim \mathcal{D}_{\oplus \ell}}\left[(f^{\text{ext}})_{\oplus \ell}(\boldsymbol{y})\chi_S(\boldsymbol{y})\right] = \left(\gamma - \Theta\Big(\frac{1}{s'}\Big)\right)4^{-k'}.$$

Step 4. Implicit in the proof of Lemma 4.4.1 is that fact that as long as the correlation between $\chi_{S^\star}$ and f^{ext} is positive, then $\chi_{S^\star}$ must in fact agree with f^{ext} on *all* of $\text{Span}(D)$, and hence with f on all of D.

4.5 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. For $R \subseteq [n]$, we write $x_R \in \{0,1\}^{|R|}$ to denote the substring of x on the coordinates in R. A string $x \in \{0,1\}^n$ is k-sparse if it has at most k nonzero entries. We use $\mathbb{F}_2$ to denote the finite field of order 2. When dealing with finite fields, it will be convenient for us to identify a Boolean function on n bits as a map $\mathbb{F}_2^n \to \mathbb{F}_2$.

Distributions. We use boldface letters e.g. $\boldsymbol{x}, \boldsymbol{y}$ to denote random variables. For a distribution $\mathcal{D}$, we write $\text{dist}_{\mathcal{D}}(f, g) = \Pr_{\boldsymbol{x} \sim \mathcal{D}}[f(\boldsymbol{x}) \neq g(\boldsymbol{x})]$. A function f is ε-close to g under $\mathcal{D}$ if $\text{dist}_{\mathcal{D}}(f, g) \leq \varepsilon$. Similarly, f is ε-far from g under $\mathcal{D}$ if $\text{dist}_{\mathcal{D}}(f, g) \geq \varepsilon$. If f is 0-close under $\mathcal{D}$ to some g having property $\mathcal{P}$, then we say that f has property $\mathcal{P}$ under $\mathcal{D}$. For example, "f is a k-parity under $\mathcal{D}$" means that there is a k-parity g which is 0-close to f under $\mathcal{D}$. For a set S, $\text{Unif}(S)$ denotes the uniform distribution over that set.

Parities and decision trees. For $S \subseteq [n]$, we write $\chi_S : \{0,1\}^n \to \{0,1\}$ to denote the parity of the coordinates in S. A k-parity function is a function χ_S for some $S \subseteq [n]$ with $|S| \leq k$. A decision tree T is a binary tree whose internal nodes query a coordinate and whose leaves are labeled by binary values. For a decision tree T, its size is the number of leaves in T and is denoted $|T|$.

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* $\text{EX}(f, \mathcal{D})$. 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 $\text{dist}_{\mathcal{D}}(f, h) \leq \varepsilon$. 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. $2/3$.

4.5.1 Complexity-theoretic assumptions

We list the hypotheses we use in order of strength of the hypothesis.

Hypothesis 5 (W[1] $\neq$ FPT, see [DF13, CFK+15b])**.** *For any computable function $\Phi : \mathbb{N} \to \mathbb{N}$, no algorithm can decide if a graph $G = (V, E)$ contains a k-clique in $\Phi(k) \cdot \text{poly}(|V|)$ time.*

Hypothesis 6 (Exponential time hypothesis (ETH) [Tov84b, IP01, IPZ01])**.** *There exists a constant $\delta > 0$ such that 3-SAT on n variables cannot be solved in $O(2^{\delta n})$ time.*

Hypothesis 7 (Gap-ETH [Din16, MR17])**.** *There exist constants $\lambda, \delta > 0$ such that no algorithm running in time $O(2^{\delta m})$ can solve the following task. Given a 3-SAT instance φ with m clauses distinguish between*

- Yes*: there exists an assignment of φ satisfying all m clauses; and*
- No*: every assignment of φ satisfies at most $(1 - \lambda)m$ clauses.*

Our hardness results will be based on randomized versions of these hypotheses make the same runtime assumption but also against randomized algorithms. We remark that W[1] $\neq$ FPT is a weaker assumption than ETH which itself is weaker than Gap-ETH.

4.5.2 Parameterized complexity of k-NCP

Bonnet, Egri, Lin, and Marx in [BELM16] (see also [BBE+21]) show that obtaining any constant approximation of k-NCP is W[1]-hard:

Theorem 35 (W[1]-hardness of approximating k-NCP, follows from [BELM16, Theorem 2])**.** *Assuming* W[1] $\neq$ FPT*, for all constants $c > 1$, there is no algorithm running in time $\Phi(k) \cdot \text{poly}(n)$ for any computable function $\Phi : \mathbb{N} \to \mathbb{N}$ that solves c-approximate k-*NCP*.*

Under ETH, a stronger hardness conjecture than W[1] $\neq$ FPT, Li, Lin, and Liu [LLL24] showed that a constant factor approximation is unattainable in time n^{k^δ} for constant $\delta > 0$.

Theorem 36 (ETH hardness of approximating k-NCP [LLL24, Corollary 4])**.** *Assuming* ETH*, for all constants $c > 1$, there is no algorithm running in time $\Phi(k) \cdot n^{k^\delta}$ for any computable function $\Phi : \mathbb{N} \to \mathbb{N}$ and $\delta = \frac{1}{\text{polylog}c}$ that solves c-approximate k-*NCP*.*

Under Gap-ETH, a stronger hardness conjecture than ETH, Manurangsi [Man20] showed the same constant factor approximation is also unattainable even in time $n^{o(k)}$.

Theorem 37 (Gap-ETH hardness of approximating k-NCP [Man20, Corollary 5])**.** *Assuming* Gap-ETH*, for all constants $c > 1$, there is no algorithm running in time $\Phi(k) \cdot n^{o(k)}$ for any computable function $\Phi : \mathbb{N} \to \mathbb{N}$ that solves c-approximate k-*NCP*.*

4.6 DT-LEARN solves the decision version of k-NCP: Proof of Theorem 34

In this section, we prove the following from which Theorem 34 follows easily.

Theorem 38 (Reducing decisional k-NCP to decision tree learning)**.** *For all $\ell \geq 2$, the following holds. Given an instance (G, z) of decisional α-approximate k-NCP over $\mathbb{F}_2^n$, there is function $g : (\mathbb{F}_2^\ell)^n \to \mathbb{F}_2$ and a distribution $\mathcal{D}$ over $(\mathbb{F}_2^\ell)^n$ such that the following holds.*

1. *One can obtain random samples from $\mathcal{D}$ labeled by g in $O(\ell n^2)$ time.*
2. *If (G, z) is a* Yes *instance of decisional α-approximate k-NCP then g is a $k\ell$-parity under $\mathcal{D}$.*
3. *If (G, z) is a* No *instance of decisional α-approximate k-NCP then g is $(\frac{1}{2} - 2^{-\Omega(\ell\alpha k)})$-far from every decision tree of size $2^{\Omega(\ell\alpha k)}$ under $\mathcal{D}$.*

4.6.1 Equivalent formulations of NCP

In proving Theorem 34, we will use the parity check view of NCP (Definition 17). The fact that this formulation is equivalent to the generator view is standard and we include it here for completeness.

Proposition 4.6.1 (Equivalence of the generator view and the parity check view of NCP)**.** *The problem in Definition 17 is equivalent to k-NCP.*

Proof. Let $G \in \mathbb{F}_2^{n \times d}$ be the generator matrix for a code $\mathcal{C}$ and $z \in \mathbb{F}_2^n$, a received message. Let $H \in \mathbb{F}_2^{(n-d)\times n}$ be such that $H^\top$ is the generator of the dual code $\mathcal{C}^\perp$. The matrix H can be efficiently computed from a generator matrix for the code $\mathcal{C}$. Furthermore, H is the parity-check matrix for $\mathcal{C}$ since $Hx = 0$ if and only if $x \in \mathcal{C}$. One can readily verify that the distance from z to $\mathcal{C}$ is k if and only if there is a k-sparse $x \in \mathbb{F}_2^n$ satisfying $Hx = Hz$. □

The parity check view also lends itself nicely to being formulated as a learning task (Definition 18). This fact is also standard and we include the equivalence for completeness.

Proposition 4.6.2 (Equivalence of parity consistency problem and NCP)**.** *The problem in Definition 18 is equivalent to the problem in Definition 17.*

Proof. Let H be the parity check matrix of a code $\mathcal{C}$ and $t \in \mathbb{F}_2^m$. The set $D = \{x^{(1)}, \ldots, x^{(m)}\}$ consisting of the rows of H and $f : D \to \mathbb{F}_2$ given by $f(x^{(i)}) = t_i$ has the property that $Hx = t$ if and only if $x^{(i)} \cdot x = t_i$ for all $i = 1, \ldots, m$. Therefore, if x is k-sparse, then f is a k-parity. Furthermore, if no k'-sparse x satisfies $Hx = t$ then f disagrees with every k'-parity on at least one point in D, and is therefore $\frac{1}{m}$-far from every such parity under $\mathrm{Unif}(D)$. □

Remark 5 (Linear independence of the vectors in D)**.** Implicit in the proof of Proposition 4.6.2 is the fact that the vectors in D can be assumed to be linearly independent. This is because the parity check matrix H is obtained by computing a basis (i.e. a set of linearly independent vectors) for the dual code $\mathcal{C}^\top$. This basis forms the rows of H which are then used to form D.

With this view in hand, we proceed with the three main steps used to prove Theorem 34.

4.6.2 Step 1: The Span operation and its properties

First, we show that we can efficiently generate random samples from the distribution $\mathrm{Unif}(\mathrm{Span}(D))$ labeled by f^{ext}.

Proposition 4.6.3 (Random samples from $\mathrm{Unif}(\mathrm{Span}(D))$ labeled by f^{ext})**.** *Given a linearly independent set of vectors $D \subseteq \mathbb{F}_2^n$ and $f : D \to \mathbb{F}_2$, random examples from $\mathrm{Unif}(\mathrm{Span}(D))$ labeled by f^{ext} can be obtained in time $O(|D|n)$.*

Proof. Let $D = \{x^{(1)}, \ldots, x^{(m)}\}$. Each $x \in \mathrm{Span}(D)$ can be written as a *unique* sum $x = \sum_{i \in I} x^{(i)}$ for $I \subseteq [m]$. Therefore, to sample a pair $(\boldsymbol{x}, f^{\mathrm{ext}}(\boldsymbol{x}))$ where $\boldsymbol{x} \sim \mathrm{Unif}(\mathrm{Span}(D))$ is uniform random, it is sufficient to sample a uniform random subset $\boldsymbol{I} \subseteq [m]$ and return $(\sum_{i \in \boldsymbol{I}} x^{(i)}, \sum_{i \in \boldsymbol{I}} f(x^{(i)}))$. □

Proof of Lemma 4.4.1

Preservation of the Yes case. Suppose that f is the parity χ_S. That is, for every $x \in D$, we have $\chi_S(x) = f(x)$. Then by linearity, we have for all $I \subseteq [m]$:

$$\chi_S\left(\sum_{i \in I} x^{(i)}\right) = \sum_{i \in I} \chi_S(x^{(i)}) = \sum_{i \in I} f(x^{(i)}).$$

This shows that $f^{\mathrm{ext}} : \mathrm{Span}(D) \to \mathbb{F}_2$ is the parity χ_S.

