Testing Properties of Boolean Functions

Eric Blais

CMU-CS-12-101
January 2012

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Ryan O'Donnell, Chair
Avrim Blum
Venkatesan Guruswami
Ronitt Rubinfeld, M.I.T.
Madhu Sudan, Microsoft Research

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.*

This research was sponsored by the National Science Foundation under grant numbers CCF-0747250, CCF-0915893, CCF-1116594, CCR-0122581; US Army Research Office under grant number DAAD-190210389; and the Fonds de recherche du Québec – Nature et technologies.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Property Testing, Boolean Functions, Juntas, Partial Symmetry, Linear Functions, Function Isomorphism, Communication Complexity, Learning Theory, Decision Tree Complexity

À mon amour, Edith.

Abstract

Given oracle access to some boolean function f, how many queries do we need to test whether f is linear? Or monotone? Or whether its output is completely determined by a small number of the input variables? This thesis studies these and related questions in the framework of property testing introduced by Rubinfeld and Sudan ('96).

The results of this thesis are grouped into three main lines of research.

I. We determine nearly optimal bounds on the number of queries required to test k-juntas (functions that depend on at most k variables) and k-linearity (functions that return the parity of exactly k of the input bits). These two problems are fundamental in the study of boolean functions and the bounds obtained for these two properties lead to tight or improved bounds on the query complexity for testing many other properties including, for example, testing sparse polynomials, testing low Fourier degree, and testing computability by small-size decision trees.

II. We give a partial characterization of the set of functions for which we can test isomorphism—that is, identity up to permutation of the labels of the variables—with a constant number of queries. This result provides some progress on the question of characterizing the set of properties of boolean functions that can be tested with a constant number of queries.

III. We establish new connections between property testing and other areas of computer science. First, we present a new reduction between testing problems and communication problems. We use this reduction to obtain many new lower bounds in property testing from known results in communication complexity. Second, we introduce a new model of property testing that closely mirrors the active learning model. We show how testing results in this new model may be used to improve the efficiency of model selection algorithms in learning theory.

The results presented in this thesis are obtained by applying tools from various mathematical areas, including probability theory, the analysis of boolean functions, orthogonal polynomials, and extremal combinatorics.

Acknowledgments

This thesis could not have been completed without the help of many people.

First and foremost, I offer my heartfelt thanks to my advisor, Ryan O'Donnell. Ryan's influence can be seen on every page of his thesis. I am deeply grateful for the guidance, support, and generosity he has shown throughout my studies.

In addition, I wish to thank the other members of my thesis committee: Avrim Blum, Venkatesan Guruswami, Ronitt Rubinfeld, and Madhu Sudan. Their advice, feedback, and insights have been invaluable in the completion of this thesis.

I thank my coauthors and collaborators: Noga Alon, James Aspnes, Maria-Florina Balcan, Paul Beame, Avrim Blum, Joshua Brody, Sourav Chakraborty, Dana Dachman-Soled, Murat Demirbas, David García Soriano, Dang-Trinh Huynh-Ngoc, Daniel Kane, Arie Matsliah, Kevin Matulef, Ryan O'Donnell, Atri Rudra, Rocco Servedio, Steve Uurtamo, Amit Weinstein, Karl Wimmer, Liu Yang, and Yuichi Yoshida. I have learned much from them and they have made the research experience truly enjoyable.

I offer special thanks to Paul Beame, Mathieu Blanchette, and Sofya Raskhodnikova, whose advice and mentorship have been most helpful in shaping my research career.

I was very fortunate to pursue my doctoral studies at CMU and, more specifically, within the Theory Group in the Computer Science Department. I wish to extend special thanks to Moritz Hardt, Krzysztof Onak, Aaron Roth, Or Sheffet, Yi Wu, and Yuan Zhou for numerous stimulating discussions throughout my studies. I also wish to thank Deborah Cavlovich, Sophie Park, Charlotte Yano, and the rest of the administrative staff for all their support throughout my studies.

Much of the research produced in the course of this thesis was fueled by the fine coffees of Tazza d'Oro, Commonplace Coffee, Kiva Han, 61C, and La Prima Espresso.

Lastly, and most importantly, I thank my family for all their support. Edith, without you this process would have been much harder. Jacob, without you it would have been much easier... but it would not have been nearly as much fun.

Bibliographic notes

Most of the research presented in this thesis has already been published in technical reports, conference abstracts, and/or journal articles. Most of this research is the result of joint work.

The algorithm for testing juntas presented in Chapter 5 was originally introduced in [20] and builds on earlier work published in [19]. The analysis of the algorithm presented in this thesis was obtained with Amit Weinstein and Yuichi Yoshida and is presented in the manuscript [26]. The results in Chapter 6 on testing partial symmetry and in Chapter 8 on testing isomorphism to partially symmetric functions are also the result of the same collaboration [26].

The lower bound on testing k-linearity presented in Chapter 7 is joint work with Daniel Kane and the result is in a manuscript that will soon appear [24].

The lower bound on the query complexity of testing isomorphism to random functions in Chapter 9 is joint work with Noga Alon and appeared in [3]. The theorem stated in that paper, however, makes a weaker claim that the lower bound applies only to non-adaptive testing algorithms. Independently, Sourav Chakraborty, David García-Soriano, and Arie Matsliah [40] obtained overlapping results. After examining [40], we realized that the argument in [3], with only very minor changes, also gives a lower bound for adaptive algorithms. The resulting argument is presented here. The two articles [3, 40] have since been combined and the result [4] is currently in journal submission.

The lower bound for testing function isomorphism in Chapter 10 is joint work with Ryan O'Donnell and was published in [25].

The connection between property testing and communication complexity, as presented in Chapter 11, is joint work with Joshua Brody and Kevin Matulef. This work was originally published in conference proceedings [22] but the presentation in this thesis more closely follows the revised journal version of the same article [23].

The material in Chapter 12 on the active property testing model is the result of joint work with Maria-Florina Balcan, Avrim Blum, and Liu Yang [13] and first appeared in the manuscript [13].

Contents

Introduction

Chapter 1

Overview

This thesis is concerned with testing properties of boolean functions. The subject of this topic is best introduced with an illustrative example.

1.1 Black boxes

Imagine that we are given some box with n switches (each of which can be either "on" or "off") and a lightbulb that lights up on some of the configurations of the switches. This box can be said to compute a boolean function $f : \{0,1\}^n \to \{0,1\}$, where we define $f(x_1, \ldots, x_n) = 1$ iff the bulb is lit when we set switch i to the "on" position iff $x_i = 1$ for $i = 1, 2, \ldots, n$.

The first thing we might want to do with such a box is to figure out what function it computes. If we don't want to open up the box, the only way we can do this is to go through all 2^n configurations of the switches and see for which configurations the light is on. Even if we are given a user's manual that purports to identify the function computed by the box, verifying that the user's manual is correct still requires checking all 2^n configurations.

Imagine now that we are again given the user's manual that describes the function computed by the box and that the user's manual also has one extra promise:

> *Due to manufacturing limitations, some boxes may not behave as described in this manual. However, the manufacturing process guarantees that faulty boxes will disagree with the table above on at least* 10% *of the possible configurations.*

With this promise, we can now test the box much more efficiently. To do so, we simply verify that the box is consistent with the definition in the user manual on 100 different switch configurations each chosen at random. When the box is correct, it will agree with the user's manual definition on all the configurations. When the box is faulty, the manufacturer's promise guarantees that for each such configuration, the box's output will disagree with the user's manual with probability $\frac{1}{10}$. So the probability that the box is faulty but passes all of the tests is $(1 - \frac{1}{10})^{100} < \frac{1}{20000}$.

This example is elementary and the result not too surprising, but it hints at a rather remarkable truth: for some decision problems, there may be "approximate" formulations of the problem that are extremely easy to solve. (In this case, the decision problem was the *function identity* testing problem, and the "approximation" aspect was the "gap promise" that any function that did not satisfy the identity condition was "far" from satisfying it.)

The generalization can again be formulated in terms of our example. Instead of identifying the function, we may receive a user's manual that only describes some of the properties of the function computed by the black box. For example, we may read the following:

> *Congratulations on buying the ACME JuntaBox! The JuntaBox, like our traditional box, contains n switches. For your convenience, however, we have built the JuntaBox in a way that the behavior of the light is entirely controlled by at most* 10 *of the switches.*

This user's manual does not identify the function computed by the box, but it makes a strong claim regarding how this function should behave. As before, if we want to verify that the claim is correct, we may have to check all 2^n switch configurations. Once again, however, the testing task may be significantly simplified by a promise:

> *Due to manufacturing limitations, some JuntaBoxes may not behave as described in this manual. However, the manufacturing process guarantees that faulty boxes will disagree with* any *valid JuntaBox on at least* 10% *of the switch configurations.*

Unlike in the function identity problem discussed earlier, it is not immediately clear whether we can make use of this promise to create a very efficient test for the "junta" property promised in the user's manual. At the very least, the naïve strategy of checking random switch configurations does not appear to be very helpful since in this case we don't have a single target function to test against. Nevertheless, as we will see in Chapter 5, the promise is indeed very helpful, and under this promise we can test the junta property very efficiently.

1.2 Property testing

The function identity and junta testing problems described in the last section are only two of many properties that one may want to test on black boxes. For example, we might want to test if the box's function is linear, if it is monotone, or if it is computed by a small boolean circuit. The testing problem (and the "approximate" decision model) for all these properties—and many more—can be described in the *property testing* framework first introduced by Rubinfeld and Sudan [90]. A thorough presentation of this framework is presented in Chapter 3. Let us also offer a quick and informal introduction to the framework here.

A *property* $\mathcal{P}$ of boolean functions $\{0,1\}^n \to \{0,1\}$ is a subset of all these functions.[1] The function f *has* property $\mathcal{P}$ if $f \in \mathcal{P}$. Conversely, we say that the function f is *ϵ-far* from $\mathcal{P}$ if $|\{x \in \{0,1\}^n : f(x) \neq g(x)\}| \geq \epsilon 2^n$ for every $g \in \mathcal{P}$.

A *q-query ϵ-tester* for $\mathcal{P}$ is a randomized algorithm $\mathcal{A}$ that, given oracle access to some function $f : \{0,1\}^n \to \{0,1\}$, queries the value of f on at most q elements from $\{0,1\}^n$ and satisfies two conditions:

1. When f has property $\mathcal{P}$, $\mathcal{A}$ accepts f with probability at least $\frac{2}{3}$.

2. When f is ϵ-far from $\mathcal{P}$, $\mathcal{A}$ rejects f with probability at least $\frac{2}{3}$.

The *query complexity* of the property $\mathcal{P}$ is the minimum value of q for which there is a *q-query ϵ-tester* for $\mathcal{P}$. A principal goal in property testing is to determine the query complexity for all natural properties of boolean functions.

1.3 Results

In its most general form, the goal of determining the query complexity for all properties of boolean functions is unattainable with the current (mathematical and algorithmic) tools at our disposal. In order to make some progress on it, therefore, we must restrict our attention to some class of properties of boolean functions. Ideally, we would like this class to include all—or almost all—the properties of boolean functions that we consider "natural".

[1] While we will generally focus on "natural" properties that can be described easily, the definition applies equally well to any arbitrary set of functions.

In this thesis, we restrict our attention to the class of properties of boolean functions that are *closed under the relabeling of the input variables*. This class includes most of the properties of boolean functions that we may consider "natural", including linearity, representability by low-degree polynomials, symmetric functions, juntas, representability by small decision trees or by small circuits, monotonicity, submodularity, halfspaces, low Fourier degree, etc.

While these properties have been extensively studied, three main questions remain largely open. First, what is the *exact* query complexity of these properties? When we began this research, there were large gaps between the best upper and lower bounds for the query complexity of most of these properties. Our results, as covered in Chapters 5–7 improve the upper and lower bounds for a number of these properties and close a number of these gaps.

The second question that remains largely open is: *why* are some properties testable efficiently? Or, in other words, what *characteristics* of properties are necessary or sufficient to guarantee, for example, that the query complexity of a property is independent of the domain size of the input functions? Over the last few years, there has been a lot of progress on this question when restricted to the class of *algebraic* properties of functions [18, 72, 95]—i.e., on properties of boolean functions that satisfy stronger invariance requirements such as invariance under linear or affine transformations. In this thesis, we take a different approach and instead study the important sub-problem of testing function isomorphism. These results are presented in Chapters 8–10.

The third and final main open question that we examine in this thesis is: what *connections* can we establish between property testing and other areas of computer science? In Chapters 11 and 12, we describe new connections to communication complexity and to learning theory.

The following subsections describe the results in this thesis in more detail.

1.3.1 Part I: Exact query complexity

We begin by examining the query complexity for some of the most fundamental properties of boolean functions. Specifically, we study *juntas* and *k-linearity*. These two classes of functions play a particularly fundamental role in property testing and, as we will discuss in the later chapters, new bounds on the query complexity for these problems improve the bounds for the query complexity of a number of other properties.

In Chapter 5, we begin by studying the problem of testing k-juntas—or, in other words, of testing whether a boolean function has at most k relevant variables. While it was known previously that it is possible to test k-juntas with a number of queries that depends only on k and ϵ [52], the exact dependence on k was not known. We present a new algorithm that settles this question up to logarithmic factors.

Besides the improved query complexity, the main contributions of the new algorithm are twofold. First, the algorithm itself is conceptually very simple, and closely parallels the optimal algorithm for *learning* juntas with membership queries [29]. Second, the analysis of the algorithm uses results on intersecting family, illustrating a new and potentially useful connection between the analysis of algorithms and extremal combinatorics.

In Chapter 6, we examine the problem of testing *partially symmetric* functions: that is, of testing whether there is a set S of at most k variables such that the input function is invariant to any relabeling of the variables outside S. (See Chapter 6 for a more detailed introduction to partially symmetric functions.) Every junta is also partially symmetric, but the converse is not true; the set of partially symmetric functions is much larger. The main result of Chapter 6 is that the junta tester from the previous chapter can be extended to test partial symmetry with roughly the same query complexity.

The motivation for the study of partially symmetric functions was to better understand how the invariance structure of juntas is responsible for the fact that the property is efficiently testable. An important contribution of this chapter, required for the analysis of the tester, is the introduction and analysis of a new measure of influence, which we call *symmetric influence*. This notion may be of independent interest.

In Chapter 7, we turn to the problem of proving lower bounds for the query complexity of properties of boolean functions. Specifically, we examine the problem of testing k-linearity—or testing whether the function returns the parity of exactly k of its variables. We show that roughly $\min\{k, n-k\}$ queries are required for this task. This confirms a conjecture of Goldreich [57] and also gives new lower bounds for a number of other properties, including juntas, functions of low Fourier degree, sparse polynomials, and functions computable by small decision trees.

The result in Chapter 7 is obtained by reducing the problem of testing k-linearity to a purely geometrical problem on the hypercube. One interesting aspect of this result is that, unlike most results in property testing, the lower bound we obtain is not just asymptotically linear in k—it is equal to $k - o(k)$.

1.3.2 Part II: Testing function isomorphism

The second part of the thesis examines property testing from a more qualitative point of view. Instead of seeking to determine the exact query complexity for some specific properties of boolean functions, the motivating question behind the research presented here is: can we *characterize* the set of properties of boolean functions that can be tested with a constant number of queries?[2]

An important sub-goal of this research direction is to characterize the set of functions for which we can test isomorphism—that is, for which we can test identity *up to relabeling of the input variables*—with a constant number of queries. This problem was first raised by Fischer et al. [52] but until recently this problem was largely open, with only three exceptions: all symmetric functions and all juntas on a constant number of variables were known to be isomorphism-testable with a constant number of queries, and it was known that testing isomorphism to parity functions on $\omega(1) \leq k \leq o(\sqrt{n})$ variables required a super-constant number of queries.