Amplification of the No case. For the second point, let χ_S be a k'-parity for $k' = \alpha k$. Let $A \subseteq [m]$ indicate the set of points which are misclassified by χ_S. That is, $i \in A$ if and only if $\chi_S(x^{(i)}) \neq f(x^{(i)})$. Then, $\chi_S\left(\sum_{i \in I} x^{(i)}\right) = \mathrm{Parity}(|I \cap A|) + \sum_{i \in I} f(x^{(i)})$ which shows that

$$\Pr_{\boldsymbol{I}}\left[\chi_S\left(\sum_{i \in \boldsymbol{I}} x^{(i)}\right) \neq \sum_{i \in \boldsymbol{I}} f(x^{(i)})\right] = \Pr_{\boldsymbol{I}}\Big[|\boldsymbol{I} \cap A| \text{ is odd}\Big]$$

where $\boldsymbol{I} \subseteq [m]$ is a uniform random subset of $[m]$. Since $A \neq \varnothing$ by our assumption that any k'-parity disagrees with f on at least one point, we have that $\Pr_{\boldsymbol{I}}\Big[|\boldsymbol{I} \cap A| \text{ is odd}\Big] = 1/2$. Indeed, $\boldsymbol{I}$ can equivalently be viewed as a uniform random string in $\boldsymbol{I} \in \{0,1\}^m$ denoting the characteristic vector of the set. In this case, $|\boldsymbol{I} \cap A|$ is odd if and only if the parity of the bits in the substring

$\boldsymbol{I}_A \in \{0,1\}^{|A|}$ is 1 which happens with probability 1/2 for a uniform random $\boldsymbol{I}$. This shows that χ_S disagrees with f^{ext} on 1/2 of the points in $\text{Span}(D)$ as desired. □

4.6.3 Step 2: Zero correlation with low-degree polynomials

Lemma 4.6.4. *Let $g : \mathbb{F}_2^n \to \mathbb{F}_2$ be a function and $\mathcal{D}$ be a distribution over $\mathbb{F}_2^n$. If*

$$\text{dist}_{\mathcal{D}}(g, \chi_S) = \tfrac{1}{2} \qquad \text{for every } k' \text{ parity } \chi_S$$

then,

$$\text{dist}_{\mathcal{D}}(g, h) = \tfrac{1}{2} \qquad \text{for every } h \text{ with Fourier degree } \leq k'.$$

Proof. This proof uses basic Fourier analysis. As such, it will be convenient for us to regard $g : \mathbb{F}_2^n \to \mathbb{F}_2$ as a function $g : \mathbb{F}_2^n \to \mathbb{R}$ (this is achieved by mapping $\mathbb{F}_2$ to $\mathbb{R}$ via $0 \to 1$ and $1 \to -1$). The correlation of g with any k'-parity χ_S under $\mathcal{D}$ is 0 since

$$\begin{aligned}
\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}}[g(x)\chi_S(x)] &= \Pr_{\boldsymbol{x}\sim\mathcal{D}}[g(\boldsymbol{x}) = \chi_S(\boldsymbol{x})] - \Pr_{\boldsymbol{x}\sim\mathcal{D}}[g(\boldsymbol{x}) \neq \chi_S(\boldsymbol{x})] \\
&= 1 - 2 \cdot \text{dist}_{\mathcal{D}}(g, \chi_S) \\
&= 0. && (\text{dist}_{\mathcal{D}}(g, \chi_S) = \tfrac{1}{2})
\end{aligned}$$

Therefore, the correlation under $\mathcal{D}$ between g and any $h : \mathbb{F}_2^n \to \mathbb{R}$ whose polynomial degree is at most k' is:

$$\begin{aligned}
\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}}[g(\boldsymbol{x})h(\boldsymbol{x})] &= \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}}\left[\left(\sum_{|S|\leq k'} \hat{h}(S)\chi_S(\boldsymbol{x})\right) g(\boldsymbol{x})\right] \\
&= \sum_{|S|\leq k'} \hat{h}(S) \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}}[\chi_S(\boldsymbol{x})g(\boldsymbol{x})] \\
&= 0.
\end{aligned}$$

This shows that $\text{dist}_{\mathcal{D}}(g, h) = \frac{1}{2}$ as desired. □

4.6.4 Step 3: Proof of Lemma 4.4.3

In this section, we prove Lemma 4.4.3. First, we establish some key properties of $f_{\oplus\ell}$ and $\mathcal{D}_{\oplus\ell}$ (recalling the relevant definitions from Definition 19).

Properties of blockwise parity distribution

If the distribution $\mathcal{D}$ can be efficiently sampled from, then so can the distribution $\mathcal{D}_{\oplus\ell}$. Likewise, if random samples from $\mathcal{D}$ can be labeled by f, then random samples from $\mathcal{D}_{\oplus\ell}$ can be labeled by

$f_{\oplus\ell}$. This follows directly from the definition of parity substitution Definition 19.

Fact 4.6.5 (Random samples from $\mathcal{D}_{\oplus\ell}$ labeled by $f_{\oplus\ell}$)**.** *If there is a time-t algorithm generating random samples from $\mathcal{D}$ labeled by $f : \mathbb{F}_2^n \to \mathbb{F}_2$, then there is an algorithm running in time $t+O(\ell n)$ for generating random samples from $\mathcal{D}_{\oplus\ell}$ labeled by $f_{\oplus\ell}$.*

As mentioned in the introduction, a key property of the distribution $\mathcal{D}_{\oplus\ell}$ is that it is "uniform-like" in the sense that the probability a random $\boldsymbol{y} \sim \mathcal{D}_{\oplus\ell}$ is consistent with a fixed restriction decays exponentially in the length of the restriction.

Proposition 4.6.6 (Formal version of Proposition 4.4.2)**.** *Let $R \subseteq [n\ell]$ be a subset of coordinates and $r \in \mathbb{F}_2^{|R|}$. Then,*

$$\Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[\boldsymbol{y}_R = r] \leq 2^{-|R|(1-1/\ell)}.$$

Proof. For $i \in [n]$, let $R^{(i)} \subseteq [\ell]$ denote the ith block of R, that is the subset of coordinates of the ith block restricted by R. Let $r^{(i)}$ denote the corresponding substring of r so that $r = (r^{(1)}, \ldots, r^{(n)})$. We observe that for all $x \in \mathbb{F}_2^n$ for which $\Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}(x)}[\boldsymbol{y}_{R^{(i)}} = r^{(i)}]$ is nonzero:

$$\Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}(x)}[\boldsymbol{y}_{R^{(i)}} = r^{(i)}] = \begin{cases} 2^{-|R^{(i)}|} & |R^{(i)}| < \ell \\ 2^{-|R^{(i)}|+1} & |R^{(i)}| = \ell \end{cases}.$$

If $|R^{(i)}| < \ell$, then the probability $\boldsymbol{y}_{R^{(i)}} = r^{(i)}$ is exactly $2^{-|R^{(i)}|}$: any subset of $\ell - 1$ coordinates of the ith block of $\boldsymbol{y}$ is distributed uniformly at random. In the other case, $R^{(i)}$ consists of the entire ith block, in which case $\ell - 1$ bits are distributed uniformly at random while the last bit is set according to x. In either case, we can write $\Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}(x)}[\boldsymbol{y}_{R^{(i)}} = r^{(i)}] \leq 2^{-|R^{(i)}|+|R^{(i)}|/\ell}$. Finally, we have

$$\begin{aligned}
\Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[\boldsymbol{y}_R = r] &= \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}}\left[\Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}(\boldsymbol{x})}[\boldsymbol{y}_R = r]\right] \\
&= \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}}\left[\prod_{i\in[n]} \Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}(\boldsymbol{x})}[\boldsymbol{y}_{R^{(i)}} = r^{(i)}]\right] \\
&\qquad \text{(Independence of the blocks of } \boldsymbol{y} \text{ conditioned on } \boldsymbol{x}) \\
&\leq \prod_{i\in[n]} 2^{-|R^{(i)}|+|R^{(i)}|/\ell} \\
&= 2^{-|R|(1-1/\ell)} \qquad \text{(Definition of } R^{(i)})
\end{aligned}$$

which completes the proof. □

A simple lemma about parity substitution

For the next lemma, we switch to viewing a Boolean function as a mapping $g : \mathbb{F}_2^n \to \{\pm 1\}$.

Lemma 4.6.7. *Let $g : \mathbb{F}_2^n \to \{\pm 1\}$ and $\mathcal{D}$ be a distribution over $\mathbb{F}_2^n$. Consider $g_{\oplus\ell} : (\mathbb{F}_2^\ell)^n \to \{\pm 1\}$ and $\mathcal{D}_{\oplus\ell}$. We say that $S \subseteq [\ell n]$ is block-complete if there is a set $S^\star \subseteq [n]$ such that S contains all the coordinates in the blocks specified by $S^\star$ and no more. (This in particular implies that $|S^\star| = |S|/\ell$.) Then*

$$\Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[g_{\oplus\ell}(\boldsymbol{y}) = \chi_S(\boldsymbol{y})] = \begin{cases} \Pr_{\boldsymbol{x}\sim\mathcal{D}}[g(\boldsymbol{x}) = \chi_{S^\star}(\boldsymbol{x})] & \text{if } S \text{ is block-complete} \\ \frac{1}{2} & \text{otherwise.} \end{cases}$$

Proof. First, suppose S is block-complete. Then, the lemma follows simply by unpacking the definitions of $D_{\oplus\ell}$ and $g_{\oplus\ell}$. We will therefore assume that S is not block-complete.

Let $S^{(i)}$ be the intersection of S and the ith block. Note that $S = \cup_{i=1}^n S^{(i)}$. For $i \in [n]$ and $x \in \mathbb{F}_2^n$, let $\mathcal{D}_{\oplus\ell}^{(i)}(x)$ denote the distribution of $\boldsymbol{y}^{(i)}$ when $\boldsymbol{y} \sim \mathcal{D}_{\oplus\ell}(x)$. We make the following key observation: if there is an $i^* \in [n]$ such that $|S^{(i^*)}| < \ell$, then for every fixed x,

$$\mathop{\mathbb{E}}_{\boldsymbol{y}^{(i^*)}\sim\mathcal{D}_{\oplus\ell}^{(i^*)}(x)}[\chi_{S^{(i^*)}}(\boldsymbol{y}^{(i^*)})] = 0.$$

This follows from the fact that the subset of $\boldsymbol{y}^{(i^*)}$ with indices in $S^{(i^*)}$ is a uniform random string, so its parity will be a uniform random bit. Note that such an i^* exists if and only if S is not block-complete. We will now show that $g_{\oplus\ell}(\boldsymbol{y})$ and $\chi_S(\boldsymbol{y})$ have 0 correlation:

$$\begin{aligned}
\mathop{\mathbb{E}}_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[g_{\oplus\ell}(\boldsymbol{y})\chi_S(\boldsymbol{y})] &= \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}}\left[\mathop{\mathbb{E}}_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}(\boldsymbol{x})}[g_{\oplus\ell}(\boldsymbol{y})\chi_S(\boldsymbol{y})]\right] && \text{(Definition of } \mathcal{D}_{\oplus\ell}) \\
&= \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}}\left[g(\boldsymbol{x})\mathop{\mathbb{E}}_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}(\boldsymbol{x})}[\chi_S(\boldsymbol{y})]\right] && \text{(Definition of } g_{\oplus\ell}) \\
&= \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}}\left[g(\boldsymbol{x})\mathop{\mathbb{E}}_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}(\boldsymbol{x})}\left[\prod_{i=1}^n \chi_{S^{(i)}}(\boldsymbol{y}^{(i)})\right]\right] && \text{(Definition of } S^{(i)}) \\
&= \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}}\left[g(\boldsymbol{x})\prod_{i=1}^n \mathop{\mathbb{E}}_{\boldsymbol{y}^{(i)}\sim\mathcal{D}_{\oplus\ell}^{(i)}(\boldsymbol{x})}[\chi_{S^{(i)}}(\boldsymbol{y}^{(i)})]\right] \\
& && \text{(Independence of } \boldsymbol{y}^{(i)} \text{ conditioned on } \boldsymbol{x}) \\
&= 0. && \text{(Assumption that } S \text{ is not block-complete)}
\end{aligned}$$

The last equality follows from our key observation because S is not block-complete, there is some $i^* \in [n]$ such that $|S^{(i^*)}| < \ell$. This shows that $\Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[g_{\oplus\ell}(\boldsymbol{y}) = \chi_S(\boldsymbol{y})] = \frac{1}{2}$ as desired. □

Corollary 4.6.8. *If $g : \mathbb{F}_2^n \to \{\pm 1\}$ is $\frac{1}{2}$-far under $\mathcal{D}$ from all k'-parities, then for all $\ell \geq 1$, $g_{\oplus\ell} : (\mathbb{F}_2^\ell)^n \to \{\pm 1\}$ is $\frac{1}{2}$-far under $\mathcal{D}_{\oplus\ell}$ from every function of Fourier degree $\ell k'$.*

Proof. We observe that $g_{\oplus\ell} : (\mathbb{F}_2^\ell)^n \to \{\pm 1\}$ is $\frac{1}{2}$-far under $\mathcal{D}_{\oplus\ell}$ from $\ell k'$-parities. This is because,

by Lemma 4.6.7, for every $\ell k'$ parity χ_S there is a set $S^\star$ of size $\leq k'$ such that:

$$\Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[g_{\oplus\ell}(\boldsymbol{y}) = \chi_S(\boldsymbol{y})] = \begin{cases} \Pr_{\boldsymbol{x}\sim\mathcal{D}}[g(\boldsymbol{x}) = \chi_{S^\star}(\boldsymbol{x})] = \frac{1}{2} & \text{if } S \text{ is block-complete} \\ \frac{1}{2} & \text{otherwise} \end{cases}$$

where we used the assumption that g is $1/2$-far under $\mathcal{D}$ from all k'-parities. The corollary then follows directly from Lemma 4.6.4. □

Proof of Lemma 4.4.3

We now prove the main lemma showing that parity substitution lifts decision tree depth lower bounds to size lower bounds.

Lemma 4.6.9 (Generalization of Lemma 4.4.3)**.** *Let $\mathcal{D}$ be a distribution over $\mathbb{F}_2^n$ and $g : \mathbb{F}_2^n \to \mathbb{F}_2$. For every $\ell \geq 2$, the distribution $\mathcal{D}_{\oplus\ell}$ and the function $g_{\oplus\ell} : (\mathbb{F}_2^\ell)^n \to \mathbb{F}_2$ satisfy the following:*

1. *If g is a k-parity under $\mathcal{D}$, then $g_{\oplus\ell}$ is a $k\ell$-parity under $\mathcal{D}_{\oplus\ell}$*
2. *If g is $\frac{1}{2}$-far under $\mathcal{D}$ from every degree-k' polynomial, then $g_{\oplus\ell}$ is $(\frac{1}{2} - 2^{-\ell k'/6})$-far under $\mathcal{D}_{\oplus\ell}$ from every decision tree of size $2^{\ell k'/3}$.*

The proof of Lemma 4.6.9 uses the following claim.

Claim 4.6.10 (Pruning the depth of a decision tree)**.** *Let T be a size-s decision tree and $c \in \mathbb{N}$ a parameter. Let T' be the decision tree obtained from T by pruning each path at depth $c\log(s)$. Then, for all $\ell \geq 1$, $\mathrm{dist}_{\mathcal{D}_{\oplus\ell}}(T', g_{\oplus\ell}) \leq \mathrm{dist}_{\mathcal{D}_{\oplus\ell}}(T, g_{\oplus\ell}) + s^{1-c(1-1/\ell)}$.*

Proof. Let Π denote the set of paths in T which have been pruned. The size of Π is at most s. First, we bound the probability that a random input follows a path in Π:

$$\begin{aligned} \Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[\boldsymbol{y} \text{ follows a path in } \Pi] &\leq \sum_{\pi\in\Pi} \Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[\boldsymbol{y} \text{ follows } \pi] && \text{(Union bound)} \\ &\leq \sum_{\pi\in\Pi} 2^{-c\log(s)(1-1/\ell)} && \text{(Proposition 4.6.6 and } |\pi| \geq c\log(s)) \\ &\leq s^{1-c(1-1/\ell)}. && (|\Pi| \leq s) \end{aligned}$$

Therefore:

$$\begin{aligned} \mathrm{dist}_{\mathcal{D}_{\oplus\ell}}(T', g_{\oplus\ell}) &\leq \mathrm{dist}_{\mathcal{D}_{\oplus\ell}}(T, g_{\oplus\ell}) + \Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[\boldsymbol{y} \text{ follows a path in } \Pi] && \text{(Union bound)} \\ &\leq \mathrm{dist}_{\mathcal{D}_{\oplus\ell}}(T, g_{\oplus\ell}) + s^{1-c(1-1/\ell)} \end{aligned}$$

which completes the proof. □

Proof of Lemma 4.6.9. We prove each point separately.

1. Let $S \subseteq [n]$ denote the k indices of the parity consistent with g under $\mathcal{D}$. Then,

$$g_{\oplus\ell}(y) = g(\text{BlockwisePar}(y)) = \bigoplus_{i \in S} \oplus y^{(i)}$$

is a $k\ell$-parity under $\mathcal{D}_{\oplus\ell}$.

2. We prove this statement by contradiction. Let T be a decision tree of size $2^{\ell k'/3}$ achieving small error: $\text{dist}_{\mathcal{D}_{\oplus\ell}}(T, g_{\oplus\ell}) < \frac{1}{2} - 2^{-\ell k'/6}$. Let T' be the decision tree obtained by pruning each path of T at depth $\ell k'$. Then,

$$\begin{aligned}
\text{dist}_{\mathcal{D}_{\oplus\ell}}(T', g_{\oplus\ell}) &\leq \text{dist}_{\mathcal{D}_{\oplus\ell}}(T, g_{\oplus\ell}) + (2^{\ell k'/3})^{1-3(1-1/\ell)} && \text{(Claim 4.6.10)} \\
&\leq \text{dist}_{\mathcal{D}_{\oplus\ell}}(T, g_{\oplus\ell}) + 2^{-\ell k'/6} && (\ell \geq 2) \\
&< \tfrac{1}{2}. && (\text{dist}_{\mathcal{D}_{\oplus\ell}}(T, g_{\oplus\ell}) < \tfrac{1}{2} - 2^{-\ell k'/6})
\end{aligned}$$

Since T' is a decision tree of depth $\ell k'$, it is a polynomial of degree $\ell k'$. However, since g is $\frac{1}{2}$-far from polynomials of degree k', we know that $g_{\oplus\ell}$ is $\frac{1}{2}$-far from polynomials of degree $\ell k'$ by Corollary 4.6.8. Therefore, we have reached a contradiction and conclude that $g_{\oplus\ell}$ must be $(\frac{1}{2} - 2^{-\ell k'/6})$-far from decision trees of size $2^{\ell k'/3}$. □

4.6.5 Putting things together: Proof of Theorem 38

Let $(H, t) \in \mathbb{F}_2^{m \times n} \times \mathbb{F}_2^m$ be an instance of decisional α-approximate k-NCP where H is the parity check matrix for the code $\mathcal{C}$. Let $D = \{x^{(1)}, \ldots, x^{(m)}\} \subseteq \mathbb{F}_2^n$ be the set corresponding to the rows of the parity check matrix H and $f : D \to \mathbb{F}_2$ be the function labeling the set according to t, $f(x^{(i)}) = t_i$. Let $\mathcal{D}$ be the distribution $\text{Unif}(\text{Span}(D))_{\oplus\ell}$. That is, $\mathcal{D}$ is the distribution obtained by substituting a parity of size ℓ into $\text{Unif}(\text{Span}(D))$. Let $f^{\text{ext}} : \text{Span}(D) \to \mathbb{F}_2$ be the linear extension of f to $\text{Span}(D)$. We prove the theorem for the function $(f^{\text{ext}})_{\oplus\ell}$. We split into cases.

Yes case: there is a k-sparse vector x such that $Hx = t$. We obtain the desired result from the following chain of observations

1. $f : D \to \mathbb{F}_2$ is a k-parity (assumption of the YES case and Definition 18)
2. ...which implies f^{ext} is a k-parity under $\text{Unif}(\text{Span}(D))$ (Lemma 4.4.1)
3. ...which implies $(f^{\text{ext}})_{\oplus\ell}$ is a ℓk-parity under $\text{Unif}(\text{Span}(D))_{\oplus\ell}$ (Lemma 4.6.9).

No case: $Hx \neq t$ for all vectors x of sparsity at most αk. In this case, we make the following observations

1. $f : D \to \mathbb{F}_2$ is disagrees with every αk-parity on some $x \in D$ (assumption of the No case and Definition 18)
2. ...which implies that $\mathrm{dist}_{\mathrm{Unif}(\mathrm{Span}(D))}(f^{\mathrm{ext}}, \chi_S) = \frac{1}{2}$ with every αk-parity (Lemma 4.4.1)
3. ...which implies $(f^{\mathrm{ext}})_{\oplus \ell}$ is $\frac{1}{2}$-far from every function of Fourier degree at most $\ell \alpha k$ under $\mathrm{Unif}(\mathrm{Span}(D))_{\oplus \ell}$ (Corollary 4.6.8)
4. ...which implies $(f^{\mathrm{ext}})_{\oplus \ell}$ is $(1/2 - 2^{-\Omega(\alpha \ell k)})$-far under $\mathrm{Unif}(\mathrm{Span}(D))_{\oplus \ell}$ from every decision tree of size $2^{O(\alpha \ell k)}$. (Lemma 4.6.9)

Finally, we remark that by Proposition 4.6.3, random samples from $\mathrm{Unif}(\mathrm{Span}(D))$ labeled by f^{ext} can be efficiently generated and therefore by Fact 4.6.5, so can random samples from $\mathrm{Unif}(\mathrm{Span}(D))_{\oplus \ell}$ labeled by $(f^{\mathrm{ext}})_{\oplus \ell}$. □

4.6.6 Proof of Theorem 33

Assume there is an algorithm $\mathcal{A}$ for solving the junta testing problem described in the theorem statement. We use $\mathcal{A}$ to solve NCP. Let (G, z) be an instance of $2^{\Theta(\log n / \log \log n)}$-approximate NCP with distance parameter k and let $f : \{0,1\}^{2n} \to \{0,1\}$ and $\mathcal{D}$ be the function and distribution from Theorem 38. We run $\mathcal{A}$ for $t(2n)$ time steps with $r = 2k$ and output "Yes" if and only if the algorithm outputs "Yes". Theorem 38 ensures that this reduction is efficient, as we can efficiently answer queries to f and obtain random samples from $\mathcal{D}$.

Correctness. Theorem 38 ensures that the reduction is correct. We expand on the Yes/No cases separately.

Yes case: If (G, z) is a Yes instance of $2^{\Theta(\log n / \log \log n)}$-approximate NCP, then f is a $2k$-parity under $\mathcal{D}$ and $\mathcal{A}$ outputs Yes w.h.p.

No case: If (G, z) is a No instance of $2^{\Theta(\log n / \log \log n)}$-approximate NCP, then f is $\frac{1}{2}$-far from every function of Fourier degree $k \cdot 2^{\Theta(\log n / \log \log n)}$ under $\mathcal{D}$ (see Lemma 4.6.4). By the inclusion $\{k' \text{ juntas}\} \subseteq \{\text{degree-}k' \text{ polynomials}\}$, this implies that f is $\frac{1}{2}$-far from every $k \cdot 2^{\Theta(\log n / \log \log n)}$ junta under $\mathcal{D}$. Therefore, $\mathcal{A}$ outputs No w.h.p.

Since $2^{\Theta(\log n / \log \log n)}$-approximate NCP is NP-hard by [DKS98a], this reduction yields an algorithm solving SAT in $\mathrm{RTIME}(t(\mathrm{poly}(n)))$. □

4.6.7 Proof of Theorem 34

Let $\mathcal{A}$ be the decision tree learning algorithm from the theorem statement.

The reduction. Let $(H, t) \in \mathbb{F}_2^{m \times n} \times \mathbb{F}_2^m$ be an instance of decisional α-approximate k-NCP where H is the parity check matrix for the code $\mathcal{C}$. Using Theorem 38, we obtain a function $g : (\mathbb{F}_2^\ell)^n \to \mathbb{F}_2$ and a distribution $\mathcal{D}$ over $(\mathbb{F}_2^\ell)^n$. We run the algorithm $\mathcal{A}$ on g and $\mathcal{D}$ for $t(\ell n, 2^{\ell k}, 2^{O(\alpha \ell k)}, \varepsilon)$ time steps for $\varepsilon = \frac{1}{2} - 2^{-\Omega(\alpha \ell k)}$. Let T_{hyp} be the decision tree learned by $\mathcal{A}$. We compute an estimate, $\overline{\varepsilon}$, of the quantity $\text{dist}_{\mathcal{D}}(g, T_{\text{hyp}})$ to accuracy $\pm 2^{-\Omega(\alpha \ell k)}$ using an additional $2^{O(\alpha \ell k)}$ samples from $\mathcal{D}$ labeled by g. We return "Yes" if $\overline{\varepsilon} \leq \frac{1}{2} - 2^{-\Omega(\alpha \ell k)}$ and $|T_{\text{hyp}}| \leq 2^{O(\alpha \ell k)}$, and "No" otherwise.[4]

Runtime. Random samples from $\mathcal{D}$ labeled by g can be obtained in $O(\ell n^2)$-time. We simulate $\mathcal{A}$ for $t(\ell n, 2^{\ell k}, 2^{O(\alpha \ell k)}, \varepsilon)$ time steps and estimating $\overline{\varepsilon}$ takes time $\text{poly}(n, \ell, 2^{\alpha \ell k})$. So the overall runtime of the reduction is $O(\ell n^2) \cdot t(\ell n, 2^{\ell k}, 2^{O(\alpha \ell k)}, \varepsilon) + \text{poly}(n, \ell, 2^{\alpha \ell k})$.