In Chapter 8, we begin by unifying and extending the results concerning the isomorphism-testability of symmetric functions and juntas. As we have seen above, the set of partially symmetric functions encompasses the set of juntas. Clearly, this set also includes the set of symmetric functions. (It also includes many other functions as well.) In this chapter, we show that we can test isomorphism to *any* partially symmetric function with a constant number of queries.

In Chapter 9, we show that the set of functions for which we cannot test isomorphism with a constant number of queries extends far beyond the set of k-linear functions for some values of k. In fact, we show that for *almost all* boolean functions, testing isomorphism to those functions requires $\Omega(n)$ queries. Since $O(n \log n)$ queries are sufficient to test isomorphism to any function, the result of this section shows that the universal upper bound is nearly tight for almost all functions.

The result in Chapter 9 is non-constructive. In Chapter 10, we give a large explicit class of functions for which testing isomorphism requires a super-constant number of queries. Specifically, we answer a question of Fischer et al. [52] by showing, roughly, that testing isomorphism to k-juntas that "strongly depend" on nearly all of the relevant variables requires at least $\log k$ queries.

1.3.3 Part III: Connections

In the third part of the thesis, we explore some connections between property testing and other areas of computer science.

Chapter 11 presents a connection between property testing and communication com-

[2] I.e., with a number of queries that does not depend on the size of the domain of the functions.

plexity. Communication complexity has been remarkably successful in proving lower bounds in many areas of computer science. The research presented in this chapter was motivated by a simple question: might communication complexity be useful for proving lower bounds in property testing as well? The results in this chapter show that the answer to this question is an emphatic “yes”. We present the basic reduction that connects the two areas in the chapter and show how it can be used to prove lower bounds for testing k-linearity, monotonicity, and computability by small decision trees.

In Chapter 12, we examine the connection between property testing and learning theory. It has long been known that the two areas are closely connected. (We review some of the details of this connection in Section 3.2.) This connection, however, is largely concerned with the connection between property testing and the *membership query* learning model—where the learner can query the target function on any input of its choosing.

In many applications of learning theory, the membership query model is not realistic. An alternative model, called *active learning*, where the learner can query any input of its choosing *among those that exist in the real-world* is often more realistic. As a result, this model is of great interest to the learning community. In Chapter 12, we introduce a new model of property testing, which we call *active property testing*, with the same relation to active learning as the standard property testing model has to membership query learning. We describe the active testing model and present results concerning the testability of dictator functions and halfspaces in this model.

Chapter 2

Boolean Functions

The main object of study in this thesis is the boolean function. The goal of this section is to introduce the basic definitions and some fundamental tools we will use throughout the rest of the thesis. We will also examine two classes of boolean functions that will appear repeatedly in later chapters: juntas and parity functions.

Readers familiar with boolean functions are encouraged to quickly skim this chapter to become acquainted with the notation. Conversely, for readers who would appreciate a more thorough introduction to boolean functions and the tools used to analyze them, we recommend the surveys [80, 45] and the book [81].

2.1 Boolean hypercube

Before examining boolean functions, let us first take a moment to establish some basic facts regarding their domains: the boolean hypercube $\{0,1\}^n$.

Given an element $x \in \{0,1\}^n$, we write $x_1, \ldots, x_n$ to refer to the individual coordinates of x. When discussing boolean functions, we refer to the set $[n] := \{1, \ldots, n\}$ as the set of *variables* of f. In this terminology, the value of the ith variable in x is x_i.

The elements $e^1, \ldots, e^n \in \{0,1\}^n$ are defined by setting the ith coordinate of e^i to 1 and all other coordinates to 0. More generally, for every set $S \subseteq [n]$, the characteristic vector for S, denoted by $e^S \in \{0,1\}^n$, is defined by setting $e^S_i = 1$ for every $i \in S$ and $e^S_i = 0$ for every $i \in [n] \setminus S$. We also use $\mathbf{0}$ and $\mathbf{1}$ to denote the all-zero and all-one vectors. (Note that $e^{\emptyset} = \mathbf{0}$ and $e^{[n]} = \mathbf{1}$.)

Given two bits $a, b \in \{0,1\}^n$, there are three basic binary operators that we use:

AND ($\wedge$), OR ($\vee$), and XOR ($\oplus$). They are all defined in the standard way.

$$a \vee b = \begin{cases} 1 & \text{if } a = 1 \text{ or } b = 1 \\ 0 & \text{otherwise.} \end{cases}$$

$$a \wedge b = \begin{cases} 1 & \text{if } a = b = 1 \\ 0 & \text{otherwise.} \end{cases}$$

$$a \oplus b = \begin{cases} 1 & \text{if } a \neq b \\ 0 & \text{otherwise.} \end{cases}$$

We define the same operators to apply bit-wise for the elements in $\{0,1\}^n$. E.g., three elements $x, y, z \in \{0,1\}^n$ satisfy $z = x \oplus y$ iff $z_i = x_i \oplus y_i$ for each $i \in [n]$.

The set $\{0,1\}^n$ with the bit-wise XOR operator $\oplus$ forms a group. This group, combined with the the trivial scalar multiplication operator defined by $1 \cdot x = x$ and $0 \cdot x = \mathbf{0}$, forms a vector space over the field $\mathbb{F}_2$. With some abuse of notation, we use $\{0,1\}^n$ to refer to the set, the group, and the vector space, depending on the context.

We can define the *inner product* of elements in $\{0,1\}^n$ by setting

$$\langle x, y \rangle = \sum_{i=1}^{n} x_i \wedge y_i$$

for every $x, y \in \{0,1\}^n$. The *norm* obtained with this inner product is

$$\|x\| = \langle x, x \rangle = \sum_{i=1}^{n} x_i,$$

which is also known as the *Hamming weight* of x.

For every $S \subseteq [n]$, we define the *projection* operator $P^S : \{0,1\}^n \to \{0,1\}^n$ by setting $P^S(x) = x \wedge e^S$. We will use the shorthand notation x_S to represent the projection $P^S(x)$. We will often use the projection operator to combine two or more boolean vectors. For example, the element $z = x_S \vee y_{\bar{S}}$ is the “hybrid” vector that is identical to x on all coordinates in S and is identical to y on all the other coordinates.

Remark 2.1. In some settings, it is more convenient to consider the boolean hypercube as the vector space over $\{-1,1\}^n$ or as $\mathbb{F}_2^n$. The three representations of the hypercube are isomorphic. In the following, we will mostly use the $\{0,1\}^n$ representation.

2.2 Boolean functions

Most of the boolean functions we consider in this thesis are of the form $f : \{0,1\}^n \to \{0,1\}$. In order to analyze these functions, it is convenient to consider the larger class of functions mapping the boolean hypercube $\{0,1\}^n$ to the set of real numbers $\mathbb{R}$.

The set of boolean functions $\{0,1\}^n \to \mathbb{R}$, with the standard notions of function addition and scalar multiplication of functions, forms a vector space over $\mathbb{R}$. We can define the *inner product* of two functions $f, g : \{0,1\}^n \to \mathbb{R}$ in this vector space by setting

$$\langle f, g \rangle = \mathop{\mathbf{E}}_{x \in \{0,1\}^n} [f(x) \cdot g(x)]$$

where the expectation is over the uniform distribution on $\{0,1\}^n$. The inner product can be used to define the *(L_2) norm* of f as

$$\|f\|_2 = \sqrt{\langle f, f \rangle}.$$

A fundamental tool in the analysis of boolean functions is the Cauchy-Schwarz inequality, which bounds the inner product of two functions by the product of their L_2 norms.

Theorem 2.2 (Cauchy-Schwarz Inequality). *For any functions $f, g : \{0,1\}^n \to \mathbb{R}$,*

$$\langle f, g \rangle \le \|f\|_2 \cdot \|g\|_2.$$

Another useful tool in the analysis of boolean functions is Fourier analysis. We introduce this tool in the next section.

2.3 Fourier Analysis

To describe the Fourier representation of boolean functions, we must first introduce the notion of characters for the boolean hypercube.

Definition 2.3. A *character* of the vector space $\{0,1\}^n$ is a function $\chi : \{0,1\}^n \to \{-1,1\}$ that satisfies

$$\chi(x \oplus y) = \chi(x) \cdot \chi(y)$$

for every $x, y \in \{0,1\}^n$.

In other words, a character χ is a group homomorphism between $\{0,1\}^n$ and $\{-1,1\}$. The following proposition identifies the set of characters for $\{0,1\}^n$.

Proposition 2.4. *For every* $\alpha \in \{0,1\}^n$*, the function* $\chi_\alpha : \{0,1\}^n \to \{-1,1\}$ *defined by*

$$\chi_\alpha(x) = (-1)^{\langle \alpha, x \rangle}$$

is a character of $\{0,1\}^n$*.*

Proof. For any $x, y \in \{0,1\}^n$,

$$\chi_\alpha(x+y) = (-1)^{\langle \alpha, x+y \rangle} = (-1)^{\langle \alpha, x \rangle + \langle \alpha, y \rangle} = (-1)^{\langle \alpha, x \rangle} \cdot (-1)^{\langle \alpha, y \rangle} = \chi_\alpha(x) \cdot \chi_\alpha(y). \quad \square$$

Fact 2.5. *For any* $\alpha \in \{0,1\}^n$*, we have* $\mathbf{E}_x[\chi_\alpha(x)] = \begin{cases} 1 & \textit{if } \alpha = \mathbf{0} \\ 0 & \textit{otherwise.} \end{cases}$

We are now ready to introduce the Fourier transform and Fourier representation of boolean functions.

Definition 2.6 (Fourier transform)**.** The *Fourier transform* of the function $f : \{0,1\}^n \to \mathbb{R}$ is the function $\hat{f} : \{0,1\}^n \to \mathbb{R}$ defined by $\hat{f}(\alpha) = \langle f, \chi_\alpha \rangle$.

Definition 2.7 (Fourier representation)**.** The *Fourier representation* (or *Fourier decomposition*) of the function $f : \{0,1\}^n \to \mathbb{R}$ is

$$f(x) = \sum_{\alpha \in \{0,1\}^n} \hat{f}(\alpha) \chi_\alpha(x).$$

A fundamental property of the Fourier transform is that it preserves the squared L_2 norm of the function.

Theorem 2.8 (Parseval's identity)**.** *For any function* $f : \{0,1\}^n \to \mathbb{R}$*,*

$$\mathop{\mathbf{E}}_x[f(x)^2] = \|f\|_2^2 = \sum_{\alpha \in \{0,1\}^n} \hat{f}(\alpha)^2.$$

A useful consequence of Parseval's identity is that the distance between two boolean functions has a nice representation in terms of the Fourier transform of the functions.

Lemma 2.9. *For any boolean functions* $f, g : \{0,1\}^n \to \{0,1\}$*,*

$$\Pr_{x \in \{0,1\}^n}[f(x) \neq g(x)] = \sum_{\alpha \in \{0,1\}^n} (\hat{f}(\alpha) - \hat{g}(\alpha))^2.$$

Proof. Since f and g are $\{0,1\}$-valued, $\Pr_x[f(x) \neq g(x)] = \mathbf{E}_x[(f(x) - g(x))^2]$. By Parseval's identity, this means that

$$\Pr_x[f(x) \neq g(x)] = \mathop{\mathbf{E}}_x[(f(x) - g(x))^2] = \sum_{\alpha \in \{0,1\}^n} \widehat{f-g}(\alpha)^2.$$

The proof is completed by noting that for any $\alpha \in \{0,1\}^n$,

$$\widehat{f-g}(\alpha) = \langle f - g, \alpha \rangle = \langle f, \alpha \rangle - \langle g, \alpha \rangle = \hat{f}(\alpha) - \hat{g}(\alpha). \qquad \square$$

In Chapter 7, we shall also make use of a basic fact regarding the Fourier transform of the pushforward of boolean functions.

Definition 2.10 (Pushforward)**.** Fix a subspace $W \subseteq \{0,1\}^n$. The *pushforward* of the function $f : \{0,1\}^n \to \mathbb{R}$ by the linear function $g : \{0,1\}^n \to W$ is the function $g_*(f) : W \to \mathbb{R}$ defined by

$$(g_*(f))(x) := \frac{1}{2^n} \sum_{y \in g^{-1}(x)} f(y) = \mathop{\mathbf{E}}_{y \in \{0,1\}^n}[\mathbb{I}[g(y) = x] \cdot f(y)].$$

Fact 2.11. *Fix $W \subseteq \{0,1\}^n$. For any function $f : \{0,1\}^n \to \mathbb{R}$ and any linear function $g : \{0,1\}^n \to W$, the Fourier transform of $g_*(f)$ satisfies*

$$\widehat{g_*(f)}(\alpha) = \frac{1}{|W|} \hat{f}(\chi_\alpha \circ g).$$

2.4 Influence

Another important tool in the analysis of boolean functions is the notion of "influence" of the variables in the function. This section defines the notion formally and proves some basic properties of influence. As a first step, let us define what we mean by "relevant" and "irrelevant" variables.

Definition 2.12 (Relevant variables)**.** The variable $i \in [n]$ is *relevant* in the boolean function $f : \{0,1\}^n \to \{0,1\}$ if there exists an element $x \in \{0,1\}^n$ such that $f(x) \neq f(x \oplus e^i)$. If no such element exists, we say that i is *irrelevant* in f.

The definition of relevance extends in a natural way to sets of variables. We can do so by defining a set $S \subseteq [n]$ to be *relevant* in the function f if some variable $i \in S$ is relevant in S. Equivalently, we can define relevant sets directly as follows.

Definition 2.13 (Relevant sets of variables)**.** The set of variables $S \subseteq [n]$ is *relevant* in $f : \{0,1\}^n \to \{0,1\}$ if there exists two elements $x, y \in \{0,1\}^n$ such that $x_{\bar{S}} = y_{\bar{S}}$ and $f(x) \neq f(y)$.

We can refine the notion of relevance to measure to take into account the fraction of elements $x \in \{0,1\}^n$ for which $f(x) \neq f(x \oplus e^i)$. The result is our definition of influence.

Definition 2.14 (Influence of a variable)**.** The *influence* of the variable $i \in [n]$ in the boolean function $f : \{0,1\}^n \to \{0,1\}$ is

$$\mathrm{Inf}_f(i) = \tfrac{1}{2} \Pr_x[f(x) \neq f(x \oplus e^i)]$$

where the probability is taken over the uniform distribution of x in $\{0,1\}^n$.

We can also introduce a similar definition to measure the influence of sets of variables.

Definition 2.15 (Influence of a set of variables)**.** The *influence* of the set $S \subseteq [n]$ of variables in the boolean function $f : \{0,1\}^n \to \{0,1\}$ is

$$\mathrm{Inf}_f(S) = \Pr_{x,y}[f(x) \neq f(x_{\bar{S}} \vee y_S)]$$

where the probability is taken over the uniform distribution of x and y in $\{0,1\}^n$.

The reader may find it slightly puzzling that the definition of influence of variables includes a factor of $\frac{1}{2}$. This is done so that the influence of a variable is equal to the influence of a singleton set containing that variable.

Fact 2.16. *For any function $f : \{0,1\}^n \to \{0,1\}$ and any variable $i \in [n]$, we have* $\mathrm{Inf}_f(i) = \mathrm{Inf}_f(\{i\})$.

An immediate observation that follows from our definitions of relevance and influence is that the variables that are relevant in a function are exactly the variables with non-zero influence in that function.