Correctness. To prove correctness, we show that if (H, t) is a Yes instance of decisional α-approximate k-NCP, then we output Yes with high probability, and otherwise if (H, t) is a No instance then our algorithm outputs No with high probability.

Yes case: there is a k sparse vector x such that $Hx = t$. In this case, g is a parity of at most $k\ell$ variables under $\mathcal{D}$ by Theorem 38. Therefore, g is a decision tree of size $2^{k\ell}$ under $\mathcal{D}$. By running $\mathcal{A}$ for $t(\ell n, 2^{\ell k}, 2^{O(\alpha \ell k)}, \varepsilon)$ time steps, we obtain a decision tree T_{hyp} of size $|T_{\text{hyp}}| \leq 2^{O(\alpha \ell k)}$ which satisfies

$$\text{dist}_{\mathcal{D}}(g, T_{\text{hyp}}) \leq \varepsilon = \frac{1}{2} - 2^{-\Omega(\alpha \ell k)}$$

and therefore our estimate $\overline{\varepsilon}$ satisfies

$$\overline{\varepsilon} \leq \text{dist}_{\mathcal{D}}(g, T_{\text{hyp}}) + 2^{-\Omega(\alpha \ell k)} \leq \frac{1}{2} - 2^{-\Omega(\alpha \ell k)}$$

with high probability which ensures that our algorithm correctly outputs "Yes."

No case: $Hx \neq t$ for all vectors x of sparsity at most αk. First, if T_{hyp} does not satisfy $|T_{\text{hyp}}| \leq 2^{O(\alpha \ell k)}$ then our algorithm correctly outputs "No". Otherwise, assume that $|T_{\text{hyp}}| \leq 2^{O(\alpha \ell k)}$. We will show that T_{hyp} must have large error so that in this case our algorithm also correctly outputs "No".

Theorem 38 implies that g is $\frac{1}{2} - 2^{-\Omega(\alpha k \ell)}$ far under $\mathcal{D}$ from every decision tree of size $2^{O(\alpha k \ell)}$. This implies that $\text{dist}_{\mathcal{D}}(g, T_{\text{hyp}}) > \frac{1}{2} - 2^{-\Omega(\alpha \ell k)}$. Therefore, our estimate $\overline{\varepsilon}$ satisfies

$$\overline{\varepsilon} \geq \text{dist}_{\mathcal{D}}(g, T_{\text{hyp}}) - 2^{-\Omega(\alpha \ell k)} > \frac{1}{2} - 2^{-\Omega(\alpha \ell k)}$$

with high probability. This ensures that our algorithm correctly outputs "No". □

[4] Concretely, the constants hidden by the big-O notation are the following. If $\beta = 1/2 - 2^{-\Omega(\alpha k \ell)}$ is the the error in the No case of Theorem 38, we require the learner output a hypothesis with error $\varepsilon = \frac{1}{2} - 2^{-c\alpha \ell k}$ where c is a constant chosen so that $\varepsilon < \beta$. Then, we estimate $\text{dist}_{\mathcal{D}}(g, T_{\text{hyp}})$ to accuracy $\pm 2^{-C\alpha \ell k}$ where C is a large enough constant such that $\varepsilon + 2^{-C\alpha \ell k} < \beta$. Finally, we "Yes" if and only if $\overline{\varepsilon} \leq \varepsilon + 2^{-C\alpha \ell k}$ and $|T_{\text{hyp}}| \leq 2^{O(\alpha \ell k)}$.

4.7 DT-Learn solves the search version of k-NCP: Proof of Theorem 32

Claim 4.7.1 (Solving the search version of k-NCP given a decision tree)**.** *Let $(H, t) \in \mathbb{F}_2^{m \times n} \times \mathbb{F}_2^m$ be an instance of* NCP *where H is the parity check matrix for the linear code, and let $D = \{y^{(1)}, \dots, y^{(m)}\}$ be the set corresponding to the rows of the parity check matrix H.*

Let $f : D \to \mathbb{F}_2$ be the function satisfying $f(y^{(i)}) = t_i$ for $i \in [m]$, $\mathcal{D}$ be the distribution $\mathrm{Unif}(\mathrm{Span}(D))_{\oplus \ell}$, and T be a size-s decision tree satisfying $\mathrm{dist}_{\mathcal{D}}(T, (f^{\mathrm{ext}})_{\oplus \ell}) \le \frac{1}{2} - \gamma$ where $\gamma \ge \Omega(s^{1-c(1-1/\ell)})$ for some $c \in \mathbb{N}$.

There is an algorithm running in time $\mathrm{poly}(n, \ell, 1/\gamma^2, s)$ which outputs with high probability a set of coordinates $S \subseteq [n]$ such that $|S| \le \frac{c \log s}{\ell}$ and $\chi_S(y) = f(y)$ for all $y \in D$.

Before proving the claim, we prove two helpful lemmas.

Lemma 4.7.2 (Extracting a well-correlated parity from a decision tree)**.** *Let T be a depth-d decision tree satisfying $\mathrm{dist}_{\mathcal{D}}(T, g) \le \frac{1}{2} - \gamma$ for some $\gamma > 0$, distribution $\mathcal{D}$ over $\mathbb{F}_2^n$, and $g : \mathbb{F}_2^n \to \mathbb{F}_2$. Then, there is a $\mathrm{poly}(n, 1/\gamma^2, 2^d)$-time algorithm which uses $2^{O(d)}/\gamma^2$ random samples from $\mathcal{D}$ labeled by g and with high probability outputs set of coordinates $S \subseteq [n]$ such that $|S| \le d$ and $\mathrm{dist}_{\mathcal{D}}(\chi_S, g) \le \frac{1}{2} - \Theta(\gamma 4^{-d})$.*

The proof of Lemma 4.7.2 relies on the following properties of the Fourier spectrum of decision trees.

Fact 4.7.3 (Fourier spectrum of decision trees)**.** *Let T be a depth-d decision tree on n variables. Then, the following properties hold.*

1. *If a Fourier coefficient of T, $\hat{T}(S)$, for $S \subseteq [n]$ is nonzero then S consists of coordinates queried along some path in T.*
2. *The number of nonzero Fourier coefficients is at most 4^d.*

Property 2 in Fact 4.7.3 follows immediately from property 1. A good reference for these properties is [O'D14, Section 3.2].

Proof of Lemma 4.7.2. We show the following algorithm proves the lemma:

1. Draw $2^{O(d)}/\gamma^2$ random samples from $\mathcal{D}$ labeled by g.
2. For every $S \subseteq [n]$ consisting of coordinates queried along some path in T, use the random samples to estimate $\mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}}[\chi_S(\boldsymbol{x}) g(\boldsymbol{x})]$.
3. Output the subset S corresponding to the parity χ_S which is most well-correlated with g over $\mathcal{D}$.

There are at most 4^d subsets $S \subseteq [n]$ to check in step (2). Therefore, the runtime of this algorithm is $\mathrm{poly}(n, 1/\gamma^2, 2^d)$. It remains to prove correctness.

Rewriting the assumption that $\Pr_{\boldsymbol{x}\sim\mathcal{D}}[T(\boldsymbol{x}) \neq g(\boldsymbol{x})] \leq \frac{1}{2} - \gamma$ in terms of correlation, we have

$$\begin{aligned}\gamma &\leq \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}}[T(\boldsymbol{x})g(\boldsymbol{x})] \\ &\leq \sum_{S\subseteq[n]} \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}}[\hat{T}(S)\chi_S(\boldsymbol{x})g(\boldsymbol{x})]. && \text{(Fourier expansion of } T\text{)}\end{aligned}$$

By Fact 4.7.3, the number of nonzero Fourier coefficients $\hat{T}(S)$ is at most 4^d and therefore, there is some $S \subseteq [n]$ such that

$$\begin{aligned}\frac{\gamma}{4^d} &\leq \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}}[\hat{T}(S)\chi_S(\boldsymbol{x})g(\boldsymbol{x})] \\ &\leq \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}}[\chi_S(\boldsymbol{x})g(\boldsymbol{x})]. && (\hat{T}(S) \leq 1)\end{aligned}$$

Moreover, this S consists of coordinates queried along some path in T by Fact 4.7.3. Using $2^{O(d)}/\gamma^2$ random samples from $\mathcal{D}$ labeled by g, the correlation $\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[\chi_S(\boldsymbol{x})g(\boldsymbol{x})]$ can be estimated to within an additive accuracy of $\Theta(\frac{\gamma}{4^d})$ with a failure probability of $2^{-\Theta(d)}$. By a union bound over all 4^d subsets $S \subseteq [n]$ that the algorithm checks, all correlation estimates are within the desired accuracy bounds, and the algorithm successfully outputs a parity which achieves accuracy $1/2 + \Theta(\gamma 4^{-d})$ in approximating g over $\mathcal{D}$. □

Lemma 4.7.4 (Obtaining a zero-error parity for f from a well-correlated parity for $(f^{\mathrm{ext}})_{\oplus\ell}$)**.** *Let $\mathcal{D}$ and $(f^{\mathrm{ext}})_{\oplus\ell} : (\mathbb{F}_2^\ell)^n \to \mathbb{F}_2$ be as in the statement of Claim 4.7.1. If there is a parity χ_S for $S \subseteq [\ell n]$ such that $\mathrm{dist}_{\mathcal{D}}(\chi_S, (f^{\mathrm{ext}})_{\oplus\ell}) \leq \frac{1}{2} - \gamma$ for $\gamma > 0$, then $\chi_{S^\star}(y) = f(y)$ for all $y \in D$ where $|S^\star| \leq |S|/\ell$ and $S^\star$ consists of the coordinates $i \in [n]$ such that the ith block in S is nonempty.*

Proof. Lemma 4.6.7 states that there is a parity $S^\star$ of size $|S|/\ell$ satisfying $\mathrm{dist}_{\mathrm{Unif}(\mathrm{Span}(D))}(\chi_{S^\star}, f^{\mathrm{ext}}) \leq \frac{1}{2} - \gamma$. Further, $S^\star$ consists of the coordinates $i \in [n]$ such that the ith block in S is nonempty. Finally, the contrapositive of the no case in Lemma 4.4.1 implies that $\chi_{S^\star}(y) = f(y)$ for all $y \in D$. Indeed, if it were the case that $\chi_{S^\star}$ disagrees with f on some input $y \in D$, then Lemma 4.4.1 shows that $\mathrm{dist}_{\mathrm{Unif}(\mathrm{Span}(D))}(\chi_{S^\star}, f^{\mathrm{ext}}) = \frac{1}{2}$ which contradicts our assumption on the error of $\chi_{S^\star}$.[5] □

[5] We are using the fact that implicit in the proof of Lemma 4.4.1 is the following: for any parity χ_S, if χ_S disagrees with f on at least one $x \in D$, then χ_S disagrees with f^{ext} on exactly half of the inputs from $\mathrm{Span}(D)$.