Fact 2.17. *The variable $i \in [n]$ is relevant in $f : \{0,1\}^n \to \{0,1\}$ iff* $\mathrm{Inf}_f(i) > 0$. *Similarly, the set $S \subseteq [n]$ of variables is relevant in f iff* $\mathrm{Inf}_f(S) > 0$.

A less obvious—but much more useful—observation is that the notions of influence have a natural representation in terms of the Fourier transformation of a boolean function.

Lemma 2.18. *For every function $f : \{0,1\}^n \to \{0,1\}$ and every variable $i \in [n]$, the influence of i in f satisfies*

$$\mathrm{Inf}_f(i) = 2 \sum_{\alpha \in \{0,1\}^n \,:\, \alpha_i = 1} \hat{f}(\alpha)^2.$$

More generally, for every set $S \subseteq [n]$, the influence of S in f satisfies

$$\mathrm{Inf}_f(S) = 2 \sum_{\alpha \in \{0,1\}^n \,:\, \|\alpha_S\| > 0} \hat{f}(\alpha)^2.$$

Proof. By Fact 2.16, it suffices to prove the latter identity.

Since f is $\{0,1\}$-valued, we have

$$\begin{aligned}\Pr_{x,y}[f(x) \neq f(x_{\bar{S}} \vee y_S)] &= \mathop{\mathbf{E}}_{x,y}[(f(x) - f(x_{\bar{S}} \vee y_S))^2] \\ &= 2 \mathop{\mathbf{E}}_{x}[f(x)^2] - 2 \mathop{\mathbf{E}}_{x,y}[f(x) f(x_{\bar{S}} \vee y_S)].\end{aligned}$$

The Fourier representation of f and linearity of expectation yield

$$\mathop{\mathbf{E}}_{x,y}[f(x) f(x_{\bar{S}} \vee y_S)] = \sum_{\alpha,\beta \in \{0,1\}^n} \hat{f}(\alpha)\hat{f}(\beta) \mathop{\mathbf{E}}_{x,y}[\chi_\alpha(x)\chi_\beta(x_{\bar{S}} \vee y_S)].$$

By the identity $\chi_\beta(x_{\bar{S}} \vee y_S) = \chi_{\beta_{\bar{S}}}(x)\chi_{\beta_S}(y)$, we can rewrite the right-most term as $\mathbf{E}_{x,y}[\chi_\alpha(x)\chi_\beta(x_{\bar{S}} \vee y_S)] = \mathbf{E}_x[\chi_{\alpha \oplus \beta_{\bar{S}}}(x)]\, \mathbf{E}_y[\chi_{\beta_S}(y)]$. Applying Fact 2.5, this means that $\mathbf{E}_{x,y}[\chi_\alpha(x)\chi_\beta(x_{\bar{S}} \vee y_S)]$ takes the value 1 when $\alpha = \beta_{\bar{S}}$ and $\beta_S = \mathbf{0}$ (or, equivalently, when $\alpha_S = \mathbf{0}$ and $\beta = \alpha$) and it takes the value 0 otherwise. This observation and Parseval's identity imply that

$$\begin{aligned}\Pr_{x,y}[f(x) \neq f(x_{\bar{S}} \vee y_S)] &= 2 \mathop{\mathbf{E}}_{x}[f(x)^2] - 2 \mathop{\mathbf{E}}_{x,y}[f(x) f(x_{\bar{S}} \vee y_S)] \\ &= 2 \sum_{\alpha} \hat{f}(\alpha)^2 - 2 \sum_{\alpha \,:\, \alpha_S = \mathbf{0}} \hat{f}(\alpha)^2 \;=\; 2 \sum_{\alpha \,:\, \|\alpha_S\| > 0} \hat{f}(\alpha)^2. \qquad \square\end{aligned}$$

The monotonicity and subadditivity properties of influence follow directly from the lemma.

Theorem 2.19 (Monotonicity and subadditivity of influence)**.** *For any boolean function $f : \{0,1\}^n \to \{0,1\}$ and any two sets $S, T \subseteq [n]$,*

$$\mathrm{Inf}_f(S) \le \mathrm{Inf}_f(S \cup T) \le \mathrm{Inf}_f(S) + \mathrm{Inf}_f(T).$$

Proof. Any $\alpha \in \{0,1\}^n$ that satisfies $\|\alpha_S\| > 0$ also satisfies $\|\alpha_{S\cup T}\| > 0$. Therefore, by Lemma 2.18,

$$\mathrm{Inf}_f(S) = 2 \sum_{\alpha\,:\,\|\alpha_S\|>0} \hat{f}(\alpha)^2 \le 2 \sum_{\alpha\,:\,\|\alpha_{S\cup T}\|>0} \hat{f}(\alpha)^2 = \mathrm{Inf}_f(S\cup T).$$

Similarly, any $\alpha \in \{0,1\}^n$ that satisfies $\|\alpha_{S\cup T}\| > 0$ must satisfy at least one of the two inequalities $\|\alpha_S\| > 0$ and $\|\alpha_T\| > 0$. So

$$\begin{aligned}\mathrm{Inf}_f(S\cup T) &= 2 \sum_{\alpha\,:\,\|\alpha_{S\cup T}\|>0} \hat{f}(\alpha)^2 \\ &\le 2 \sum_{\alpha\,:\,\|\alpha_S\|>0} \hat{f}(\alpha)^2 + 2 \sum_{\alpha\,:\,\|\alpha_T\|>0} \hat{f}(\alpha)^2 \\ &= \mathrm{Inf}_f(S) + \mathrm{Inf}_f(T).\end{aligned}$$

□

2.5 Juntas

Juntas figure prominently in the rest of this thesis. In this section, we define the term more formally and provide a basic fact about functions that are “far” from being juntas for later use. A more complete introduction to juntas and their role in property testing is deferred to Chapter 5.

Definition 2.20 (Junta)**.** Fix $0 \le k \le n$. The function $f : \{0,1\}^n \to \{0,1\}$ is a *k-junta* iff it contains at most k relevant variables.

Note that when we say that f is a *junta* (without specifying k), we mean that f is a k-junta for some $k = O(1)$. (I.e., for some k independent of n.)

A simple characterization of the influence of variables in functions that are far from juntas is as follows.

Lemma 2.21. *Fix $0 < k < n$. Let $f : \{0,1\}^n \to \{0,1\}$ be ϵ-far from all functions that are k-juntas. Then every set $J \subseteq [n]$ of size $|J| \le k$ satisfies* $\mathrm{Inf}_f(\overline{J}) \ge \epsilon$.

Proof. Fix any set $J \subseteq [n]$ of size $|J| \le k$. Define $h : \{0,1\}^n \to \{0,1\}$ to be the function obtained by setting

$$h(x) = \begin{cases} 1 & \text{if } \mathbf{E}_z[f(x_J \vee z_{\overline{J}})] \ge \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

for every $x \in \{0,1\}^n$. The variables that are relevant in h are contained in J, so h is a k-junta. By the assumption in the lemma statement, we must therefore have that

$$\Pr_x[f(x) \neq h(x)] \geq \epsilon.$$

For each $w \in \{0,1\}^n$, define

$$p_w := \Pr[f(x) \neq h(x) \mid x_J = w_J].$$

By the definition of h, for every w we have $p_w \leq \frac{1}{2}$. We also have that

$$\Pr_x[f(x) \neq h(x)] = \mathop{\mathbf{E}}_x[p_{x_J}]$$

and that

$$\mathrm{Inf}_f(\overline{J}) = \Pr_x[f(x) \neq f(x_J \vee z_{\overline{J}})] = 2\mathop{\mathbf{E}}_x[p_{x_J}(1-p_{x_J})].$$

Combining the above results, we get

$$\mathrm{Inf}_f(\overline{J}) \geq 2\mathop{\mathbf{E}}_x[\tfrac{1}{2}p_{x_J}] = \Pr_x[f(x) \neq h(x)] \geq \epsilon. \qquad \square$$

2.6 Parity functions

Another class of functions that appears prominently throughout this thesis—and, indeed, that often appears in almost any study of boolean functions—is the set of parity functions.

Definition 2.22 (Parity)**.** Fix $S \subseteq [n]$. The *parity* function corresponding to S is the function $\mathrm{Parity}_S : \{0,1\}^n \to \{0,1\}$ defined by

$$\mathrm{Parity}_S(x) = \bigoplus_{i \in S} x_i.$$

The parity functions are *linear* functions since they satisfy the identity

$$\mathrm{Parity}_S(x \oplus y) = \mathrm{Parity}_S(x) \oplus \mathrm{Parity}_S(y)$$

for every $x, y \in \{0,1\}^n$. When $|S| = k$, we also say that the function Parity_S is *k-linear*.

The parity functions have a simple Fourier representation.

Proposition 2.23. *Fix $S \subseteq [n]$. The Fourier coefficients of the parity function* $\mathrm{Parity}_S : \{0,1\}^n \to \{0,1\}$ *are defined by*

$$\widehat{\mathrm{Parity}}_S(\alpha) = \begin{cases} \frac{1}{2} & \textit{if } \alpha = \mathbf{0} \textit{ or } \alpha = e^S \\ 0 & \textit{otherwise.} \end{cases}$$

We can use the Fourier decomposition of parity functions to establish the following useful facts.

Proposition 2.24. *Fix $0 < k < n$ and let $S \subseteq [n]$ be a set of size $|S| > k$. Then the parity function* $\mathrm{Parity}_S : \{0,1\}^n \to \{0,1\}$ *is*

i. $\frac{1}{2}$*-far from* Parity_T *for every $T \neq S$,*

ii. $\frac{1}{2}$*-far from every k-linear function,*

iii. $\frac{1}{2}$*-far from every k-junta, and*

iv. $\frac{1}{2}$*-far from all functions of Fourier degree at most k.*

Proof. By Lemma 2.9 and Proposition 2.23,

$$\Pr_x[\mathrm{Parity}_S(x) \neq \mathrm{Parity}_T(x)] = \sum_\alpha \left(\widehat{\mathrm{Parity}}_S(\alpha) - \widehat{\mathrm{Parity}}_T(\alpha)\right)^2 = 2(\tfrac{1}{2})^2 = \tfrac{1}{2}.$$

This completes the proof of i. It also immediately implies ii since k-linear functions are necessarily parity functions on some set $T \neq S$. To prove iii and iv, it suffices to prove the latter statement, since k-juntas have Fourier degree at most k.

Let $f : \{0,1\}^n \to \{0,1\}$ be a function of Fourier degree at most k. Since $|S| > k$, we can again apply Lemma 2.9 and Proposition 2.23 to obtain

$$\Pr_x[f(x) \neq \mathrm{Parity}_S(x)] = \sum_\alpha \left(\hat{f}(\alpha) - \widehat{\mathrm{Parity}}_S(\alpha)\right)^2 = (\hat{f}(\mathbf{0}) - \tfrac{1}{2})^2 + \sum_{0 < \|\alpha\| \leq k} \hat{f}(\alpha)^2 + (\tfrac{1}{2})^2.$$

By Parseval's identity and the Fourier degree of f,

$$\sum_{0 < \|\alpha\| \leq k} \hat{f}(\alpha)^2 = \sum_{\|\alpha\| > 0} \hat{f}(\alpha)^2 = \|f\|_2^2 - \hat{f}(\mathbf{0})^2.$$

The function f is $\{0,1\}$-valued, so $\|f\|_2^2 = \mathbf{E}_x[f(x)^2] = \mathbf{E}_x[f(x)] = \hat{f}(\mathbf{0})$. As a result,

$$\Pr_x[f(x) \neq \mathrm{Parity}_S(x)] = (\hat{f}(\mathbf{0}) - \tfrac{1}{2})^2 + (\hat{f}(\mathbf{0}) - \hat{f}(\mathbf{0})^2) + \tfrac{1}{4} = \tfrac{1}{2}. \qquad \square$$

Chapter 3

Property Testing

This section presents a brief introduction to property testing. We introduce all the necessary definitions, then explore two fundamental topics that we will use in later sections: the connection between property testing and learning theory, and a main lemma for proving lower bounds on the query complexity for testing properties via Yao's Minimax Lemma. For a more thorough introduction to property testing, we recommend the surveys [87, 89] and the collection [58].

3.1 Definitions

Definition 3.1 (Property)**.** A *property* of boolean functions is a subset of all boolean functions.

When $f : \{0,1\}^n \to \{0,1\}$ is in $\mathcal{P}$, we say that f has property $\mathcal{P}$. We say that f is "far" from $\mathcal{P}$ when every function in $\mathcal{P}$ disagrees with f on a large fraction of the inputs in $\{0,1\}^n$. To make this notion precise, we introduce measures of distance for functions and for properties of boolean functions.

Definition 3.2 (Distance between functions)**.** Fix a distribution $\mathcal{D}$ over $\{0,1\}^n$. The distance between the two functions $f, g : \{0,1\}^n \to \{0,1\}$ under $\mathcal{D}$ is

$$\mathrm{dist}_{\mathcal{D}}(f, g) := \Pr_{x \sim \mathcal{D}}[f(x) \neq g(x)].$$

When $\mathcal{D}$ is the uniform distribution over $\{0,1\}^n$, we omit the subscript and write simply $\mathrm{dist}(f, g)$.

Definition 3.3 (Distance between properties)**.** Fix a distribution $\mathcal{D}$ over $\{0,1\}^n$. The distance between two properties $\mathcal{P}, \mathcal{Q} \subseteq \{0,1\}^n \to \{0,1\}$ is

$$\mathrm{dist}_{\mathcal{D}}(\mathcal{P}, \mathcal{Q}) := \min_{f \in \mathcal{P}, g \in \mathcal{Q}} \mathrm{dist}_{\mathcal{D}}(f, g).$$

Once again, we omit the subscript when $\mathcal{D}$ is the uniform distribution over $\{0,1\}^n$.

Our main measure of interest is the distance between a function and a property. We define this measure using the distance between properties.

Definition 3.4 (Distance between functions and properties)**.** Fix a distribution $\mathcal{D}$ over $\{0,1\}^n$. The distance between the function $f : \{0,1\}^n \to \{0,1\}$ and the property $\mathcal{P} \subseteq \{0,1\}^n \to \{0,1\}$ is

$$\mathrm{dist}_{\mathcal{D}}(f, \mathcal{P}) := \mathrm{dist}_{\mathcal{D}}(\{f\}, \mathcal{P}).$$

As above, we omit the subscript when $\mathcal{D}$ is the uniform distribution over $\{0,1\}^n$.

When $\mathrm{dist}_{\mathcal{D}}(f, \mathcal{P}) \geq \epsilon$, we say that f is *ϵ-far* from $\mathcal{P}$ (under the distribution $\mathcal{D}$). When $\mathrm{dist}_{\mathcal{D}}(f, \mathcal{P}) \leq \epsilon$, then f is *ϵ-close* to $\mathcal{P}$.

We are now ready to present the notion of property testers, as originally defined by Rubinfeld and Sudan [90].

Definition 3.5 (Property tester)**.** Fix a property $\mathcal{P} \subseteq \{0,1\}^n \to \{0,1\}$ and a distribution $\mathcal{D}$ over $\{0,1\}^n$. A *q-query ϵ-tester* for $\mathcal{P}$ over $\mathcal{D}$ is a randomized algorithm that queries an unknown function $f : \{0,1\}^n \to \{0,1\}$ on at most q inputs and

(i) Accepts with probability at least $\frac{2}{3}$ when f has property $\mathcal{P}$; and

(ii) Rejects with probability at least $\frac{2}{3}$ when f is ϵ-far from $\mathcal{P}$.

Remark 3.6. Note that the tester is free to accept or reject f when $0 < \mathrm{dist}_{\mathcal{D}}(f, \mathcal{P}) < \epsilon$.