4.7.1 Proof of Claim 4.7.1

First, we prune all paths in T at depth $c \log s$ to obtain a tree T'. Claim 4.6.10 ensures that this process doesn't increase the error of T' too much:

$$\begin{aligned} \mathrm{dist}_{\mathcal{D}}(T', (f^{\mathrm{ext}})_{\oplus \ell}) &\le \mathrm{dist}_{\mathcal{D}}(T, (f^{\mathrm{ext}})_{\oplus \ell})) + s^{1-c(1-1/\ell)} && \text{(Claim 4.6.10)} \\ &\le \frac{1}{2} - \gamma + s^{1-c(1-1/\ell)}. && \text{(Assumption on } T\text{)} \end{aligned}$$

After pruning, T' has depth small enough that, in polynomial time, we can apply Lemma 4.7.2 to obtain a well-correlated parity χ_S of size $\le c \log s$. The error of this parity is bounded:

$$\mathrm{dist}_{\mathcal{D}}(\chi_S, (f^{\mathrm{ext}})_{\oplus \ell}) \le \frac{1}{2} - \Theta\left(\frac{\gamma - s^{1-c(1-1/\ell)}}{s^{2c}}\right). \tag{4.1}$$

By our assumption that $\gamma \ge \Omega(s^{1-c(1-1/\ell)})$, Equation (4.1) can be rewritten as $\mathrm{dist}_{\mathcal{D}}(\chi_S, (f^{\mathrm{ext}})_{\oplus \ell}) \le \frac{1}{2} - \gamma'$ for some $\gamma' > 0$. Therefore, Lemma 4.7.4 implies that we can find a parity $S^\star$ of size $\le \frac{c \log s}{\ell}$ such that $\chi_{S^\star}(y) = f(y)$ for all $y \in D$ as desired. □

4.7.2 Proof of Theorem 32

By Theorem 38, for any $\alpha > 1$, given an NCP instance where the nearest codeword is within distance k of the received word, there is an algorithm running in time $O(\ell n) \cdot t(\ell n, 2^{\ell k}, 2^{O(\alpha \ell k)}, \varepsilon)$ for $\varepsilon = \frac{1}{2} - 2^{-\Omega(\alpha \ell k)}$ which outputs a decision tree of size $2^{O(\alpha \ell k)}$ for $(f^{\mathrm{ext}})_{\oplus \ell}$ and has error ε in computing $(f^{\mathrm{ext}})_{\oplus \ell}$ over $\mathcal{D} = \mathrm{Unif}(\mathrm{Span}(D))_{\oplus \ell}$. Therefore, by Claim 4.7.1 we can extract a parity of size $|S| \le \alpha k$ which is consistent with f over D. Equivalently, we have found a codeword within distance αk of the received word as desired. Since this extraction step requires an additional $\mathrm{poly}(n, \ell, 2^{\alpha \ell k})$ time, the proof is completed. □

4.8 Proofs of corollaries

4.8.1 Proof of Corollary 4.2.1

Let $\mathcal{A}$ be the learner from the corollary statement. Using Theorem 32 with $\ell = 2$, we show that $\mathcal{A}$ solves $O(\log n)$-approximate k-NCP. Given a decision tree target of size 2^{2k} and random labeled examples from $\mathcal{D}$, $\mathcal{A}$ runs in time $n^{o(k)}$ and outputs a decision tree hypothesis with accuracy $\frac{1}{2} + \frac{1}{\mathrm{poly}(n)}$. If $\alpha = O(\log n)$, then the size of the decision tree hypothesis is at most $n^{o(k)} \le 2^{O(\alpha k)}$ and the error of the hypothesis satisfies $\varepsilon = \frac{1}{2} - \frac{1}{\mathrm{poly}(n)} \le \frac{1}{2} - 2^{-\Omega(\alpha k)}$. Therefore, Theorem 32 shows that $\mathcal{A}$ solves $O(\log n)$-approximate k-NCP for $k = \Theta(\log s)$.

4.8.2 Proof of Corollary 4.2.2

Suppose for contradiction there is a learner $\mathcal{A}$ satisfying the constraints of the corollary statement. We will use $\mathcal{A}$ to solve c-approximate k-NCP for some constant $c > 1$ in randomized polynomial-time. By Theorem 35, this implies that there is a randomized FPT algorithm for all of W[1].

Let c' be a constant so that $\mathcal{A}$ runs in time $n^{c'}$ when given random examples from $\mathcal{D}$ labeled by a size-n decision tree and outputs a hypothesis with error $\frac{1}{2} - \frac{1}{n^{c'}}$ under $\mathcal{D}$. Let c be a sufficiently large constant relative to c' (to be chosen later). We will use $\mathcal{A}$ to solve c-approximate k-NCP over $\mathbb{F}_2^n$. We assume that n is large enough so that $\log n \geq k$. Let $\ell = (\log n)/k$. Given a decision tree target of size $2^{\ell k} = n$, $\mathcal{A}$ runs in time $n^{c'}$ and outputs a decision tree hypothesis of size at most $n^{c'} \leq 2^{O(c\ell k)} = n^{O(c)}$, assuming c is a large enough. Likewise, the error of the hypothesis is at most $\frac{1}{2} - n^{-c'} \leq \frac{1}{2} - 2^{-\Omega(c\ell k)} = \frac{1}{2} - n^{-\Omega(c)}$, again assuming that c is large enough. By Theorem 32, this shows that $\mathcal{A}$ solves c-approximate k-NCP in $\mathrm{poly}(n)$ time as desired.

4.8.3 Proof of Corollary 4.2.3

Let $\mathcal{A}$ be the learner from the corollary statement. Let $c' > 1$ be a constant such that $\mathcal{A}$ learns decision tree targets of size n with decision tree hypotheses of size $n^{c'}$. We start by proving ETH hardness.

ETH hardness. Combining [LLL24, Theorem 1] and [LRSW22, Theorem 11] yields the following reduction from 3-SAT to k-NCP.

Theorem 39 (Reduction from solving 3-SAT exactly to approximating k-NCP)**.** *For all constant $c > 1$, there is a constant $q > 1$ such that for all $k \in \mathbb{N}$ the following holds. There is a reduction running in time $\mathrm{poly}(m, 2^{m/k}) + \mathrm{poly}(m, 2^k)$ which maps* 3-SAT *instances φ consisting of m clauses to* NCP *instances (G, z) of size $\mathrm{poly}(m, 2^{m/k})$ such that*

- Yes *case: if φ is satisfiable then (G, z) is a* Yes *instance of c-approximate k^q-*NCP*;*
- No *case: if φ is not satisfiable, then (G, z) is a* No *instance of c-approximate k^q-*NCP*.*

Using Theorem 39, we show how to refute randomized ETH if $\mathcal{A}$ runs in time $n^{(\log n)^\delta}$ for sufficiently small $\delta > 0$. Let φ be a 3-SAT instance on n variables with m clauses. By Theorem 39, for a constant $c > 1$ (which is sufficiently larger than c' and is chosen later), there is a constant $q > 1$ such that the reduction holds for all $k \in \mathbb{N}$. Let $k = m^\lambda$ for any $0 < \lambda < 1/(2q)$ and let $(H, t) \in \mathbb{F}_2^{M \times N} \times \mathbb{F}_2^M$ for $M + N = \mathrm{poly}(m, 2^{m/k}) = 2^{O(m^{1-\lambda})}$ be the k^q-NCP instance from Theorem 39. To refute randomized ETH, it is sufficient to solve c-approximate k^q-NCP with respect to (H, t) in randomized time $2^{o(m)}$. Assume that δ is small enough so that $(1 - \lambda)(1 + \delta) < 1$. We claim that by Theorem 32 with $\ell = 2$, the learner $\mathcal{A}$ solves the k^q-NCP instance in the desired amount of time.

Given a decision tree target of size 2^{2k^q} over $2N$ variables, $\mathcal{A}$ runs in time $(2N)^{(\log 2N)^\delta} = 2^{O(m^{(1-\lambda)(1+\delta)})} = 2^{o(m)}$ by our assumption on δ. We use here the fact that the size of the target satisfies $2^{2k^q} = 2^{2m^{\lambda q}} \leq 2N$ by our choice of λ. Moreover, $\mathcal{A}$ outputs a decision tree hypothesis of size $(2N)^{c'} \leq 2^{O(ck^q)}$ with error $\frac{1}{2} - \frac{1}{N^{c'}} \leq \frac{1}{2} - 2^{-\Omega(ck^q)}$ for sufficiently large c. By Theorem 32, this shows that $\mathcal{A}$ solves c-approximate k-NCP with high probability in $2^{o(m)}$ time as desired.

Gap-ETH hardness. The following reduction is implicit in [Man20] by stringing together the reduction from 3-SAT to Label Cover ([Man20, Theorem 9]), and from Label Cover to NCP ([Man20, Corollary 5]).

Theorem 40 (Reduction from gapped 3-SAT to approximating k-NCP)**.** *For all constants $c > 1$ and $\lambda > 0$, and for every $k \in \mathbb{N}$, there is a reduction running in time* $\mathrm{poly}(k, m, 2^{m/k})$ *which maps 3-*SAT *instances φ consisting of m clauses to* NCP *instances (G, z) of size* $\mathrm{poly}(k, m, 2^{m/k})$ *such that*

- Yes *case: if φ is satisfiable then (G, z) is a* Yes *instance of c-approximate k-*NCP*;*
- No *case: if every assignment to φ satisfies at most $(1 - \lambda)m$ clauses, then (G, z) is a* No *instance of c-approximate k-*NCP*.*

Using Theorem 40, we show how to refute randomized Gap-ETH if $\mathcal{A}$ runs in time $n^{o(\log n)}$. Let φ be a 3-SAT instance on n variables and with m clauses and let $\lambda > 0$ be given. Using Theorem 40 with $k = \sqrt{m}$ and for c larger than c' (to be specified later), we obtain a c-approximate k-NCP instance $(H, t) \in \mathbb{F}_2^{M \times N} \times \mathbb{F}_2^M$ where H is the parity check matrix for a linear code and $M + N = \mathrm{poly}(k, m, 2^{O(m/k)}) = 2^{O(\sqrt{m})}$. Note in particular we can assume $2^{2k} \leq 2N$ (this will satisfy our assumption on the size of the target decision tree). By Theorem 40, to approximate the number of satisfiable clauses of φ, it is sufficient to solve c-approximate k-NCP on (H, t) in randomized time $2^{o(m)}$. We claim that by Theorem 32 with $\ell = 2$, the learner $\mathcal{A}$ solves the k-NCP instance in the desired amount of time.

Given a decision tree target of size 2^{2k} over $2N$ variables, $\mathcal{A}$ runs in time $(2N)^{o(\log 2N)} = (2^{O(\sqrt{m})})^{o(\sqrt{m})} = 2^{o(m)}$. Moreover, $\mathcal{A}$ outputs a decision tree hypothesis of size $(2N)^{c'} \leq 2^{O(ck)}$ with error $\frac{1}{2} - \frac{1}{N^{c'}} \leq \frac{1}{2} - 2^{-\Omega(ck)}$ for sufficiently large c. By Theorem 32, this shows that $\mathcal{A}$ solves c-approximate k-NCP with high probability in $2^{o(m)}$ time as desired. □

Bibliography

[ABF+09] Misha Alekhnovich, Mark Braverman, Vitaly Feldman, Adam Klivans, and Toniann Pitassi. The complexity of properly learning simple concept classes. *Journal of Computer & System Sciences*, 74(1):16–34, 2009. Preliminary version in FOCS 2004. 1.2, 2.1, 2.2.1, 2.2.1, 2.2.1, 2.1, 2.3, 2.5, 2.6.3, 3.1.1, 3.2.1, 3.8.1, 3.11, 4.2.2, 4.2.3

[ABSS97] Sanjeev Arora, László Babai, Jacques Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *Journal of Computer and System Sciences*, 54(2):317–331, 1997. 4.1.1

[AH12] Micah Adler and Brent Heeringa. Approximating optimal binary decision trees. *Algorithmica*, 62(3-4):1112–1121, 2012. 2.1

[AKKV11] Mikhail Alekhnovich, Subhash Khot, Guy Kindler, and Nisheeth K Vishnoi. Hardness of approximating the closest vector problem with pre-processing. *Comput. Complex.*, 20(4):741–753, 2011. 4.1.1

[Ale11] Michael Alekhnovich. More on average case vs approximation complexity. *computational complexity*, 20:755–786, 2011. 4.1.1

[ALM+05] 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, 1998-05. 3.3.1

[ALW14] Amir Abboud, Kevin Lewi, and Ryan Williams. Losing weight by gaining edges. In *Proceedings of the 22th Annual European Symposium on Algorithms*, pages 1–12, 2014. 4.1.1

[Ang] Dana Angluin. Remarks on the difficulty of finding a minimal disjunctive normal form for boolean functions. Unpublished Manuscript. 1.2, 3.1, 3.1.1, 3.2.1, 3.8.1

[Ang88] Dana Angluin. Queries and concept learning. *Machine learning*, 2:319–342, 1988. 3.1.1

[APY09] Noga Alon, Rina Panigrahy, and Sergey Yekhanin. Deterministic approximation algorithms for the nearest codeword problem. In *Proceedings of the 12th International Workshop and 13th International Workshop on Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, APPROX '09 / RANDOM '09, pages 339–351. Springer-Verlag, 2009. 4.1, 4.1.1, 4.1, 4.3

[AS01] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, 1998-01. 3.3.1

[BA24] Enric Boix-Adsera. Towards a theory of model distillation. *arXiv preprint arXiv:2403.09053*, 2024. 4.1.1

[Bar03] J. Barnes. *Porphyry Introduction*. Clarendon later ancient philosophers. Oxford University Press, 2003. 1.1

[BB19] 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. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019. 2.6.1, 3.9

[BBE+21] Arnab Bhattacharyya, Édouard Bonnet, László Egri, Suprovat Ghoshal, Karthik C. S., Bingkai Lin, Pasin Manurangsi, and Dániel Marx. Parameterized intractability of even set and shortest vector problem. *J. ACM*, 68(3):16:1–16:40, 2021. 4.5.2

[BCGR23] Huck Bennett, Mahdi Cheraghchi, Venkatesan Guruswami, and João Ribeiro. Parameterized inapproximability of the minimum distance problem over all fields and the shortest vector problem in all $_p$ norms. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing (STOC)*, pages 553–566, 2023. 4.1.1

[BDK18] Shalev Ben-David and Robin Kothari. Randomized query complexity of sabotaged and composed functions. *Theory of Computing*, 14(5):1–27, 2018. 3.9

[BEHW89] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM (JACM)*, 36(4):929–965, 1989. 3.1.1

[BELM16] Édouard Bonnet, László Egri, Bingkai Lin, and Dániel Marx. Fixed-parameter approximability of boolean MinCSPs. *arXiv preprint arXiv:1601.04935*, 2016. 4.1.1, 4.2, 4.1, 4.3, 2, 4.5.2, 35

[BFJ+94] 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. 2.1, 3.1.1, 3.8.2

[BFKL93] Avrim Blum, Merrick Furst, Michael Kearns, and Richard J Lipton. Cryptographic primitives based on hard learning problems. In *Annual International Cryptology Conference (CRYPTO)*, pages 278–291, 1993. 4.3

[BFSO84] Leo Breiman, Jerome Friedman, Charles Stone, and Richard Olshen. *Classification and regression trees*. Wadsworth International Group, 1984. 1.1

[BGKM18] Arnab Bhattacharyya, Suprovat Ghoshal, Karthik C. S., and Pasin Manurangsi. Parameterized intractability of even set and shortest vector problem from Gap-ETH. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 107 of *LIPIcs*, pages 17:1–17:15, 2018. 4.1.1

[BHK25] Lorenzo Beretta, Nathaniel Harms, and Caleb Koch. Feature selection and junta testing are statistically equivalent. *arXiv preprint arXiv:2505.04604*, 2025. 1.3

[BHKT24] Guy Blanc, Alexandre Hayderi, Caleb Koch, and Li-Yang Tan. The Sample Complexity of Smooth Boosting and the Tightness of the Hardcore Theorem . In *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1431–1450. IEEE Computer Society, 2024. 1.3

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, 2006. 1.1, 1

[BIWX11] Arnab Bhattacharyya, Piotr Indyk, David P Woodruff, and Ning Xie. The complexity of linear dependence problems in vector spaces. In *ICS*, pages 496–508, 2011. 4.1.1

[BK02] Piotr Berman and Marek Karpinski. Approximating minimum unsatisfiability of linear equations. In *Proceedings of the 13th annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 514–516, 2002. 4.1, 4.1.1, 4.1, 4.3

[BKB17] Osbert Bastani, Carolyn Kim, and Hamsa Bastani. Interpretability via model extraction. In *Proceedings of the 4th Workshop on Fairness, Accountability, and Transparency in Machine Learning (FAT/ML)*, 2017. 2.1, 3.1

[BKL+23] Guy Blanc, Caleb Koch, Jane Lange, Carmen Strassle, and Li-Yang Tan. Certification with an NP Oracle. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference (ITCS 2023)*, volume 251 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. 1.3

[BKLS20] Joshua Brody, Jae Tak Kim, Peem Lerdputtipongporn, and Hariharan Srinivasulu. A strong XOR lemma for randomized query complexity. *arXiv preprint arXiv:2007.05580*, 2020. 2.6.1, 3.9

[BKLT22a] Guy Blanc, Caleb Koch, Jane Lange, and Li-Yang Tan. The query complexity of certification. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, pages 623–636. Association for Computing Machinery, 2022. 1.3

[BKLT22b] Guy Blanc, Caleb Koch, Jane Lange, and Li-Yang Tan. A query-optimal algorithm for finding counterfactuals. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 2075–2090. PMLR, 2022. 1.3

[BKST23] Guy Blanc, Caleb Koch, Carmen Strassle, and Li-Yang Tan. A strong composition theorem for junta complexity and the boosting of property testers . In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1757–1777. IEEE Computer Society, 2023. 1.3

[BKST24] Guy Blanc, Caleb Koch, Carmen Strassle, and Li-Yang Tan. A Strong Direct Sum Theorem for Distributional Query Complexity. In Rahul Santhanam, editor, *39th Computational Complexity Conference (CCC 2024)*, volume 300 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:30. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. 1.3

[BKST25] Guy Blanc, Caleb Koch, Carmen Strassle, and Li-Yang Tan. Computational-Statistical Tradeoffs from NP-hardness. In *2025 IEEE 66th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society, 2025. 1.3

[BKW03] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM (JACM)*, 50(4):506–519, 2003. 4.3

[BL97] Avrim Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997. 4.1.1

[Bla16] Eric Blais. Testing juntas and related properties of boolean functions. In *Encyclopedia of Algorithms*, pages 2222–2226. 2016. 4.2.3

[BLQT22] Guy Blanc, Jane Lange, Mingda Qiao, and Li-Yang Tan. Properly learning decision trees in almost polynomial time. *Journal of the ACM (JACM)*, 69(6):39:1–39:19, 2022. 2.1, 3.1.1, 3.18, 4.1.1, 4.3

[BLT20] Guy Blanc, Jane Lange, and Li-Yang Tan. Top-down induction of decision trees: rigorous guarantees and inherent limitations. In *Proceedings of the 11th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 151, pages 1–44, 2020. 2.1, 4.1.1

[Blu92] Avrim Blum. Rank-r decision trees are a subclass of r-decision lists. *Inform. Process. Lett.*, 42(4):183–185, 1992. 2.1, 3.8.2

[Blu94] Avrim Blum. Relevant examples and relevant features: Thoughts from computational learning theory. In *AAAI Fall Symposium on Relevance*, volume 5, page 1, 1994. 4.1.1

[BM02] Nader H Bshouty and Yishay Mansour. Simple learning algorithms for decision trees and multivariate polynomials. *SIAM Journal on Computing*, 31(6):1909–1925, 2002. 2.1

[BMvT78] Elwyn Berlekamp, Robert McEliece, and Henk van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3):384–386, 1978. 4.1.1

[BN90] Jehoshua Bruck and Moni Naor. The hardness of decoding linear codes with preprocessing. *IEEE Transactions on Information Theory*, 36(2):381–385, 1990. 4.1.1

[Bre96] Leo Breiman. Heuristics of instability and stabilization in model selection. *The Annals of Statistics*, 24(6):2350–2383, 1996. 1.1

[Bre01] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. 1.1

[BS96] Leo Breiman and Nong Shang. Born again trees, 1996. 2.1, 3.1

[Bsh93] Nader Bshouty. Exact learning via the monotone theory. In *Proceedings of 34th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 302–311, 1993. 1.2, 2.1, 3.1, 3.1.1, 3.7, 3.8.2, 4.1.1, 4.3

[Bsh23] Nader H. Bshouty. Superpolynomial Lower Bounds for Learning Monotone Classes. In Nicole Megow and Adam Smith, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023)*, volume 275 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 34:1–34:20, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. 2.4, 3.1.1, 3.2.1

[BSS08] Eli Ben-Sasson and Madhu Sudan. Short PCPs with polylog query complexity. *SIAM Journal on Computing*, 38(2):551–607, 2008. 17

[CFK+15a] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. 4

[CFK+15b] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. 5

[CG16] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 785–794, 2016. 4.3

[CHK20] Marek Cygan, Magnús M Halldórsson, and Guy Kortsarz. Tight bounds on subexponential time approximation of set cover and related problems. In *International Workshop on Approximation and Online Algorithms*, pages 159–173. Springer, 2020. 2.2, 2.2.2

[CHKX06] Jianer Chen, Xiuzhen Huang, Iyad A Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *Journal of Computer and System Sciences*, 72(8):1346–1367, 2006. 2.2, 2.2.2, 2.3, 9

[CIKP08] Chris Calabro, Russell Impagliazzo, Valentine Kabanets, and Ramamohan Paturi. The complexity of unique k-SAT: An isolation lemma for k-CNFs. *Journal of Computer and System Sciences*, 74(3):386–393, 2008. Computational Complexity 2003. 2

[CKW09] Marek Cygan, Łukasz Kowalik, and Mateusz Wykurz. Exponential-time approximation of weighted set cover. *Information Processing Letters*, 109(16):957–961, 2009. 2

[CL19] Yijia Chen and Bingkai Lin. The constant inapproximability of the parameterized dominating set problem. *SIAM J. Comput.*, 48(2):513–533, 2019. 2.2, 2.2.2

[CM19] 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. 2.1, 3.8.2

[CPR+07] Venkatesan T Chakaravarthy, Vinayaka Pandit, Sambuddha Roy, Pranjal Awasthi, and Mukesh Mohania. Decision trees for entity identification: Approximation algorithms and hardness results. In *Proceedings of the 26th ACM Symposium on Principles of Database Systems (PODS)*, pages 53–62, 2007. 2.1

[CS95] Mark Craven and Jude Shavlik. Extracting tree-structured representations of trained networks. *Proceedings of the 8th Conference on Advances in Neural Information Processing Systems (NeurIPS)*, 8:24–30, 1995. 2.1, 3.1

[DF13] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Springer Publishing Company, Incorporated, 2013. 4, 5

[DFVW99] Rod G Downey, Michael R Fellows, Alexander Vardy, and Geoff Whittle. The parametrized complexity of some fundamental problems in coding theory. *SIAM Journal on Computing*, 29(2):545–570, 1999. 4.1, 4.1.1

[DHM+08] Holger Dell, Thore Husfeldt, Dániel Marx, Nina Taslaman, and Martin Wahlén. Exponential time complexity of the permanent and the tutte polynomial. *ACM Trans. Algorithms*, 10(4), 2014-08. 2

[Din16] Irit Dinur. Mildly exponential reduction from gap-3sat to polynomial-gap label-cover. In *Electronic colloquium on computational complexity ECCC; research reports, surveys and books in computational complexity*, 2016. 7

[Din06] Irit Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3):12–es, 2007-06. 17

[DKRS03] Irit Dinur, Guy Kindler, Ran Raz, and Shmuel Safra. Approximating CVP to within almost-polynomial factors is NP-hard. *Comb.*, 23(2):205–243, 2003. 4.1.1, 4.3

[DKS98a] I. Dinur, G. Kindler, and S. Safra. Approximating-cvp to within almost-polynomial factors is np-hard. In *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No.98CB36280)*, pages 99–109, 1998. 4.6.6

[DKS98b] Irit Dinur, Guy Kindler, and Shmuel Safra. Approximating CVP to within almost-polynomial factors is NP-hard. In *Proceedings 39th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 99–109, 1998. 4.1.1, 4.3

[DMS03] Ilya Dumer, Daniele Micciancio, and Madhu Sudan. Hardness of approximating the minimum distance of a linear code. *IEEE Transactions on Information Theory*, 49(1):22–37, 2003. 4.1.1

[Dru12] Andrew Drucker. Improved direct product theorems for randomized query complexity. *computational complexity*, 21(2):197–244, 2012. 3.9

[DS05] Irit Dinur and Samuel Safra. On the hardness of approximating minimum vertex cover. *Annals of mathematics*, pages 439–485, 2005. 3.9.1

[DS14] Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 624–633. Association for Computing Machinery, 2014. 2.2, 2.2.2, 7, 4.2.3

[EH89] Andrzej Ehrenfeucht and David Haussler. Learning decision trees from random examples. *Information and Computation*, 82(3):231–246, 1989. 1.2, 2.1, 2.1, 3.1.1, 3.8.1, 3.11, 3.18, 4.1, 4.1.1, 4.2, 4.1, 4.2.1, 4.3

[Fei98] Uriel Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45(4):634–652, 1998. 2.2, 2.2.2, 4.2.3