Remark 3.7. The choice of $\frac{2}{3}$ for the acceptance and rejection probabilities is somewhat arbitrary. More generally, we could require those probabilities to be $1 - \delta$ for any $0 < \delta < \frac{1}{2}$. Standard boosting arguments show that the two settings are essentially equivalent, so we do not bother with the more general definition in this thesis.

Definition 3.8 (One-sided error). A q-query ϵ-tester for $\mathcal{P}$ that is guaranteed to always accept functions with property $\mathcal{P}$ (instead of only accepting them with probability at least $\frac{2}{3}$) is said to have *one-sided error*. Otherwise, $\mathcal{P}$ has *two-sided error*.

Definition 3.9 (Non-adaptive testers)**.** A q-query ϵ-tester for $\mathcal{P}$ that determines all of its q queries before observing the value of f on any of those queries is *non-adaptive*. Otherwise, the tester is *adaptive*.

A well-known folkloric result in property testing states that the most query-efficient non-adaptive tester for a property $\mathcal{P}$ has query complexity that is at most exponential in the query complexity of the best adaptive tester for $\mathcal{P}$.

Proposition 3.10. *Fix a property $\mathcal{P} \subseteq \{0,1\}^n \to \{0,1\}$. If there is an adaptive q-query ϵ-tester for $\mathcal{P}$, then there is a non-adaptive 2^q-query ϵ-tester for $\mathcal{P}$.*

Proof. Let $\mathcal{A}$ be the adaptive q-query ϵ-tester for $\mathcal{P}$. The behavior of $\mathcal{A}$ is determined by a decision tree of depth d. By fixing the randomness in advance, we can simulate $\mathcal{A}$ with a non-adaptive algorithm by making all the (at most) 2^d queries that are present in the tree then following the path that $\mathcal{A}$ would have traversed in making its d adaptive queries. □

3.2 Testing and Learning

The connection between property testing and learning theory that we describe in this section was first established by Goldreich, Goldwasser, and Ron [59]. Before describing the connection itself, let us first define the concept of a "proper learner".

Definition 3.11 (Proper learner)**.** An algorithm $\mathcal{A}$ with query access to a boolean function is an *(ϵ,δ,q) proper learner* for property $\mathcal{P}$ over the distribution $\mathcal{D}$ if for every function $f : \{0,1\}^n \to \{0,1\}$ in $\mathcal{P}$, the algorithm $\mathcal{A}$ queries f on at most q inputs from $\{0,1\}^n$ then outputs a function $h : \{0,1\}^n \to \{0,1\}$ in $\mathcal{P}$ such that with probability at least $1-\delta$, the distance between f and h is bounded by $\mathrm{dist}_{\mathcal{D}}(f,h) \le \epsilon$.

Remark 3.12. We emphasize that the learner is free to output *any* function $h \in \mathcal{P}$ when $f \notin \mathcal{P}$.

Our definition corresponds to the concept of a proper learner in the *membership query* learning model. We will examine other learning models in Chapter 12. Also, the common terminology in the learning theory community refers to learners over *classes* of functions instead of over properties; the two definitions are identical.

Proper learners can be used to test properties, as the following lemma shows.

Lemma 3.13. *If $\mathcal{P}$ has a ($\frac{\epsilon}{2}$,$\frac{1}{6}$,q) proper learner over $\mathcal{D}$, then $\mathcal{P}$ can be tested with $q + O(1/\epsilon)$ queries.*

Proof. Let $\mathcal{A}$ be a ($\frac{\epsilon}{2}$,$\frac{1}{6}$,q) proper learner over $\mathcal{D}$ for $\mathcal{P}$. We can use $\mathcal{A}$ to test $\mathcal{P}$ by running the following simple testing algorithm:

1. Run $\mathcal{A}$ on f to obtain the hypothesis function $h : \{0,1\}^n \to \{0,1\}$.

2. Query f on $s = O(1/\epsilon)$ samples drawn independently at random from $\mathcal{D}$.

3. Let $\tilde{d}$ be the fraction of inputs chosen in the last step on which f and h disagree.

4. Accept iff $\tilde{d} \leq \frac{3\epsilon}{4}$.

Let us now examine why this algorithm is a valid tester for $\mathcal{P}$. First, when $f \in \mathcal{P}$, with probability at least $\frac{5}{6}$, $\mathcal{A}$ returns a function h that satisfies $\mathrm{dist}_{\mathcal{D}}(f,h) \leq \frac{\epsilon}{2}$. Conversely, when f is ϵ-far from $\mathcal{P}$, then no matter what hypothesis function $h \in \mathcal{P}$ is returned by the learner, $\mathrm{dist}_{\mathcal{D}}(f,h) \geq \epsilon$. Since $\tilde{d}$ is an unbiased estimator for $\mathrm{dist}_{\mathcal{D}}(f,h)$, we can pick s to be large enough to guarantee that $\Pr[|\tilde{d} - \mathrm{dist}_{\mathcal{D}}(f,h)| \geq \frac{\epsilon}{4}] < \frac{5}{6}$ and the resulting algorithm is a valid tester for $\mathcal{P}$. □

When the property $\mathcal{P}$ is small, there is a simple but query-efficient proper learning algorithm for target functions in $\mathcal{P}$: draw $O(\log|\mathcal{P}|)$ random samples, and output any hypothesis in $\mathcal{P}$ that is consistent with the function on all the samples. As a result, Lemma 3.13 has the following widely-applicable corollary.

Corollary 3.14. *Fix $\mathcal{P} \subseteq \{0,1\}^n \to \{0,1\}$. There is an ϵ-tester for $\mathcal{P}$ that requires only $O(\log(|\mathcal{P}|)/\epsilon)$ queries.*

Proof. By Occam's Razor (see, e.g. [73, §2]), there is an $(\frac{\epsilon}{2}, \frac{1}{6}, q)$ proper learner for $\mathcal{P}$ that makes $q = O(\log(|\mathcal{P}|)/\epsilon)$ queries to the target function.[1] The corollary then follows immediately from Lemma 3.13. □

3.3 Lower Bounds via Yao's Minimax Principle

We use the following standard property testing lemmas in the following sections.

[1] In fact, as discussed above, the proper learner only needs to receive *q random samples* drawn from $\{0,1\}^n$.

Lemma 3.15. *Let $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$ be any two distributions over functions $\{0,1\}^n \to \{0,1\}$. If for every set $X \subseteq \{0,1\}^n$ of size $|X| = q$ and any vector $r \in \{0,1\}^q$ we have that*

$$\left| \Pr_{f\sim\mathcal{D}_{\text{yes}}}[f(X) = r] - \Pr_{f\sim\mathcal{D}_{\text{no}}}[f(X) = r] \right| < \tfrac{1}{36}\, 2^{-q},$$

then any algorithm that distinguishes functions drawn from $\mathcal{D}_{\text{yes}}$ from those drawn from $\mathcal{D}_{\text{no}}$ with probability at least $\frac{2}{3}$ makes at least $q+1$ queries.

Proof. Define $\mathcal{D}$ to be the distribution obtained by drawing a function from $\mathcal{D}_{\text{yes}}$ or from $\mathcal{D}_{\text{no}}$, each with probability $1/2$. By Yao's Minimax Principle[100], to prove the lemma it suffices to show that any deterministic testing algorithm needs at least $q+1$ queries to distinguish functions drawn from $\mathcal{D}_{\text{yes}}$ or from $\mathcal{D}_{\text{no}}$ with probability at least $2/3$.

A deterministic testing algorithm can be described by a decision tree with a query $x \in \{0,1\}^n$ at each internal node and a decision to accept or reject at every leaf. Each boolean function f defines a path through the tree according to the value of $f(x)$ at each internal node.

Consider a testing algorithm that makes at most q queries. Its decision tree has depth at most q and it has at most 2^q leaves. Let us call a leaf ℓ *negligible* if the probability that a function $f \sim \mathcal{D}$ defines a path that terminates at ℓ is at most $\frac{1}{12}2^{-d}$. The total probability that $f \sim \mathcal{D}$ defines a path to a negligible leaf is at most $\frac{1}{12}$.

Fix ℓ to be some non-negligible leaf. This leaf corresponds to a set $X \subseteq \{0,1\}^n$ of q queries and a vector $r \in \{0,1\}^q$ of responses; a function f defines a path to the leaf ℓ iff $f(X) = r$. Since ℓ is non-negligible, $\Pr_{f\sim\mathcal{D}}[f(X) = r] > \frac{1}{12}2^{-d}$. So by the hypothesis of the lemma,

$$\left| \Pr_{f\sim\mathcal{D}_{\text{yes}}}[f(X) = r] - \Pr_{f\sim\mathcal{D}_{\text{no}}}[f(X) = r] \right| \le \tfrac{1}{36}\, 2^{-d} < \tfrac{1}{3} \Pr_{f\sim\mathcal{D}}[f(X) = r].$$

Then by Bayes' theorem

$$\begin{aligned} &\left| \Pr_{f\sim\mathcal{D}}[f \in \mathcal{P} \mid f(X) = r] - \Pr_{f\sim\mathcal{D}}[f \text{ } \epsilon\text{-far from } \mathcal{P} \mid f(X) = r] \right| \\ &\qquad = \left| \frac{\Pr_{f\sim\mathcal{D}_{\text{yes}}}[f(X) = r] - \Pr_{f\sim\mathcal{D}_{\text{no}}}[f(X) = r]}{2\Pr_{f\sim\mathcal{D}}[f(X) = r]} \right| < \frac{1}{6}. \end{aligned}$$

Therefore, the probability that the testing algorithm correctly classifies a function $f \sim \mathcal{D}$ that lands at a non-negligible leaf ℓ is less than $\frac{7}{12}$. So even if the algorithm correctly classifies all functions that land in negligible leaves, it still correctly classifies f with probability less than $\frac{11}{12} \cdot \frac{7}{12} + \frac{1}{12} < \frac{2}{3}$, so it is not a valid tester for $\mathcal{P}$. □

Chapter 4

Mathematical Tools

This section introduces the mathematical tools that we use in subsequent chapters to analyze property testing algorithms and to establish lower bounds on the query complexity for different property testing tasks.

Many of the tools that we use are taken from probability theory and statistics. We introduce those tools first. In the subsequent sections, we then introduce the results from combinatorics and from the theory of orthogonal polynomials that we will require in the following chapters.

4.1 Probability theory

We adopt most of the standard terminology of probability theory as found, for example, in [50, 51].

Many of our in later chapter require us to show that two related distributions are "close". We formalize this concept with the total variation distance between distributions.

Definition 4.1. Given two random variables X, Y defined on a common discrete sample space Ω, the *total variation* distance between X and Y is

$$\mathrm{d_{TV}}(X, Y) = \frac{1}{2} \sum_{\omega \in \Omega} \left| \Pr[X = \omega] - \Pr[Y = \omega] \right| .$$

The most basic statistics of a random variable x drawn from some distribution $\mathcal{D}$ are its mean $\mathbf{E}[x]$ and variance $\mathbf{Var}[x] = \mathbf{E}[(x - \mathbf{E}[x])^2]$. When $x = (x_1, \ldots, x_n)$ is a random variable drawn from a multivariate distribution $\mathcal{D}'$, we define its mean to be the vector $\mathbf{E}[x] = (\mathbf{E}[x_1], \ldots, \mathbf{E}[x_n])$ and we define the *covariance* matrix of x to be the $n \times n$ matrix $\mathbf{Cov}[x]$ whose (i, j)th entry is defined by

$$\mathbf{Cov}[x]_{i,j} = \mathbf{E}[(x_i - \mathbf{E}[x_i])(x_j - \mathbf{E}[x_j])] = \mathbf{E}[x_i x_j] - \mathbf{E}[x_i]\,\mathbf{E}[x_j].$$

4.1.1 Hypergeometric Distribution

Consider the experiment where we have n balls, r of which are red, and we draw d balls uniformly at random without replacement. The distribution on the number w of balls drawn that are red is called the *hypergeometric* distribution. We write $\mathcal{H}_{n,r,d}$ to represent this distribution.

Intuitively, when $n \approx n'$ and $r \approx r'$, the distributions $\mathcal{H}_{n,r,d}$ and $\mathcal{H}_{n',r',d}$ should be close. The following lemma confirms and formalizes this intuition.

Lemma 4.2. *Let n, r, n', r', d be non-negative integers with $d, n' \leq \gamma n$ for some $\gamma \leq \frac{1}{2}$. Suppose that $|r - \frac{n}{2}| \leq t\sqrt{n}$ and $|r' - \frac{n'}{2}| \leq t\sqrt{n'}$ hold for some $t \leq \frac{1}{100\gamma}$. Then,*

$$\mathrm{d}_{\mathrm{TV}}(\mathcal{H}_{n,r,d}, \mathcal{H}_{n-n',r-r',d}) \leq c(1+t)\gamma\ .$$

holds for some universal constant c.

Proof. Our proof uses the connection between hypergeometric distribution and the binomial distribution, which we denote by $\mathcal{B}_{n,p}$ (for n experiments, each with success probability p). Specifically, we use the following two lemmas.

Lemma 4.3 (Example 1 in [94])**.** $\mathrm{d}_{\mathrm{TV}}(\mathcal{H}_{n,r,d}, \mathcal{B}_{d,\frac{r}{n}}) \leq \frac{d}{n}$.

Lemma 4.4 ([1])**.** *Let $0 < p < 1$ and $0 < \delta < 1 - p$. Then,*

$$\mathrm{d}_{\mathrm{TV}}(\mathcal{B}_{n,p}, \mathcal{B}_{n,p+\delta}) \leq \frac{\sqrt{e}}{2} \frac{\tau_{n,p}(\delta)}{(1 - \tau_{n,p}(\delta))^2}$$

provided $\tau_{n,p}(\delta) < 1$ where

$$\tau_{n,p}(\delta) = \delta \sqrt{\frac{n+2}{2p(1-p)}}\ .$$

The lemma trivially holds when $k = 0$ so from now on we assume $k \geq 1$. By the triangle inequality,

$$\begin{aligned} \mathrm{d_{TV}}(\mathcal{H}_{n,r,d}, \mathcal{H}_{n-n',r-r',d}) \leq\ & \mathrm{d_{TV}}(\mathcal{H}_{n,r,d}, \mathcal{B}_{d,p}) \\ & + \mathrm{d_{TV}}(\mathcal{H}_{n-n',r-r',d}, \mathcal{B}_{d,p'}) + \mathrm{d_{TV}}(\mathcal{B}_{d,p}, \mathcal{B}_{d,p'}) \end{aligned} \tag{4.1}$$

where $p = \frac{r}{n}$ and $p' = \frac{r-r'}{n-n'}$.

Now, we assume $p \leq p'$ (the other case can be treated in the same manner). Let $\delta = p' - p$, then

$$\begin{aligned} \delta &= \frac{mn' - nm'}{n(n-n')} \leq \frac{1}{n(n-n')} \left(\left(\frac{n}{2} + t\sqrt{n} \right) n' - n \left(\frac{n'}{2} - t\sqrt{n'} \right) \right) \\ &= \frac{t(n\sqrt{n'} + \sqrt{n}n')}{n(n-n')} \leq \frac{2t\sqrt{\gamma}n^{3/2}}{(1-\gamma)n^2} \leq 4t\sqrt{\frac{\gamma}{n}} \quad (\text{from } \gamma \leq \frac{1}{2}) \ . \end{aligned}$$

Then, $\tau_{k,p}(\delta)$ in Lemma 4.4 is

$$\begin{aligned} \tau_{k,p}(\delta) &\leq 4t\sqrt{\frac{\gamma}{n}}\sqrt{\frac{k+2}{2p(1-p)}} \leq 4t\sqrt{\frac{2\gamma(k+2)}{n}} \quad (\text{from } \frac{1}{p(1-p)} \leq 4) \\ &\leq 4t\sqrt{\frac{6\gamma k}{n}} \quad (\text{from } k \geq 1) \\ &\leq 4\sqrt{6}t\gamma \quad (\text{from } k \leq \gamma n) \end{aligned}$$

Note that, from the assumption, we have $\tau_{k,p}(\delta) \leq \frac{1}{2}$. From Lemmas 4.3 and 4.4, we have

$$\begin{aligned} (4.1) &\leq \frac{k}{n} + \frac{k}{n} + \frac{\sqrt{e}}{2} \frac{\tau_{k,p}(\delta)}{(1-\tau_{k,p}(\delta))^2} \\ &\leq 2\gamma + 2\sqrt{e} \cdot 4\sqrt{6}t\gamma \quad (\text{from } \tau_{k,p}(\delta) \leq \frac{1}{2}) \\ &\leq c(1+t)\gamma \end{aligned}$$

for some universal constant c. □

4.1.2 Random permutations

A *permutation* $\pi : [n] \to [n]$ is a bijection on $[n]$. The set of permutations on $[n]$ is denoted by $\mathcal{S}_n$. Fix $i \in [n]$. When $\pi(i) = i$, we say that i is a *fixed point* of π. We write $\mathcal{S}_T$ to represent the set of permutations π where each element $i \in [n] \setminus T$ is a fixed point in π.

A permutation $\pi \in \mathcal{S}_n$ acts on vectors in $\{0,1\}^n$ in the natural way: for $x \in \{0,1\}^n$, we define

$$\pi x = (x_{\pi(1)}, \dots, x_{\pi(n)}).$$

In Chapter 6, we will be interested in bounding the total variation distance between permuted strings under slightly different choices of random permutations. The following lemma shows that this distance is equal to the total variation distance between related hypergeometric distributions.

Lemma 4.5. *Fix $J, K \subseteq [n]$ and $x \in \{0,1\}^n$. Let $\mathcal{D}_{\pi_{J\cup K}x}$ and $\mathcal{D}_{\pi_J \pi_K x}$ be the distributions on $\pi_{J\cup K}x$ and $\pi_J\pi_K x$, respectively, when $\pi_{J\cup K}, \pi_J, \pi_K$ are drawn uniformly at random from $\mathcal{S}_{J\cup K}, \mathcal{S}_J, \mathcal{S}_K$, respectively. Then*

$$\mathrm{d_{TV}}(D_{\pi_{J\cup K}x}, D_{\pi_J\pi_K x}) = \mathrm{d_{TV}}(\mathcal{H}_{|J\cup K|, |x_{J\cup K}|, |K\setminus J|}, \mathcal{H}_{|K|,|x_K|,|K\setminus J|}).$$

Proof. Since both distributions $D_{\pi_{J\cup K}x}$ and $D_{\pi_J\pi_K x}$ only modify coordinates in $J \cup K$, we can ignore all other coordinates. Moreover, it is in fact suffices to look only at the number of ones in the coordinates of $K \setminus J$ and $J \cup K$, which completely determines the distributions. Let D_z denote the uniform distribution over all elements $y \in \{0,1\}^n$ such that $|y| = |x|$, $y_{\overline{J\cup K}} = x_{\overline{J\cup K}}$ and $|y_{K\setminus J}| = z$. (This also fixes the number of ones in y_J.) Notice that this distribution is well defined only for values of z such that $\max\{0, |x_{J\cup K}| - |J|\} \le z \le \min\{|x_{J\cup K}|, |K \setminus J|\}$.

Given this notation, $D_{\pi_{J\cup K}x}$ can be looked at as choosing $z \sim \mathcal{H}_{|J\cup K|,|x_{J\cup K}|,|K\setminus J|}$ and returning $y \sim D_z$. This is because we apply a random permutation over all elements of $J \cup K$, and therefore the number of ones inside $K \setminus J$ is indeed distributed like z. Moreover, the order inside both sets $K \setminus J$ and J is uniform.

The distribution $D_{\pi_J\pi_K x}$ can be looked at as choosing $z \sim \mathcal{H}_{|K|,|x_K|,|K\setminus J|}$ and returning $y \sim D_z$. The number of ones in $K \setminus J$ is determined already after applying π_K. It is distributed like z as we care about the choice of $|K \setminus J|$ out of the $|K|$ elements, and $|x_K|$ of them are ones (and their order is uniform). Later, we apply a random permutation π_J over all other relevant coordinates, so the order of elements in J is also uniform.

Since the distributions D_z are disjoint for different values of z, this implies that the distance between the two distributions $D_{\pi_{J\cup K}x}$ and $D_{\pi_J\pi_K x}$ depends only on the number of ones chosen to be inside $K \setminus J$. Therefore we have

$$\mathrm{d_{TV}}(D_{\pi_{J\cup K}x}, D_{\pi_J\pi_K x}) = \mathrm{d_{TV}}(\mathcal{H}_{|J\cup K|,|x_{J\cup K}|,|K\setminus J|}, \mathcal{H}_{|K|,|x_K|,|K\setminus J|})$$

as required. □

4.1.3 U-statistics

Many of the statistics on functions $f : \{0,1\}^n \to \{0,1\}$ that we need to evaluate in this thesis are of the form $\tau = \mathbf{E}_x[\psi(f(x))]$ for some function $\psi : \mathbb{R} \to \mathbb{R}$. These statistics can be estimated efficiently by sampling inputs $x^1, \ldots, x^m$ independently at random and evaluating $\tilde{\tau} = m^{-1} \sum_{i=1}^{m} \psi(f(x^i))$. The error of $\tilde{\tau}$ can then be controlled by the Chernoff bound or other similar concentration inequalities.

In Chapter 12, we need to estimate a slightly different kind of statistic: one that is of the form $\tau = \mathbf{E}_{x,y}[\psi(f(x), f(y))]$ for some $\psi : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. The best way to estimate this statistic is by using U-statistics, a tool first introduced by Halmos [64] and Hoeffding [66].

Definition 4.6. The *U-statistic* (of order 2) with symmetric kernel function $g : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is the function $U_g^m : \mathbb{R}^n \to \mathbb{R}$ defined by

$$U_g^m(x^1, \ldots, x^m) := \binom{m}{2}^{-1} \sum_{1 \le i < j \le m} g(x^i, x^j).$$

Tight concentration bounds are known for U-statistics with well-behaved kernel functions. The specific result that we use in Chapter 12 is a Bernstein-type inequality due to Arcones [9].

Theorem 4.7 (Arcones [9])**.** *For a symmetric function $h : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, let $\Sigma^2 = \mathbf{E}_x[\mathbf{E}_y[h(x,y)]^2] - \mathbf{E}_{x,y}[h(x,y)]^2$, let $b = \|h - \mathbf{E}\,h\|_\infty$, and let $U_m(h)$ be a random variable obtained by drawing $x^1, \ldots, x^m$ independently at random and setting $U_m(h) = \binom{m}{2}^{-1} \sum_{i<j} h(x^i, x^j)$. Then for every $t > 0$,*

$$\Pr[|U_m(h) - \mathbf{E}\,h| > t] \le 4 \exp\left(\frac{mt^2}{8\Sigma^2 + 100bt}\right).$$

For details on U-statistics, Arcones' theorem and other related topics, see [44].

4.1.4 Random matrices

Our study of the active testing model in Chapter 12 and, more specifically, the lower bounds on the query complexity for testing linear threshold functions in this model, rely on the non-asymptotic analysis of random matrices. In this short section, we introduce the definitions and results we will need for our intended application. For a more thorough introduction to the subject, see [98].

We begin with some basic matrix definitions. Given an $n \times m$ matrix A with real entries $\{a_{i,j}\}_{i \in [n], j \in [m]}$, the *adjoint* (or *transpose* – the two are equivalent since A contains only real values) of A is the $m \times n$ matrix A^* whose (i,j)-th entry equals $a_{j,i}$. Let us write $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m$ to denote the eigenvalues of $\sqrt{A^* A}$. These values are the *singular values* of A. The matrix $A^* A$ is positive semidefinite, so the singular values of A are all non-negative. We write $\lambda_{\max}(A) = \lambda_1$ and $\lambda_{\min}(A) = \lambda_m$ to represent its largest and smallest singular values. Finally, the *induced norm* (or *operator norm*) of A is

$$\|A\| = \max_{x \in \mathbb{R}^m \setminus \{0\}} \frac{\|Ax\|_2}{\|x\|_2} = \max_{x \in \mathbb{R}^m : \|x\|_2^2 = 1} \|Ax\|_2.$$

For more details on these definitions, see any standard linear algebra text (e.g., [93]). We will also use the following strong concentration bounds on the singular values of random matrices.

Lemma 4.8 (See [98, Cor. 5.35]). *Let A be an $n \times m$ matrix whose entries are independent standard normal random variables. Then for any $t > 0$, the singular values of A satisfy*

$$\sqrt{n} - \sqrt{m} - t \leq \lambda_{\min}(A) \leq \lambda_{\max}(A) \leq \sqrt{n} + \sqrt{m} + t \tag{4.2}$$

with probability at least $1 - 2e^{-t^2/2}$.

The proof of this lemma follows from Talagrand's inequality and Gordon's Theorem for Gaussian matrices. See [98] for the details. The lemma implies the following corollary which we will use in the proof of our theorem.

Corollary 4.9. *Let A be an $n \times m$ matrix whose entries are independent standard normal random variables. For any $0 < t < \sqrt{n} - \sqrt{m}$, the $m \times m$ matrix $\frac{1}{n} A^* A$ satisfies both inequalities*

$$\left\| \tfrac{1}{n} A^* A - I \right\| \leq 3 \frac{\sqrt{m} + t}{\sqrt{n}} \qquad \text{and} \qquad \det\left(\tfrac{1}{n} A^* A\right) \geq e^{-m\left(\frac{(\sqrt{m}+t)^2}{n} + 2\frac{\sqrt{m}+t}{\sqrt{n}}\right)} \tag{4.3}$$

with probability at least $1 - 2e^{-t^2/2}$.

Proof. When there exists $0 < z < 1$ such that $1 - z \leq \frac{1}{\sqrt{n}} \lambda_{\max}(A) \leq 1 + z$, the identity $\frac{1}{\sqrt{n}} \lambda_{\max}(A) = \|\frac{1}{\sqrt{n}} A\| = \max_{\|x\|_2^2 = 1} \|\frac{1}{\sqrt{n}} Ax\|_2$ implies that

$$1 - 2z \leq (1 - z)^2 \leq \max_{\|x\|_2^2 = 1} \left\| \tfrac{1}{\sqrt{n}} Ax \right\|_2^2 \leq (1 + z)^2 \leq 1 + 3z.$$

These inequalities and the identity $\|\frac{1}{n}A^*A - I\| = \max_{\|x\|_2^2=1} \|\frac{1}{\sqrt{n}}Ax\|_2^2 - 1$ imply that $-2z \le \|\frac{1}{n}A^*A - I\| \le 3z$. Fixing $z = \frac{\sqrt{m}+t}{\sqrt{n}}$ and applying Lemma 4.8 completes the proof of the first inequality.

Recall that $\lambda_1 \ge \cdots \ge \lambda_m$ are the eigenvalues of $\sqrt{A^*A}$. Then

$$\det(\tfrac{1}{n}A^*A) = \frac{\det(\sqrt{A^*A})^2}{n} = \frac{(\lambda_1 \cdots \lambda_m)^2}{n} \ge \left(\frac{\lambda_m^2}{n}\right)^m = \left(\frac{\lambda_{\min}(A)^2}{n}\right)^m .$$

Lemma 4.8 and the elementary inequality $1 + x \le e^x$ complete the proof of the second inequality. □

4.2 Combinatorics

4.2.1 Intersecting families

A family $\mathcal{F}$ of subsets of $[n]$ is *t-intersecting* if for every pair of sets $S, T \in \mathcal{F}$, their intersection size is at least $|S \cap T| \ge t$. The family $\mathcal{F}$ is called *s-uniform* if all sets in the family have size s Erdős, Ko, and Rado [49] asked: what is the maximum size of a t-intersecting s-uniform family? They gave the answer to this question when $t = 1$, and a sequence of later works led to a complete solution for this question [49, 53, 99, 2].

More recently, Dinur and Safra [47] and Friedgut [55] considered a variant on the original question of Erdős, Ko, and Rado. For a fixed $0 < p < 1$, define the *p-biased* measure of a family $\mathcal{F}$ of subsets of $[n]$ to be

$$\mu_p(\mathcal{F}) = \Pr_J[J \in \mathcal{F}]$$

where J is a random subset of $[n]$ obtained by including each element $i \in [n]$ in J independently with probability p. We can now ask: for a fixed p, what is the *maximum* p-biased measure of a t-intersecting family? Dinur and Safra [47] showed that 2-intersecting families have small p-biased measure and Friedgut [55] showed how the same result also extends to t-intersecting families for any $t > 2$. Specifically, they obtained the following bound on the maximum biased measure of intersecting families.

Theorem 4.10 (Dinur and Safra [47]; Friedgut [55]). *Let $\mathcal{F}$ be a t-intersecting family of subsets of $[n]$ for some $t \ge 1$. For any $p < \frac{1}{t+1}$, the p-biased measure of $\mathcal{F}$ is bounded by $\mu_p(\mathcal{F}) \le p^t$.*

The original motivation of Dinur and Safra [47] in the study of intersecting families was an application in hardness of approximation for the vertex cover problem. As we will see in Chapters 5 and 6, Theorem 4.10 is also particularly useful for the analysis of algorithms for testing juntas and partial symmetry.

4.2.2 Graph colorings

The topic of graph coloring—of assigning colors to the vertices of a graph $G = (V, E)$ such that no two neighboring vertices share the same color—has developed into a broad area of research with many interesting problems and results [67, 78]).

In Chapter 9, we make use of a celebrated theorem of this area due to Hajnal and Szemerédi [60]. Recall that the degree of a vertex in a graph is the number of edges adjacent to that vertex. Hajnal and Szemerédi's theorem states that graphs with small maximum vertex degree can be colored with very few colors and, furthermore, that this coloring can be done in a way that each color is assigned to approximately the same number of vertices.

Theorem 4.11 (Hajnal–Szemerédi [60])**.** *Let G be a graph on n vertices with maximum vertex degree $\Delta(G) \le d$. Then G has a $(d+1)$-coloring in which all the color classes have size $\lfloor \frac{n}{d+1} \rfloor$ or $\lceil \frac{n}{d+1} \rceil$.*

4.3 Orthogonal polynomials

The last tools that we introduce in this section are orthogonal polynomials. The Krawtchouk polynomials, which have been quite useful in coding theory [97], will be used in Chapter 7 to give a strong lower bound on the query complexity for testing k-linearity. The Hermite polynomials, which we introduce afterwards, are used in Chapter 12 in the analysis of testers for linear threshold functions.