[Fel06] 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. 3.1.2

[Fel16] Vitaly Feldman. Hardness of proper learning. In *Encyclopedia of Algorithms*, pages 897–900. Springer New York, 2016. 1.2, 3.1, 3.7

[FH17] Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, 2017. 2.1, 3.1

[FKLM20] Andreas Emil Feldmann, Karthik C. S., Euiwoong Lee, and Pasin Manurangsi. A survey on approximation in parameterized complexity: Hardness and algorithms. *Algorithms*, 13(6):146, 2020. 4.3

[FM04] Uriel Feige and Daniele Micciancio. The inapproximability of lattice and coding problems with preprocessing. *Journal of Computer and System Sciences*, 69(1):45–67, 2004. 4.1.1

[GGR98] Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45:653–750, 1998. 1.4.4, 4.2.3

[GJ79] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979. 2.1

[GKK08] 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. 2.1, 3.8.2, 4.1.1

[GKMP20] Mika Göös, Sajin Koroth, Ian Mertz, and Toniann Pitassi. Automating cutting planes is NP-hard. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 68–77, 2020. 3

[GLR99] David Guijarro, Vıctor Lavın, and Vijay Raghavan. Exact learning when irrelevant variables abound. *Information Processing Letters*, 70(5):233–239, 1999. 1.2, 2.1, 2.2.2, 3.1, 3.1.1, 3.7

[GLR+24] Venkatesan Guruswami, Bingkai Lin, Xuandi Ren, Yican Sun, and Kewen Wu. Parameterized inapproximability hypothesis under eth. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing (STOC)*, 2024. 4.1.1, 4.3

[GOV22] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in Neural Information Processing Systems (NeurIPS)*, 35:507–520, 2022. 4.3

[GV05] Venkatesan Guruswami and Alexander Vardy. Maximum-likelihood decoding of reed-solomon codes is np-hard. *IEEE Transactions on Information Theory*, 51(7):2249–2256, 2005. 4.1.1

[Han93] 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. 2.1, 3.8.2

[Hau88] David Haussler. Quantifying inductive bias: AI learning algorithms and valiant's learning framework. *Artificial Intelligence*, 36(2):177–221, 1988. 2.3, 3.1.1, 3.2.1

[HDW94] G. Holmes, A. Donkin, and I.H. Witten. Weka: a machine learning workbench. In *Proceedings of ANZIIS '94 - Australian New Zealnd Intelligent Information Systems Conference*, pages 357–361, 1994. 1.1

[Hir22] Shuichi Hirahara. Np-hardness of learning programs and partial mcsp. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 968–979, 2022. 2

[HJLT96] 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. 2.1, 2.2.1, 2.2.1, 2.3, 3.1.1, 3.2.1, 3.4.3, 3.8.1, 3.11, 3.8.2, 4.2.2, 4.2.3

[HKY18] Elad Hazan, Adam Klivans, and Yang Yuan. Hyperparameter optimization: A spectral approach. *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018. 2.1, 3.8.2

[HR76] Laurent Hyafil and Ronald L Rivest. Constructing optimal binary decision trees is NP-complete. *Information processing letters*, 5(1):15–17, 1976. 2.1, 2.2

[Hå07] Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001-07. 3.9.1

[IP01] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. 1, 3, 6

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. 1, 3, 6

[IRW94] 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. 3.9

[JKS10] 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. 3.9

[JS05] Jeffrey C Jackson and Rocco A Servedio. Learning random log-depth decision trees under uniform distribution. *SIAM Journal on Computing*, 34(5):1107–1128, 2005. 2.1, 4.1.1

[JS06] Jeffrey C. Jackson and Rocco A. Servedio. On learning random dnf formulas under the uniform distribution. *Theory of Computing*, 2(8):147–172, 2006. 3.8.2

[KI21] CS Karthik and Livni-Navon Inbal. On hardness of approximation of parameterized set cover and label cover: Threshold graphs from error correcting codes. In *Symposium on Simplicity in Algorithms (SOSA)*, pages 210–223. SIAM, 2021. 2.2, 2.2.2

[KLM18] CS Karthik, Bundit Laekhanukit, and Pasin Manurangsi. On the parameterized complexity of approximating dominating set. In *50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1283–1296. ACM, 2018. 2.2, 2.2.2

[KM96] Michael Kearns and Yishay Mansour. On the boosting ability of top-down decision tree learning algorithms. In *Proceedings of the 28th Annual Symposium on the Theory of Computing (STOC)*, pages 459–468, 1996. 2.1

[KM99] Michael Kearns and Yishay Mansour. On the boosting ability of top-down decision tree learning algorithms. *Journal of Computer and System Sciences*, 58(1):109–128, 1999. 1.1

[KM12] Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the Fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, 1993-12. 2.1, 3.1.1, 3.8.2, 4.1.1

[KPB99] S Rao Kosaraju, Teresa M Przytycka, and Ryan Borgstrom. On an optimal split tree problem. In *Workshop on Algorithms and Data Structures*, pages 157–168. Springer, 1999. 2.1

[KPV14] Subhash A Khot, Preyas Popat, and Nisheeth K Vishnoi. Almost polynomial factor hardness for closest vector problem with preprocessing. *SIAM Journal on Computing*, 43(3):1184–1205, 2014. 4.1.1

[KS06] Adam Klivans and Rocco Servedio. Toward attribute efficient learning of decision lists and parities. *Journal of Machine Learning Research*, 7(Apr):587–602, 2006. 2.1, 3.8.2

[KST09] 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. 2.1, 3.8.2, 4.1.1

[KST23a] Caleb Koch, Carmen Strassle, and Li-Yang Tan. Properly learning decision trees with queries is NP-hard . In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 2383–2407. IEEE Computer Society, 2023. 1.2, 3.11, 3.8.1, 4.2.2

[KST23b] Caleb Koch, Carmen Strassle, and Li-Yang Tan. Superpolynomial lower bounds for decision tree learning and testing. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1962–1994. SIAM, 2023. 1.2, 3.1.1, 3.2.1, 3.8.1, 3.11, 4.2.2

[KST24a] Caleb Koch, Carmen Strassle, and Li-Yang Tan. Fast Decision Tree Learning Solves Hard Coding-Theoretic Problems . In *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1893–1910. IEEE Computer Society, 2024. 1.2

[KST24b] Caleb Koch, Carmen Strassle, and Li-Yang Tan. Superconstant inapproximability of decision tree learning. In Shipra Agrawal and Aaron Roth, editors, *Proceedings of Thirty Seventh Conference on Learning Theory (COLT)*, volume 247 of *Proceedings of Machine Learning Research*, pages 2979–3010. PMLR, 2024. 1.2

[KV08] Michael J. Kearns and Umesh Vazirani. *An Introduction to Computational Learning Theory*. The MIT Press, 1994-08. 2.4

[KvdW07] 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. 3.9

[Lev73] Leonid A Levin. Universal sorting problem. *Problemy Predaci Informacii*, 9:265–266, 1973. 3.1.1, 3.2.1

[Lin35] Carl Linnaeus. *Systema Naturae per Regna Tria Naturae, secundum classes, ordines, genera, species*. Lugduni Batavorum, 1735. 1.1

[Lin19] Bingkai Lin. A Simple Gap-Producing Reduction for the Parameterized Set Cover Problem. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 81:1–81:15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019. 2.2, 2.2.2, 2.3, 2.5.1, 6, A.1, A.1.1

[LLL24] Shuangle Li, Bingkai Lin, and Yuwei Liu. Improved lower bounds for approximating parameterized nearest codeword and related problems under ETH, 2024. 4.1.1, 4.2, 4.1, 4.3, 2, 4.5.2, 36, 4.8.3

[Llu95] Ramon Llull. *Arbor scientiae venerabilis et caelitus illuminati patris Raymundi Lullij Maioricensis: opus nuperrime recognitum, reuisum & correctum.* Ex officina Ioannis Pillehotte, sumpt. Ioannis Caffin, & Francisci Plaignard, 1295. 1.1

[LN04] Eduardo S Laber and Loana Tito Nogueira. On the hardness of the minimum height decision tree problem. *Discrete Applied Mathematics*, 144(1-2):209–212, 2004. 2.1

[Lob90] Antoine Lobstein. The hardness of solving subset sum with preprocessing. *IEEE Transactions on Information Theory*, 36(4):943–946, 1990. 4.1.1

[LRSW22] Bingkai Lin, Xuandi Ren, Yican Sun, and Xiuhan Wang. On lower bounds of approximating parameterized k-clique. In *Proceedings of the 49th International Colloquium on Automata, Languages, and Programming (ICALP)*, 2022. 4.1.1, 4.8.3

[LY94] Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM (JACM)*, 41(5):960–981, 1994. 2.2, 2.2.2, 4.2.3

[Man20] Pasin Manurangsi. Tight running time lower bounds for strong inapproximability of maximum k-coverage, unique set cover and related problems (via t-wise agreement testing theorem). In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 62–81. SIAM, 2020. 4.1.1, 4.2, 4.1, 4.3, 2, 4.5.2, 37, 4.8.3

[MBY+16] Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, DB Tsai, Manish Amde, Sean Owen, Doris Xin, Reynold Xin, Michael J. Franklin, Reza Zadeh, Matei Zaharia, and Ameet Talwalkar. Mllib: machine learning in apache spark. *J. Mach. Learn. Res.*, 17(1):1235–1241, January 2016. 1.1

[Mic] Daniele Micciancio. Project: Complexity of coding problems. https://cseweb.ucsd.edu/~daniele/Research/CodeComp.html. 4.1.1

[Mit97] T.M. Mitchell. *Machine Learning.* McGraw-Hill International Editions. McGraw-Hill, 1997. 1.1, 1

[MOS04] Elchanan Mossel, Ryan O'Donnell, and Rocco A. Servedio. Learning functions of k relevant variables. *Journal of Computer and System Sciences*, 69(3):421–434, 2004. 1.4.4, 2.2.1, 4.1.1

[Mos15] Dana Moshkovitz. The projection games conjecture and the NP-hardness of $\ln n$-approximating set-cover. *Theory of Computing*, 11(7):221–235, 2015. 2.2, 2.2.2, 7, 4.2.3

[MPW19] Ian Mertz, Toniann Pitassi, and Yuanhao Wei. Short proofs are hard to find. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019. 3

[MR02] Dinesh Mehta and Vijay Raghavan. Decision tree approximations of boolean functions. *Theoretical Computer Science*, 270(1-2):609–623, 2002. 1.2, 2.1, 2.2.2, 3.1, 3.1.1, 3.7

[MR17] Pasin Manurangsi and Prasad Raghavendra. A Birthday Repetition Theorem and Complexity of Approximating Dense CSPs. In *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 80, pages 78:1–78:15, 2017. 7

[NRS94] 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. 3.9

[O'D14] Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014. 4.7

[OS07] Ryan O'Donnell and Rocco Servedio. Learning monotone decision trees in polynomial time. *SIAM Journal on Computing*, 37(3):827–844, 2007. 2.1, 3.8.2, 4.1.1

[PV88] Leonard Pitt and Leslie G Valiant. Computational limitations on learning from examples. *Journal of the ACM (JACM)*, 35(4):965–984, 1988. 1.2, 3.1, 3.1.1, 3.8.1, 3.11, 4.2.2, 4.2.2, 4.2.3

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 1.1

[PW10] Mihai Pătraşcu and Ryan Williams. On the possibility of faster sat algorithms. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1065–1075. SIAM, 2010. 2.3, 10

[PY91] Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–440, 1991. 3.3.1, 3.3.1, 3.9.1

[Qui86] Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986. 1.1

[Rav13] Netanel Raviv. Truth table minimization of computational models. *CoRR*, abs/1306.3766, 2013. 2.1

[RCC+22] 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. 4.1.1

[Reg03] Oded Regev. Improved inapproximability of lattice and coding problems with preprocessing. In *Proceedings of the 18th IEEE Annual Conference on Computational Complexity (CCC)*, pages 363–370, 2003. 4.1.1

[Riv87] Ronald Rivest. Learning decision lists. *Machine learning*, 2(3):229–246, 1987. 2.1, 3.8.2

[RRV07] Dana Ron, Amir Rosenfeld, and Salil Vadhan. The hardness of the expected decision depth problem. *Information processing letters*, 101(3):112–118, 2007. 2.1

[RS97] Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC)*, pages 475–484. Association for Computing Machinery, 1997. 4.2.3