4.3.1 Krawtchouk polynomials

For $n > 0$ and $k = 0, 1, \ldots, n$, the (binary) *Krawtchouk polynomial* $K_k^n : \{0, 1, \ldots, n\} \to \mathbb{Z}$ is defined by

$$K_k^n(m) = \sum_{j=0}^{k} (-1)^j \binom{m}{j} \binom{n-m}{k-j}.$$

The generating function representation of the Krawtchouk polynomial $K_k^n(m)$ is

$$K_k^n(m) = [x^k]\,(1-x)^m(1+x)^{n-m}.$$

Krawtchouk polynomials have a number of useful properties (see, e.g., [96]). We make use of the following identities in our proofs:

Fact 4.12. *Fix $n > 0$. Then*

i. For every $2 \le k \le n$, $K_k^n(m) - K_{k-2}^n(m) = K_k^{n+2}(m+1)$.

ii. $\sum_{k=0}^n K_k^n(m)^2 = (-1)^m K_n^{2n}(2m)$.

iii. For every $0 \le d \le \frac{n}{2}$, $\sum_{j=0}^d \binom{d}{j}(-1)^j K_{\frac{n}{2}}^n(2j+2) = 2^{2d} K_{\frac{n}{2}-d}^{n-2d}(2)$.

iv. For every $-\frac{n}{2} \le k \le \frac{n}{2}$, $K_{\frac{n}{2}+k}^n(m) = \dfrac{2^{n-1} i^m}{\pi} \displaystyle\int_0^{2\pi} \sin^m\theta \cos^{n-m}\theta e^{i2k\theta}\,\mathrm{d}\theta$.

v. $K_n^{2n}(2m+1) = 0$ and $(-1)^m K_n^{2n}(2m)$ is positive and decreasing in $\min\{m, n-m\}$.

Proof. We prove each statement individually.

i. The first statement follows directly from the generating function representation of Krawtchouk polynomials.

$$\begin{aligned} K_k^n(m) - K_{k-2}^n(m) &= \big([x^k]\,(1-x)^m(1+x)^{n-m}\big) - \big([x^{k-2}]\,(1-x)^m(1+x)^{n-m}\big) \\ &= [x^k]\,(1-x)^m(1+x)^{n-m}(1-x^2) \\ &= [x^k]\,(1-x)^{m+1}(1+x)^{n-m+1} = K_k^{n+2}(m+1). \end{aligned}$$

ii. By some more elementary manipulation of generating functions, we have

$$\begin{aligned} K_k^n(m) &= [x^k]\,(1-x)^m(1+x)^{n-m} \\ &= [x^{-k}]\,(1-\tfrac{1}{x})^m(1+\tfrac{1}{x})^{n-m} \\ &= [x^{n-k}]\,(x-1)^m(x+1)^{n-m} = (-1)^m K_{n-k}^n(m). \end{aligned}$$

Therefore,

$$\sum_{k=0}^n K_k^n(m)^2 = (-1)^m \sum_{k=0}^n K_k^n(m) K_{n-k}^n(m).$$

The Cauchy product of two sequences $\{a_0, a_1, \ldots\}$ and $\{b_0, b_1, \ldots\}$ is

$$\Big(\sum_{n\geq 0} a_n\Big)\Big(\sum_{n\geq 0} b_n\Big) = \sum_{n\geq 0}\Big(\sum_{k=0}^{n} a_k b_{n-k}\Big).$$

Let $a_k = b_k = [x^k]\ (1-x)^m(1+x)^{n-m}$. Then $(\sum_{n\geq 0} a_n) = (1-x)^m(1+x)^{n-m}$ and

$$\sum_{k=0}^{n} K_k^n(m)K_{n-k}^n(m) = [x^n]\ (1-x)^{2m}(1+x)^{2(n-m)} = K_n^{2n}(2m).$$

iii. Considering generating functions and applying the binomial theorem, we get

$$\begin{aligned}\sum_{j=0}^{d}\binom{d}{j}(-1)^j K_n^{2n}(2j+2) &= [x^n]\ \sum_{j=0}^{d}\binom{d}{j}(-1)^j(1-x)^{2j+2}(1+x)^{2n-2j-2}\\ &= [x^n]\ (1-x)^2(1+x)^{2n-2d-2}\sum_{j=0}^{d}\binom{d}{j}\big(-(1-x)^2\big)^j\big((1+x)^2\big)^{d-j}\\ &= [x^n]\ (1-x)^2(1+x)^{2n-2d-2}(4x)^d = 2^{2d}K_{n-d}^{2(n-d)}(2).\end{aligned}$$

iv. By elementary manipulation of generating functions, we obtain

$$\begin{aligned}K_{\frac{n}{2}+k}^n(m) &= [x^{\frac{n}{2}+k}]\ (1-x)^m(1+x)^{n-m}\\ &= [x^k]\ (\tfrac{1}{\sqrt{x}}-\sqrt{x})^m(\tfrac{1}{\sqrt{x}}+\sqrt{x})^{n-m}\\ &= [x^{-2k}]\ (x-\tfrac{1}{x})^m(x+\tfrac{1}{x})^{n-m}.\end{aligned}$$

Applying Cauchy's integral formula to this expression, we get

$$K_{\frac{n}{2}+k}^n(m) = \frac{1}{2\pi}\int_0^{2\pi}(e^{i\theta}-e^{-i\theta})^m(e^{i\theta}+e^{-i\theta})^{n-m}e^{i2k\theta}\,\mathrm{d}\theta.$$

From the trigonometric identities $\sin\theta = \frac{e^{i\theta}-e^{-i\theta}}{2i}$ and $\cos\theta = \frac{e^{i\theta}+e^{-i\theta}}{2}$, we get

$$K_{\frac{n}{2}+k}^n(m) = \frac{2^n}{2\pi}i^m\int_0^{2\pi}\sin^m\theta\cos^{n-m}\theta e^{i2k\theta}\,\mathrm{d}\theta.$$

v. By the last statement, $K_n^{2n}(2m+1)$ is pure imaginary. Since it is also real, it must be 0.

The last statement also yields

$$\begin{aligned}(-1)^m K_n^{2n}(2m) &= \frac{2^{2n-1}}{\pi}\int_0^{2\pi} \sin^{2m}(\theta)\cos^{2n-2m}(\theta)d\theta \\ &= \frac{2^{2n-2}}{\pi}\int_0^{2\pi} \sin^{2m}(\theta)\cos^{2n-2m} + \cos(\theta)^{2m}\sin(\theta)^{2n-2m}(\theta)d\theta.\end{aligned}$$

By AM-GM, for fixed n, the integrand is a decreasing function of $\min\{m, n-m\}$. □

Krawtchouk polynomials are widely used in coding theory (see, e.g.,[97]) and in our proofs because of their close connection with the Fourier coefficients of the (Hamming weight indicator) function $I_{W_k} : \{0,1\}^n \to \{0,1\}$ defined by $I_{W_k}(x) = \mathbf{1}[\|x\| = k]$.

Fact 4.13. *Fix $n > 0$, $0 \le k \le n$, and $\alpha \in \{0,1\}^n$. Then $\widehat{I}_{W_k}(\alpha) = 2^{-n} K_k^n(|\alpha|)$.*

Proof. The Fourier coefficient of I_{W_k} at α is

$$\widehat{I}_{W_k}(\alpha) = 2^{-n} \sum_{x \in \{0,1\}^n : \|x\| = k} (-1)^{\alpha\cdot x} = 2^{-n}\sum_{j=0}^{k}(-1)^j \binom{|\alpha|}{j}\binom{n-|\alpha|}{k-j} = 2^{-n}K_k^n(|\alpha|).$$

□

4.3.2 Hermite polynomials

When considering the uniform distribution on $\{0,1\}^n$, we found that the set of linear functions, via the Fourier transform, played an important role in the analysis of boolean functions. When, instead, we consider the standard Gaussian distribution on $\mathbb{R}^n$, the Hermite polynomials play a similarly important role.

Definition 4.14. The *Hermite polynomials* are a set of polynomials

$$\begin{aligned} h_0(x) &= 1, \\ h_1(x) &= x, \\ h_2(x) &= \tfrac{1}{\sqrt{2}}(x^2 - 1), \\ &\vdots \end{aligned}$$

that form a complete orthogonal basis for (square-integrable) functions $f : \mathbb{R} \to \mathbb{R}$ over the inner product space defined by the inner product $\langle f, g\rangle = \mathbf{E}_x[f(x)g(x)]$, where the expectation is over the standard Gaussian distribution $\mathcal{N}(0,1)$.

The *Hermite decomposition* and *Hermite coefficients* of a function are defined as follows.

Definition 4.15. For any $S \in \mathbb{N}^n$, define $H_S = \prod_{i=1}^n h_{S_i}(x_i)$. The *Hermite coefficient* of $f : \mathbb{R}^n \to \mathbb{R}$ corresponding to S is $\hat{f}(S) = \langle f, H_S \rangle = \mathbf{E}_x[f(x)H_S(x)]$.

Definition 4.16. The *Hermite decomposition* of $f : \mathbb{R}^n \to \mathbb{R}$ is $f(x) = \sum_{S \in \mathbb{N}^n} \hat{f}(S)H_S(x)$.

Definition 4.17. The *degree* of the coefficient $\hat{f}(S)$ is $|S| := \sum_{i=1}^n S_i$. and the *level-k Hermite weight* of f is $\sum_{S:|S|=k} \hat{f}(S)^2$.

The following basic lemma regarding the level-1 Hermite weight of functions will be of fundamental importance in the analysis in Chapter 12.

Lemma 4.18. *For any function $f : \mathbb{R}^n \to \mathbb{R}$, we have*

$$\sum_{i=1}^n \hat{f}(e^i)^2 = \mathop{\mathbf{E}}_{x,y}[f(x)f(y)\langle x, y\rangle]$$

where $\langle x, y \rangle = \sum_{i=1}^n x_i y_i$ is the standard vector dot product.

Proof. Applying the Hermite decomposition of f and linearity of expectation,

$$\mathop{\mathbf{E}}_{x,y}[f(x)f(y)\langle x, y\rangle] = \sum_{i=1}^n \sum_{S,T \in \mathbb{N}^n} \hat{f}(S)\hat{f}(T) \mathop{\mathbf{E}}_x[H_S(x)x_i] \mathop{\mathbf{E}}_y[H_T(y)y_i].$$

By definition, $x_i = h_1(x_i) = H_{e^i}(x)$. The orthonormality of the Hermite polynomials therefore guarantees that $\mathbf{E}_x[H_S(x)H_{e^i}(x)] = 1$ when $S = e^i$; otherwise it takes the value 0. Similarly, $\mathbf{E}_y[H_T(y)y_i] = 1$ iff $T = e^i$. □

Part I

Exact Query Complexity

Chapter 5

Testing Juntas

We begin by studying the problem of testing juntas—that is, of testing if a function has at most k relevant variables for some fixed k. The motivation for studying juntas comes from the fundamental role of these functions in different areas of computer science.

Motivation. In learning theory, juntas provide an elegant and useful framework for studying the problem of learning in the presence of irrelevant attributes [27, 30, 29]. More precisely, a target function is a k-junta when only k of the n possible attributes determine the value of the function on all inputs. The problem of *learning* juntas is a fundamental problem in learning theory [28, 79].

The important role of juntas in learning theory motivates the study of junta testing for two reasons. First, the insights on the structure of juntas learned in the course of research in testing juntas may lead to new learning algorithms as well. Second, junta testers may be used in learning theory directly in the *model selection* framework. The idea of this application is as follows: if we don't know ahead of time whether a target function is a k-junta or not, we can use a junta tester to quickly test the property and, if the target function is far from being a junta, we may save ourselves the query and computational cost that would have been wasted in trying to learn the target function under the erroneous assumption that it is a junta. (We discuss model selection in more detail in Chapter 12.)

Juntas also play an important role in complexity theory. The special case of testing *dictator* functions—1-juntas of the form $f(x) = x_i$ for some $i \in [n]$—in particular, has been at the heart of the development of probabilistically checkable proofs (PCPs) and in the corresponding advances in hardness of approximation [16, 17, 15, 65]. Juntas have also appeared directly in some constructions for hardness of approximation results—most notably for the vertex cover [47] and satisfiability of linear equations [75] problems.

Let us also mention briefly that the importance of juntas is not limited only to computer science: these functions are also fundamental objects of study in the analysis of boolean function [32, 54] and in social choice theory [69].

Lastly, juntas are play a central role in property testing itself. In fact, in a remarkable result, Diakonikolas et al. [46] showed that efficient junta testers, combined with the *testing by implicit learning* method, lead to efficient testers for a large number of properties of boolean functions such as low Fourier degree, computability by small decision trees, computability by small boolean circuits, representability by DNFs with a small number of terms, and sparse polynomials.

For many of the query complexity upper bounds obtained by Diakonikolas et al. [46], the barrier to obtaining improved query complexities is the query complexity of the junta test. This state of affairs suggested that improvements on the query complexity for testing juntas was likely to lead to improved query complexities for a number of other properties.

Previous work. The first result on testing dictator functions was obtained by Bellare, Goldreich, and Sudan [15] in the context of testing the "long code" for PCP constructions and further generalized by Parnas, Ron, and Samorodnitsky [84]. These results showed that it is possible to test dictator functions with $O(1/\epsilon)$ queries. This result also immediately implies that 1-juntas can be tested with the same number of queries. (See [21] for a more detailed discussion of testing 1-juntas and the other results mentioned in this section.)

The first general results on testing k-juntas for any $k \geq 1$ were obtained by Fischer et al. [52], who showed that the query complexity for testing k-juntas is $\mathrm{poly}(k/\epsilon)$. Most importantly, this showed that juntas can be tested with a number of queries that is *independent* of the size of the domain of the functions. In fact, they exhibited a number of different algorithms for testing k-juntas, with the most efficient one requiring $O(k^2 \log^2 k)$ queries.

Fischer et al. [52] also introduced the first lower bounds on the query complexity for testing juntas. They showed that for k small enough, any non-adaptive tester for k-juntas makes at least $\Omega(\sqrt{k}/\log(k))$ queries. This lower bound translates to a $\Omega(\log k)$ lower bound for general testers. That lower bound was subsequently improved to $\Omega(k)$ by Chockler and Gutfreund [41].

Result. The main result of this chapter is a new algorithm for testing juntas with a number of queries that nearly matches Chockler and Gutfreund's lower bound. Specifically, we establish the following result.

Theorem 5.1. *It is possible to ϵ-test the function $f : \{0,1\}^n \to \{0,1\}$ for the property of being a k-junta with $O\left(k/\epsilon + k \log k\right)$ queries.*

The algorithm that we introduce for testing juntas is conceptually quite simple. The main technical component of this chapter lies in the analysis of this algorithm.

5.1 The algorithm

The JuntaTest algorithm is based on two simple but powerful ideas. The first idea, initially presented by Fischer et al. [52], is that there is a very natural test for determining whether a set $S \subseteq [n]$ of coordinates contains a relevant variable in the function $f : \{0,1\}^n \to \{0,1\}$. This test, which we call the RelevantTest algorithm,[1] picks $x, y \in \{0,1\}^n$ independently and uniformly at random conditioned on $x_{\bar{S}} = y_{\bar{S}}$ and tests whether $f(x) \neq f(y)$.

```
RelevantTest($f$, $S$)
    1. Generate $x, z \in \{0,1\}^n$ independently and uniformly at random.
    2. Set $y = x_{\bar{S}} \vee z_S$.
    3. If $f(x) \neq f(y)$, accept and return $(x, y)$.
    4. Else, reject.
```

When none of the variables in $S \subseteq [n]$ are relevant in f, the RelevantTest always rejects. In fact, we can say more: the probability that the test accepts is equal to the influence of S in f. (See Lemma 5.2 below.) When the RelevantTest accepts, it returns a *witness* to the fact that S contains a relevant variable in the form of two inputs $x, y \in \{0,1\}^n$ for which $x_{\bar{S}} = y_{\bar{S}}$ and $f(x) \neq f(y)$.

The second idea we use is an observation of Blum, Hellerstein, and Littlestone [29] first made in the context of learning juntas: if we have two inputs $x, y \in \{0,1\}^n$ such that $f(x) \neq f(y)$, then we can perform a binary search over the hybrid vectors between x and y to find a coordinate that is relevant in f with $O(\log n)$ queries. We build on this

[1] In [52], the test is called the IndependenceTest and reverses the accept and reject actions.

observation by noting that if we have a partition of the coordinates into s parts and only care to identify a part that contains a relevant coordinate (rather than the coordinate itself), then we can optimize the binary search to only take $O(\log s)$ queries. We call the algorithm that implements this strategy FINDRELEVANTPART.

```
FINDRELEVANTPART($f$, $\{I_1, \ldots, I_s\}$, $x$, $y$)
    1. Initialize $\ell = 0$, $u = s$.
    2. While $u - \ell > 1$,
        2.1. Set $m = \ell + \lceil \frac{u-\ell}{2} \rceil$ and $S = \bigcup_{j=m+1}^{s} I_j$.
        2.2. Define $z = x_{\bar{S}} \vee y_S$.
        2.3. If $f(x) = f(z)$, then update $u = m$.
        2.4. Else, update $\ell = m$.
    3. Return $I_u$.
```

The JUNTATEST algorithm combines the two observations above in the obvious way. It maintains a set S of coordinates that may or may not be relevant to the function, then it runs the RELEVANTTEST a number of times to determine whether S contains a relevant variable. If so, then it obtains a pair $x, y \in \{0,1\}^n$ such that $f(x) \neq f(y)$ and $x_{\bar{S}} = y_{\bar{S}}$. By calling FINDRELEVANTPART with x and y, the algorithm identifies a part I that contains a relevant variable. It then removes the variables in I from S and repeats the process. If this algorithm identifies $k + 1$ different parts with relevant coordinates, it rejects the function; otherwise it accepts the function as a k-junta. The details of the algorithm are presented in Figure 5.1.

5.2 Analysis of the Algorithm

This section is dedicated to the analysis of the JUNTATEST algorithm. The first step in this analysis is to determine the probability that RELEVANTTEST accepts.

Lemma 5.2. *For any function $f : \{0,1\}^n \to \{0,1\}$ and any set $S \subseteq [n]$, a call to* RELEVANTTEST*(f, S) accepts with probability* $\mathrm{Inf}_f(S)$.

Proof. RELEVANTTEST(f, S) accepts iff $f(x) \neq f(x_{\bar{S}} \vee z_S)$, where x and z are picked independently and uniformly at random from $\{0,1\}^n$. By Definition 2.15, the probability that $f(x) \neq f(x_{\bar{S}} \vee z_S)$ is $\mathrm{Inf}_f(S)$. □

```
JUNTATEST($f$, $k$, $\epsilon$)
Additional parameters: $s = 24k^2,\ r = 12(k+1)/\epsilon$
  1. Randomly partition the coordinates in $[n]$ into $\mathcal{I} = \{I_1, \ldots, I_s\}$.
  2. Initialize $S \leftarrow [n]$, $\ell \leftarrow 0$.
  3. For each of $r$ rounds,
     3.1. If RELEVANTTEST($f$, $S$) accepts and returns $(x, y)$, then
          3.1.1. $I \leftarrow$ FINDRELEVANTPART($f, \mathcal{I}, x, y$).
          3.1.2. Update $S \leftarrow S \setminus I$ and $\ell \leftarrow \ell + 1$.
          3.1.3. If $\ell > k$, then reject the function.
  4. Accept the function.
```

Figure 5.1: The algorithm for ϵ-testing k-juntas.

This lemma will enable us to prove the correctness of the JUNTATEST algorithm if we can show that when f is far from being a k-junta, every set that includes all but at most k parts of the random partition $\mathcal{I}$ will have large influence. We formalize this statement and prove it in the next sub-section.

5.2.1 Main Technical Lemma

We begin with a definition that extends the notion of juntas with respect to partitions of the coordinates.

Definition 5.3 (Partition juntas)**.** Let $\mathcal{I}$ be a partition of $[n]$. The function $f : \{0,1\}^n \to \{0,1\}$ is a *k-part junta* with respect to $\mathcal{I}$ if the relevant coordinates in f are all contained in at most k parts of $\mathcal{I}$. Conversely, f is *ϵ-far from being a k-part junta* with respect to $\mathcal{I}$ if for every set J formed by taking the union of k parts in $\mathcal{I}$, $\mathrm{Inf}_f(\overline{J}) \geq \epsilon$.

When f is a k-junta, it is also a k-part junta with respect to any partition of $[n]$. The following lemma shows that when f is far from being a k-junta and $\mathcal{I}$ is a sufficiently fine random partition, then with large probability f is also far from being a k-part junta with respect to $\mathcal{I}$.

Lemma 5.4. *Let $\mathcal{I}$ be a random partition of $[n]$ with $s = 24k^2$ parts obtained by uniformly and independently assigning each coordinate to a part. With probability at least $\frac{5}{6}$, a function $f : \{0,1\}^n \to \{0,1\}$ that is ϵ-far from being a k-junta is also $\frac{\epsilon}{2}$-far from being a k-part junta with respect to $\mathcal{I}$.*

Proof. For $\tau > 0$, let $\mathcal{F}_\tau = \{J \subseteq [n] : \mathrm{Inf}_f(\overline{J}) < \tau\}$ be the family of all sets whose complements have influence at most τ. For any two sets $J, K \in \mathcal{F}_{\frac{\epsilon}{2}}$, the sub-additivity of influence implies that

$$\mathrm{Inf}_f(\overline{J \cap K}) = \mathrm{Inf}_f(\overline{J} \cup \overline{K}) \leq \mathrm{Inf}_f(\overline{J}) + \mathrm{Inf}_f(\overline{K}) < 2 \cdot \tfrac{\epsilon}{2} = \epsilon.$$

But f is ϵ-far from k-juntas, so by Lemma 2.21 every set $S \subseteq [n]$ of size $|S| \leq k$ satisfies $\mathrm{Inf}_f(\overline{S}) \geq \epsilon$. This implies that $|J \cap K| > k$ and, since this argument applies to every pair of sets in the family, that $\mathcal{F}_{\frac{\epsilon}{2}}$ is a $(k+1)$-intersecting family.

Let us now consider two separate cases: when $\mathcal{F}_{\frac{\epsilon}{2}}$ contains a set of size less than $2k$; and when it does not. In the first case, let $J \in \mathcal{F}_{\frac{\epsilon}{2}}$ be one of the sets of size $|J| < 2k$. By the union bound, the probability that J is completely separated by the partition $\mathcal{I}$ is at least $1 - 2k \cdot \binom{2k}{s} = \frac{5}{6}$. For every set $K \in \mathcal{F}_{\frac{\epsilon}{2}}$, we have $|J \cap K| \geq k + 1$. So when J is completely separated by $\mathcal{I}$, no set K in $\mathcal{F}_{\frac{\epsilon}{2}}$ is covered by the union of k parts in $\mathcal{I}$. Therefore, with probability at least $\frac{5}{6}$, the function f is $\frac{\epsilon}{2}$-far from k-part juntas with respect to $\mathcal{I}$, as we wanted to show.

Consider now the case where $\mathcal{F}_{\frac{\epsilon}{2}}$ contains only sets of size at least $2k$. Then we claim that $\mathcal{F}_{\frac{\epsilon}{4}}$ is a $2k$-intersecting family: otherwise, we could find sets $J, K \in \mathcal{F}_{\frac{\epsilon}{4}}$ such that $|J \cap K| < 2k$ and $\mathrm{Inf}_f(\overline{J \cap K}) \leq \mathrm{Inf}_f(\overline{J}) + \mathrm{Inf}_f(\overline{K}) < \frac{\epsilon}{2}$, contradicting our assumption.

Let $J \subseteq [n]$ be the union of k parts in $\mathcal{I}$. Since $\mathcal{I}$ is a random partition, J is a random subset obtained by including each element of $[n]$ in J independently with probability $p = \frac{k}{s} = \frac{1}{24k} < \frac{1}{2k+1}$. By Theorem 4.10,

$$\Pr_{\mathcal{I}}[\mathrm{Inf}_f(\overline{J}) < \tfrac{\epsilon}{4}] = \Pr[J \in \mathcal{F}_{\frac{\epsilon}{4}}] = \mu_{\frac{k}{s}}(\mathcal{F}_{\frac{\epsilon}{4}}) \leq \left(\tfrac{k}{s}\right)^{2k} .$$

Applying the union bound over the possible choices of J, we get that f is $\frac{\epsilon}{2}$-close to a k-part junta with respect to $\mathcal{I}$ with probability at most

$$\binom{s}{k}\left(\frac{k}{s}\right)^{2k} \leq \left(\frac{es}{k}\right)^{k}\left(\frac{k}{s}\right)^{2k} \leq \left(\frac{ek}{s}\right)^{k} < \frac{1}{6}. \qquad \square$$

5.2.2 Proof of Theorem 5.1

We are now ready to complete the analysis of the JUNTATEST algorithm.

Theorem 5.1 (Restated)**.** *It is possible to ϵ-test the function $f : \{0,1\}^n \to \{0,1\}$ for the property of being a k-junta with $O(k/\epsilon + k \log k)$ queries.*

Proof. We begin by determining the query complexity of the JUNTATEST algorithm. At most $2r = 24(k+1)/\epsilon$ queries are made in the execution of line 3.2 of the algorithm, and at most $(k+1)\log s = (k+1)\log(24k^2)$ queries are made in line 3.2.1 of the algorithm. So the algorithm makes a total of $O(k/\epsilon + k\log k)$ queries to the input function.

The completeness of the JUNTATEST algorithm is easy to establish: when the input function is a k-junta, it contains at most k parts with relevant coordinates, so the algorithm always accepts the function. Therefore, the JUNTATEST algorithm has one-sided error.

Finally, we analyze the soundness of the JUNTATEST algorithm. By Lemma 5.4, with probability at least $5/6$ a function f that is ϵ-far from being a k-junta is also $\epsilon/2$-far from being a k-part junta with respect to the random partition of the coordinates. When this is the case, the influence of S is at least $\epsilon/2$ until $k+1$ parts with relevant coordinates are identified. So the expected number of rounds required to identify $k+1$ parts with relevant variables is $2(k+1)/\epsilon$. By Markov's Inequality, the probability that the algorithm does not identify $k+1$ relevant parts in $12(k+1)/\epsilon$ rounds is at most $1/6$, and the overall probability that the JUNTATEST algorithm fails to reject f is at most $1/3$. □

5.3 Notes and Discussion

We conclude this chapter by discussing some implications of Theorem 5.1, the problem of testing non-boolean functions for the property of being a junta, and non-adaptive testing of juntas.

Implications. We mentioned in the introduction that one of the motivations for studying the junta testing problem is the *testing by implicit learning* method of Diakonikolas et al. [46], which uses junta testers to obtain efficient testing algorithms for a variety of other properties. Our hope was that a more query-efficient junta test would lead to improved upper bounds on the query complexity for a number of other properties of boolean functions.

Chakraborty, García Soriano, and Matsliah showed that the JUNTATEST algorithm presented in this section does indeed lead to improved upper bounds for the query complexity of other properties of boolean functions [39]. Their work introduces an efficient "sample extractor" for juntas that, combined with the JUNTATEST algorithm, improves the query complexity for testing computability by small DNFs, decision trees, boolean circuits, or branching programs, for testing low Fourier degree, and for testing sparse polynomials. We summarize their results in Table 5.1.

Property	**Upper bound**	**Previous upper bound**	**Lower bound**
s-term DNFs	$O(s \log s)$	$\tilde{O}(s^4)$ [46]	$\Omega(\log s)$ [39]
s-term monotone DNFs	$O(s \log s)$	$\tilde{O}(s^2)$ [84]	?
Size-s boolean formulae	$O(s \log s)$	$\tilde{O}(s^4)$ [46]	$s^{\Omega(1)}$ [39]
Size-s branching programs	$O(s \log s)$	$\tilde{O}(s^4)$ [46]	$\Omega(s)$ (§7)
Size-s decision trees	$O(s \log s)$	$\tilde{O}(s^4)$ [46]	$\Omega(\log s)$ (§7)
Size-s boolean circuits	$O(s^2)$	$\tilde{O}(s^6)$ [46]	$s^{\Omega(1)}$ [39]
s-sparse polynomials	$O(s \log s)$	$\tilde{O}(s^4)$ [46]	$\Omega(s)$ (§7)
Fourier degree $\leq d$	$O(2^{2d})$	$\tilde{O}(2^{6d})$ [46]	$\Omega(d)$ [40]

Table 5.1: Results obtained by Chakraborty, García Soriano, and Matsliah [39] by combining an efficient sample extractor with the JuntaTest algorithm.

Non-adaptive testing. The JuntaTest algorithm presented in this chapter is adaptive—indeed, adaptivity is the key component that enables the FindRelevantPart algorithm to require only $O(\log k)$ queries. A natural question to ask is whether we can also test juntas non-adaptively with $O(k \log k)$ queries. That question remains open. The best algorithm for testing juntas non-adaptively requires $O(k^{3/2})$ queries [19], and the best lower bound for non-adaptive testing of juntas is $\Omega(k \log k)$ queries.

Testing general functions. Let $X_1, \ldots, X_n, Y$ be arbitrary finite sets. The notion of juntas can be extended to functions $f : X_1 \times \cdots \times X_n \to Y$ over general product domains, and we may ask if such functions can also be tested efficiently for the property of being a k-junta.

Indeed, they can. The JuntaTest algorithm works essentially as-is in the more general setting and, with the appropriate generalizations of the notion of influence, the same analysis applies mostly as-is in the more general setting as well. For the details, we refer the reader to [20].

Chapter 6

Testing Partial Symmetry

The main result of the last chapter showed that juntas are efficiently testable. Looking back on this result, we may ask: can we explain *why* the k-junta property is so efficiently testable? In other words, what characteristics of the junta property did we use to obtain an efficient tester?

Intuitively, it seems that the main characteristic of juntas that lets us test the corresponding property efficiently is that all the irrelevant variables are "essentially the same". More precisely, juntas are invariant under any relabeling of the irrelevant variables. This characteristic is useful, notably, in arguing that a partition of $[n]$ is "good" when it completely separates the set of relevant variables.

It is not immediately clear, however, how accurate this intuition is. As a test for this intuition, we examine in this section the problem of testing *partially symmetric functions*. For a fixed $2 \leq t \leq n$, the function $f : \{0,1\}^n \to \{0,1\}$ is *t-symmetric* if there is a set S of $|S| = t$ coordinates for which f is invariant under all relabeling of the variables in S. In the informal terms of the last paragraph, the function f is t-symmetric if it has t variables that are "essentially the same". The set of $(n-k)$-symmetric functions includes all k-juntas as well as many other functions as well. (For example, the parity function $x_1 \oplus \cdots \oplus x_n$ is t-symmetric for every $2 \leq t \leq n$.)

We show that we can indeed test $(n-k)$-symmetry as efficiently as we can test k-juntas.

Theorem 6.1. *Fix $n > 0$ and $0 < k < \frac{n}{10}$. The property of being $(n-k)$-symmetric is ϵ-testable with $O(k \log k + k/\epsilon)$ queries.*

The algorithm used to test partially symmetric functions is similar to the junta testing algorithm, but the analysis is more involved. An important ingredient of the analysis is a new notion of influence, called "symmetric influence", that we introduce in Section 6.2.