[Rud19] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019. 2.9

[Sav02] Petr Savickỳ. On determinism versus unambiquous nondeterminism for decision trees. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 9, 2002. 3.9, 3.14, 27

[SDV19] Noah Stephens-Davidowitz and Vinod Vaikuntanathan. SETH-hardness of coding problems. In *Proceedings of the 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 287–301, 2019. 4.1.1

[Sha04] Ronen Shaltiel. Towards proving strong direct product theorems. *Computational Complexity*, 12(1/2):1–22, 2004. 3.9

[Sie08] Detlef Sieling. Minimization of decision trees is hard to approximate. *Journal of Computer and System Sciences*, 74(3):394–403, 2008. 1.2, 2.1, 3.1.3, 3.2.2, 3.4.3, 3.4.3, 19, 3.4.3, 3.7, 3.17, 3.17, 30

[SS93] Robert Schapire and Linda Sellie. Learning sparse multivariate polynomials over a field with queries and counterexamples. In *Proceedings of the 6th Annual Conference on Computational Learning Theory (COLT)*, pages 17–26, 1993. 3.1.1, 4.1.1

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014. 1.1, 1

[SZA22] Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022. 4.3

[Tov84a] Craig A. Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, 8(1):85–89, 1984. 1

[Tov84b] Craig A. Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, 8(1):85–89, 1984. 6

[Tre14] Luca Trevisan. *Inapproximability of Combinatorial Optimization Problems*, chapter 13, pages 381–434. John Wiley & Sons, Ltd, 2014. 3.3.1

[VAB07] Anneleen Van Assche and Hendrik Blockeel. Seeing the forest through the trees: Learning a comprehensible model from an ensemble. In *European Conference on Machine Learning (ECML)*, pages 418–429, 2007. 2.1, 3.1

[Val84] Leslie Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984. 1.1, 1.2, 1.2, 1.4.4, 2.2.1, 3.1, 3.1.1, 3.1.2, 4.1.1

[Val85] Leslie G Valiant. Learning disjunction of conjunctions. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 560–566, 1985. 3.1.2

[Val05] Gregory Valiant. Finding correlations in subquadratic time, with applications to learning parities and the closest pair problem. *J. ACM*, 62(2), 2015-05. 2.2.1, 4.1.1

[VLJ+17] Gilles Vandewiele, Kiani Lannoye, Olivier Janssens, Femke Ongenae, Filip De Turck, and Sofie Van Hoecke. A genetic algorithm for interpretable model extraction from decision tree ensembles. In *Trends and Applications in Knowledge Discovery and Data Mining*, pages 104–115, 2017. 2.1, 3.1

[VS20] Thibaut Vidal and Maximilian Schiffer. Born-again tree ensembles. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 9743–9753, 2020. 2.1, 3.1

[WKRQ+08] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, Philip S Yu, et al. Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37, 2008. 1.1

[ZB00] Hans Zantema and Hans Bodlaender. Finding small equivalent decision trees is hard. *International Journal of Foundations of Computer Science*, 11(2):343–354, 2000. 1.2, 2.1, 3.1.3, 3.2.2, 3.4.3, 3.7, 3.17

[ZH16] Yichen Zhou and Giles Hooker. Interpreting models via single tree approximation, 2016. 2.1, 3.1

Appendix A

Appendix

A.1 Hardness of Approximating Set Cover

We first state a lemma due to [Lin19], translated into our notation.

Lemma A.1.1 (Lin's lemma [Lin19, Lemma 3.6])**.** *There is an algorithm which given $k \in \mathbb{N}$, $\delta > 0$ with $(1+1/k^3)^{1/k} \leq (1+\delta)/(1+\delta/2)$ and $(1+\delta/2)^k \geq 2k^4$ and a SAT instance ϕ with n variables and Cn clauses, where n is much larger than k and C, outputs an integer $N \leq 2^{n/k} + n/k^3$ and a set cover instance $\mathcal{S} = (S, U, E)$ satisfying*

- $|S| + |U| \leq N$;
- *if ϕ is satisfiable, then* $\operatorname{opt}(\mathcal{S}) \leq k$;
- *if ϕ is unsatisfiable, then* $\operatorname{opt}(\mathcal{S}) > \frac{1}{1+\delta}\left(\frac{\log N}{\log\log N}\right)^{1/k}$

The exact version of this lemma we use is the following.

Lemma A.1.2 (Reducing SAT to Set-Cover)**.** *There is an algorithm that takes an n-variate SAT instance φ of size $|\varphi|$ and an integer $k \geq 100$ with $k^2 \leq n/\log n$ and produces a set cover instance $\mathcal{S}$ of size $N \leq 2^{2|\varphi|/k}$ in time $\leq 2^{5|\varphi|/k}$ such that*

1. *if φ is satisfiable then* $\operatorname{opt}(S) \leq k$;
2. *if φ is unsatisfiable then* $\operatorname{opt}(S) > \frac{1}{2}\left(\frac{\lg N}{\lg\lg N}\right)^{1/k}$.

Proof. We use Lemma A.1.1 with $\delta = 1/2$. For this value of δ, if $k \geq 100$, then both conditions $(1+1/k^3)^{1/k} \leq (1+\delta)/(1+\delta/2)$ and $(1+\delta/2)^k \geq 2k^4$ of Lemma A.1.1 are satisfied. Moreover, an inspection of the proof of Lemma A.1.1 shows that the condition "n is much larger than k" in the lemma statement means $k^2 \leq n/\log n$.

Therefore, Lemma A.1.1 returns a set cover instance $\mathcal{S}$ satisfying

1. if φ is satisfiable then $\text{opt}(S) \leq k$;
2. if φ is unsatisfiable then $\text{opt}(S) > \frac{1}{1+\delta}\left(\frac{\lg N}{\lg\lg N}\right)^{1/k}$.

By our choice of δ, $1/(1+\delta) = 2/3 > 1/2$ as desired. Since $n \leq |\varphi|$, the size of the set cover instance is $N \leq 2^{|\varphi|/k+|\varphi|/k^3} \leq 2^{2|\varphi|/k}$. The runtime of the reduction is $\leq 2^{5|\varphi|/k}$. □

We can now prove the main theorem from Section 2.5.1.

Proof of Theorem 6. Suppose there exists an algorithm that can solve $\left(k, \frac{1}{2}\left(\frac{\log N}{\log\log N}\right)^{1/k}\right)$-Set-Cover on N vertices with high probability in time N^{ck}. Then we show how to solve SAT with high probability for SAT formulas with n variables in time 2^{3cn}.

Let φ be a SAT instance with n variables. Choose $k \in \mathbb{N}$ so that

$$k = \frac{1}{2} \cdot \frac{\log\log(2^{2n/k})}{\log\log\log(2^{2n/k})} = \frac{1}{2} \cdot \frac{\log(2n/k)}{\log\log(2n/k)}.$$

Given n this equation can be numerically solved efficiently, and k will be some value between $\log\log n$ and $\log n$. We then apply Lemma A.1.2 with this value of k to obtain a set cover instance $\mathcal{S}$ of size $N \leq 2^{2n/k}$ in time $\leq 2^{5n/k}$. If $N < 2^{2n/k}$ then we add dummy items/dummy sets to the universe so that $N = 2^{2n/k}$. Note this padding will not affect the optimal set cover for the optimal set cover size. Hence, by construction, we have an instance of $\left(k, \frac{1}{2}\left(\frac{\log N}{\log\log N}\right)^{1/k}\right)$-Set-Cover of size N where

$$k = \frac{1}{2} \cdot \frac{\log\log(N)}{\log\log\log(N)}.$$

We can therefore run our algorithm for set cover on this instance $\mathcal{S}$ and output "Yes" if the algorithm outputs Yes and "No" if the algorithm outputs No.

Runtime. Our reduction runs in time

$$\begin{aligned} 2^{5n/k} + N^{ck} &\leq 2^{5n/k} + (2^{2n/k})^{ck} \\ &= 2^{5n/k} + 2^{2cn} \\ &\leq 2^{3cn}. \end{aligned}$$

Correctness. By assumption, the set cover algorithm solves $\left(k, \frac{1}{2}\left(\frac{\log N}{\log\log N}\right)^{1/k}\right)$-Set-Cover with high probability and therefore by Lemma A.1.2 our algorithm solves SAT with high probability. We also note that $2k^k < \frac{\log N}{\log\log N}$ and so $k < \frac{1}{2}\left(\frac{\log N}{\log\log N}\right)^{1/k}$ for our choice of k. If instead, one were to choose e.g. $k = \log\log n$, then $2k^k > \frac{\log N}{\log\log N}$ and so the set cover instance would fail to determine the satisfiability of φ.

It follows that if SAT cannot be solved in randomized time $O(2^{\delta n})$ for some $\delta \in (0,1)$ then $\left(k, \frac{1}{2}\left(\frac{\log N}{\log\log N}\right)^{1/k}\right)$-Set-Cover cannot be solved in randomized time $N^{\delta/3\cdot k}$. □

A.2 Proof of Proposition 2.6.4

We first compute:

$$\begin{aligned}
\Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[\boldsymbol{y}=y] &= \Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[\mathrm{BlockwisePar}(\boldsymbol{y})=\mathrm{BlockwisePar}(y)] \\
&\quad\cdot \Pr_{\boldsymbol{y}\sim\mathcal{D}_{\oplus\ell}}[\boldsymbol{y}=y \mid \mathrm{BlockwisePar}(\boldsymbol{y})=\mathrm{BlockwisePar}(y)] \qquad \text{(Law of total probability)} \\
&= \Pr_{\boldsymbol{x}\sim\mathcal{D}}[\boldsymbol{x}=\mathrm{BlockwisePar}(y)] \cdot \Pr_{\boldsymbol{y}\sim\mathcal{U}_{n\cdot\ell}}[\boldsymbol{y}=y \mid \mathrm{BlockwisePar}(\boldsymbol{y})=\mathrm{BlockwisePar}(y)] \\
&\qquad\qquad \text{(Definition of } \mathcal{D}_{\oplus\ell}) \\
&= \Pr_{\boldsymbol{x}\sim\mathcal{D}}[\boldsymbol{x}=\mathrm{BlockwisePar}(y)] \cdot \prod_{i\in[n]} \Pr_{\boldsymbol{y}_i\sim\mathcal{U}_\ell}[\boldsymbol{y}_i=y_i \mid \oplus\boldsymbol{y}_i=\oplus y_i] \\
&\qquad\qquad \text{(Independence of } \boldsymbol{y}_i\text{'s)} \\
&= \Pr_{\boldsymbol{x}\sim\mathcal{D}}[\boldsymbol{x}=\mathrm{BlockwisePar}(y)] \cdot \prod_{i\in[n]} 2^{-(\ell-1)}
\end{aligned}$$

where the last step follows from the fact that conditioning on the parity of $\boldsymbol{y}_i$ being a specific bit removes 1 out of ℓ degrees of freedom. With an analogous calculation for $\mathcal{D}^j_{\oplus\ell}$, we obtain

$$\begin{aligned}
\Pr_{\boldsymbol{y}\sim\mathcal{D}^j_{\oplus\ell}}[\boldsymbol{y}=y] &= \Pr_{\boldsymbol{y}\sim\mathcal{D}^j_{\oplus\ell}}[\mathrm{BlockwisePar}(\boldsymbol{y})=\mathrm{BlockwisePar}(y)] \\
&\quad\cdot \Pr_{\boldsymbol{y}\sim\mathcal{D}^j_{\oplus\ell}}[\boldsymbol{y}=y \mid \mathrm{BlockwisePar}(\boldsymbol{y})=\mathrm{BlockwisePar}(y)] \qquad \text{(Law of total probability)} \\
&= \Pr_{\boldsymbol{x}\sim\mathcal{D}}[\boldsymbol{x}=\mathrm{BlockwisePar}(y)] \cdot \prod_{i\in[n]} \Pr_{\boldsymbol{y}_i\sim\mathcal{U}_{\ell-1}}\left[\boldsymbol{y}_i=y_i^{-j}\right] \qquad \text{(Definition of } \mathcal{D}^j_{\oplus\ell}) \\
&= \Pr_{\boldsymbol{x}\sim\mathcal{D}}[\boldsymbol{x}=\mathrm{BlockwisePar}(y)] \cdot \prod_{i\in[n]} 2^{-(\ell-1)}
\end{aligned}$$

where $y_i^{-j} \in \{0,1\}^{\ell-1}$ is the string y_i with its jth bit removed.