6.1 The Algorithm

To create an algorithm for testing partial symmetry, we extend the JuntaTest algorithm from the last chapter. The first component of the JuntaTest was the RelevantTest algorithm that, given a function f and a set S of coordinates, aimed to determine if S contained a variable that was relevant in f. The analogous algorithm for our present purposes is the AsymmetricTest that aims to determine whether f is invariant under all relabeling of the variables in S or not.

```
AsymmetricTest($f$, $S$)
  1. Generate $x \in \{0,1\}^n$ uniformly at random.
  2. Generate the permutation $\pi$ on $[n]$ uniformly at random conditioned on $\pi(i) = i$ for each $i \in [n] \setminus S$.
  3. If $f(x) \neq f(\pi x)$, accept and return $(x, \pi x)$.
  4. Else, reject.
```

When f is invariant under all permutations of the labels of the variables in S, AsymmetricTest always rejects. Furthermore, when the test accepts, it also returns a witness (x, y) of the asymmetry of S in f. The probability that the test accepts when S is asymmetric, however, is not immediately clear. As we will see in Lemma 6.7, this probability is determined by the notion of symmetric influence that we define in the next section.

The second main component of the JuntaTest was the FindRelevantPart algorithm. Once again, we can define a similar algorithm for finding a part that is asymmetric in f. Recall that in the last chapter, the algorithm found a relevant part by performing a binary search over the hybrid strings between a pair of elements x, y that satisfy $f(x) \neq f(y)$. In the present situation, our task is complicated by the fact that we must perform a binary search over elements z with the same Hamming weight as x and y. We do so by treating one of the parts as a "workspace" that we use to control the Hamming weight of the hybrid strings. We then consider a binary search in which the hybrid strings we generate have Hamming weight close to that of x and y. The following BalancedSplit algorithm is a key component that lets us achieve this goal.

```
BalancedSplit($S$, $a_1, \dots, a_n$, $\tau_{\min}$, $\tau_{\max}$)
  1. Initialize $T \leftarrow \emptyset$.
  2. While $|T| < \lfloor \frac{|S|}{2} \rfloor$,
     2.1. Find $j \in S \setminus T$ that satisfies $\tau_{\min} \le a_j + \sum_{i \in T} a_i \le \tau_{\max}$.
     2.2. Update $T \leftarrow T \cup \{j\}$.
  3. Return $T$.
```

We are now ready to complete the definition of the FindAsymmetricPart algorithm. When (a) the part I_s passed into this function is larger than the other parts and contains no asymmetric variable and (b) x and y satisfy $\|x\| = \|y\|$ and $f(x) \neq f(y)$, this algorithm identifies a part that contains an asymmetric variable. (The analysis of this algorithm will be completed in Lemma 6.9 in Section 6.3.)

```
FindAsymmetricPart($f$, $x$, $y$, $I_1, \dots, I_s$)
  1. For $i = 0, 1, \dots, |I_s|$,
     1.1. Set $w^i$ to be a string satisfying $\|w^i_{I_s}\| = i$ and $\|w^i_{\overline{I_s}}\| = 0$.
  2. Fix $\tau_{\min} = -\|x_{I_s}\|$ and $\tau_{\max} = |I_s| - \|x_{I_s}\|$.
  3. For $j = 1, \dots, s-1$,
     3.1. Set $a_j = \|x_{I_j}\| - \|y_{I_j}\|$.
  4. Initialize $J_x \leftarrow \emptyset$, $J_y \leftarrow \emptyset$, and $J_? \leftarrow [s-1]$.
  5. While $|J_?| > 1$,
     5.1. Set $J' \leftarrow$ BalancedSplit$(J_?, a_1, \dots, a_{s-1}, \tau_{\min}, \tau_{\max})$.
     5.2. Set $S \leftarrow \bigcup_{j \in J_x \cup J'} I_j$ and $T \leftarrow \bigcup_{j \in J_y \cup (J_? \setminus J')} I_j$.
     5.3. Set $z \leftarrow x_S \vee y_T \vee w^{\|x\| - \|x_S \vee y_T\|}$.
     5.4. If $f(x) = f(z)$, then
          5.4.1. Update $J_x \leftarrow J_x \cup J'$ and $J_? \leftarrow J_? \setminus J'$.
     5.5. Else
          5.5.1. Update $J_y \leftarrow J_y \cup (J_? \setminus J')$ and $J_? \leftarrow J'$.
  6. Return $I_j$ for the element $j$ in the singleton set $J_? = \{j\}$.
```

Finally, we obtain the PartialSymmetryTest by combining the two components

```
PartialSymmetryTest($f$, $k$, $\epsilon$)
Additional parameters: $s = \Theta(k^2/\epsilon^2)$, $r = \Theta(k/\epsilon)$
    1. Randomly partition $[n]$ into $\{I_1, \ldots, I_{s-1}, I'_s, I'_{s+1}, I'_{s+2}, I'_{s+3}\}$.
    2. Set $I_s \leftarrow I'_s \cup I'_{s+1} \cup I'_{s+2} \cup I'_{s+3}$.
    3. If $|I_i| > \frac{1}{2}|I_s|$ for some $i \in [s-1]$, fail.
    4. Initialize $S \leftarrow [n] \setminus W$ and $\ell \leftarrow 0$.
    5. For each of $r$ rounds,
        5.1. If AsymmetricTest($f$, $S$) accepts and returns $(x, y)$, then
            5.1.1. $I \leftarrow$ FindAsymmetricPart($f$, $x$, $y$, $I_1, \ldots, I_s$).
            5.1.2. Update $S \leftarrow S \setminus I$ and $\ell \leftarrow \ell + 1$.
            5.1.3. If $\ell > k$, then reject the function.
    6. Accept the function.
```

Figure 6.1: The algorithm for ϵ-testing $(n-k)$-symmetric functions.

AsymmetricTest and FindAsymmetricPart in the natural way. That is, we generate random pairs of elements $x, y \in \{0,1\}^n$ with the same Hamming weight and use AsymmetricTest to determine if x and y form an asymmetry witness for f. If so, we use FindAsymmetricPart to identify one of the parts that contains an asymmetric variable. We reject the function if we identify more than k parts with asymmetric variables. The resulting algorithm is very similar to the JuntaTest. It is presented in Figure 6.1.

In the rest of the chapter, we analyze the PartialSymmetryTest algorithm to verify that it is indeed a valid ϵ-tester for $(n-k)$-symmetry. To complete this analysis, however, we must first introduce a new measure of influence, which we call "symmetric influence".

6.2 Symmetric influence

Recall that our definition of influence of a set of variables measures the sensitivity of a function to re-randomizing the values of the variables in that set. Similarly, we define the *symmetric influence* of a set of variables to measure the sensitivity of a function to *reordering* the values of the variables in that set—or, equivalently, to *relabeling* the variables in that set.

Definition 6.2 (Symmetric influence)**.** The *symmetric influence* of a set $J \subseteq [n]$ of variables in a Boolean function $f : \{0,1\}^n \to \{0,1\}$ is defined as

$$\mathrm{SymInf}_f(J) = \Pr_{x\in\{0,1\}^n, \pi\in\mathcal{S}_n}\left[f(x) \neq f(\pi x) \,\middle|\, \forall i \in \bar{J}, \pi(i) = i\right].$$

It follows from this definition that a function f is t-symmetric iff there exists a set J of size $|J| = t$ such that $\mathrm{SymInf}_f(J) = 0$. In fact, we can establish a much stronger statement.

Lemma 6.3. *Given a function $f : \{0,1\}^n \to \{0,1\}$ and a subset $J \subseteq [n]$, let f_J be the J-symmetric function closest to f. Then, the symmetric influence of J satisfies*

$$\mathrm{dist}(f, f_J) \leq \mathrm{SymInf}_f(J) \leq 2 \cdot \mathrm{dist}(f, f_J)\ .$$

Proof. For every weight $0 \leq w \leq n$ and $z \in \{0,1\}^{|\overline{J}|}$, define the layer $L^w_{\overline{J}\leftarrow z} := \{x \in \{0,1\}^n \mid \|x\| = w \wedge x_{\overline{J}} = z\}$ to be the vectors of Hamming weight w which identifies with z over the set $\overline{J}$ (notice that some of these layers may be empty). Let p^w_z be the fraction of the vectors in $L^w_{\overline{J}\leftarrow z}$ that one has to modify in order to make the restriction of f over $L^w_{\overline{J}\leftarrow z}$ to be constant (notice that $p^w_z \in [0, \frac{1}{2}]$ for every z, w).

With this notation, we can restate the definition of the symmetric influence of J as follows.

$$\begin{aligned}\mathrm{SymInf}_f(J) &= \sum_z \sum_w \Pr_{x\in\{0,1\}^n}[x \in L^w_{\overline{J}\leftarrow z}] \cdot \Pr_{x\in\{0,1\}^n, \pi\in\mathcal{S}_J}[f(x) \neq f(\pi x) \mid x \in L^w_{\overline{J}\leftarrow z}] \\ &= \frac{1}{2^n}\sum_z \sum_w \binom{|J|}{w-|z|} 2p^w_z(1-p^w_z)\ .\end{aligned}$$

This holds as in each such layer, the probability that x and πx would result in two different outcomes is the probability that the x would be chosen out of the smaller part and πx from the complement, or vise versa.

The function f_J can be obtained by modifying f at $p^z_{\overline{J}}$ fraction of the inputs in each layer $L^w_{\overline{J}\leftarrow z}$, as each layer can be addressed separately and we want to modify as few inputs as possible. By this observation, we have the following equality.

$$\mathrm{dist}(f, f_J) = \frac{1}{2^n}\sum_z \sum_w \binom{|J|}{w-|z|} p^w_z\ .$$

But since $1 - p^w_z \in [\frac{1}{2}, 1]$, we have that $p^w_z \leq 2p^w_z(1-p^w_z) \leq 2p^w_z$ and therefore $\mathrm{dist}(f, f_J) \leq \mathrm{SymInf}_f(J) \leq 2 \cdot \mathrm{dist}(f, f_J)$ as required. □

An immediate corollary of Lemma 6.3 gives a characterization of the symmetric influence of functions that are far from partially symmetric.

Corollary 6.4. *Let $f : \{0,1\}^n \to \{0,1\}$ be a function that is ϵ-far from being t-symmetric. Then for every set $J \subseteq [n]$ of size $|J| \geq t$, $\mathrm{SymInf}_f(J) \geq \epsilon$ holds.*

Proof. Fix $J \subseteq [n]$ of size $|J| \geq t$ and let g be a J-symmetric function closest to f. Since g is symmetric on any subset of J, it is in particular t-symmetric and therefore $\mathrm{dist}(f,g) \geq \epsilon$ as f is ϵ-far from being t-symmetric. Thus, by Lemma 6.3, $\mathrm{SymInf}_f(J) \geq \mathrm{dist}(f,g) \geq \epsilon$ holds. □

The analysis of the JUNTATEST relied crucially on the fact that the notion of influence is both monotone and sub-additive (Theorem 2.19). The following lemma shows that symmetric influence is also monotone.

Lemma 6.5 (Monotonicity)**.** *For any function $f : \{0,1\}^n \to \{0,1\}$ and any sets $J \subseteq K \subseteq [n]$,*

$$\mathrm{SymInf}_f(J) \leq \mathrm{SymInf}_f(K) .$$

Proof. Fix a function f and two sets $J, K \subseteq [n]$ so that $J \subseteq K$. We have seen before that the symmetric influence can be computed in layers, where each layer is determined by the Hamming weight and the elements outside the set we are considering. Using the fact that $\mathbf{Var}(X) = \Pr[X=0] \cdot \Pr[X=1]$, the symmetric influence is twice the expected variance over all the layers (considering also the size of the layers). Using the same notation as before,

$$\begin{aligned}\mathrm{SymInf}_f(J) &= \frac{1}{2^n} \sum_z \sum_w \binom{|J|}{w-|z|} 2 \cdot \mathop{\mathbf{Var}}_x[f(x) \mid x \in L^w_{\overline{J} \leftarrow z}] \\ &= 2 \cdot \mathop{\mathbf{E}}_y \left[\mathop{\mathbf{Var}}_x[f(x) \mid x \in L^{|y|}_{\overline{J} \leftarrow y_{\overline{J}}}] \right] .\end{aligned}$$

A key observation is that since $\overline{K} \subseteq \overline{J}$, the layers determined when considering J are a refinement of the layers determined when considering K. Together with the fact that $\mathbf{Var}(X) = \Pr[X=0] \cdot \Pr[X=1]$ is a concave function in the range $[0,1]$, we can apply Jensen's inequality on each layer before and after the refinement to get the desired inequality. More precisely, for every $z \in \{0,1\}^{|\overline{K}|}$ and $0 \leq w \leq n$,

$$\mathop{\mathbf{Var}}_x[f(x) \mid x \in L^w_{\overline{K} \leftarrow z}] \geq \mathop{\mathbf{E}}_y \left[\mathop{\mathbf{Var}}_x[f(x) \mid x \in L^w_{\overline{J} \leftarrow y_{\overline{J}}}] \mid y \in L^w_{\overline{K} \leftarrow z} \right] .$$

Averaging this over all layers, we get the desired result. □

Finally, to complete the analogy with influence, we would like to show that symmetric influence is also sub-additive. Unfortunately, that is not the case. Consider for example the function $f : \{0,1\}^6 \to \{0,1\}$ defined by

$$f(x) = (x_1 \vee x_2 \vee x_3) \oplus (x_4 \vee x_5 \vee x_6).$$

This function f is invariant under relabelings of the variables in $\{1,2,3\}$ or in $\{4,5,6\}$ so $\mathrm{SymInf}_f(\{1,2,3\}) = \mathrm{SymInf}_f(\{4,5,6\}) = 0$. But $\mathrm{SymInf}_f(\{1,\ldots,6\}) > 0$, as shown for instance by the fact that $f(1,1,1,0,0,0) \neq f(1,1,0,1,0,0)$. Therefore, for this function, we have

$$\mathrm{SymInf}_f(\{1,2,3\} \cup \{4,5,6\}) > 0 = \mathrm{SymInf}_f(\{1,2,3\}) + \mathrm{SymInf}_f(\{4,5,6\}).$$

More generally, we can consider the function $f(x) = f_1(x_J) \oplus f_2(x_K)$ for some partition $[n] = J \cup K$ and two randomly chosen symmetric functions f_1 and f_2. With high probability, f will be very far from symmetric and we will have $\mathrm{SymInf}_f([n]) = \mathrm{SymInf}_f(J \cup K) \approx \frac{1}{2}$ while $\mathrm{SymInf}_f(J) = \mathrm{SymInf}_f(K) = 0$. This example shows that symmetric influence can be far from sub-additive in general. We can show, however, that the symmetric influence of sets of variables is *approximately* sub-additive when the sets are small enough.

Lemma 6.6 (Weak sub-additivity). *There is a universal constant c such that, for any constant $0 < \gamma < 1$, a function $f : \{0,1\}^n \to \{0,1\}$, and sets $J, K \subseteq [n]$ of size at least $(1-\gamma)n$,*

$$\mathrm{SymInf}_f(J \cup K) \leq \mathrm{SymInf}_f(J) + \mathrm{SymInf}_f(K) + c\sqrt{\gamma}\,.$$

Proof. We will choose $c \geq \sqrt{2}$. Then, the RHS becomes more than one when $\gamma > \frac{1}{2}$ and the inequality trivially holds. Thus, we assume $\gamma \leq \frac{1}{2}$ in what follows. We use the same notions as in Lemma 4.5. From Lemma 4.5,

$$\begin{aligned}
\mathrm{SymInf}(J \cup K) &= \Pr_{x,\pi}[f(x) \neq f(\pi x)] \\
&= \mathop{\mathbf{E}}_{x}\big[\Pr_{\pi}[f(x) \neq f(\pi x)]\big] \\
&\leq \mathop{\mathbf{E}}_{x}\left[\Pr_{\pi_J,\pi_K}[f(x) \neq f(\pi_J \pi_K x)] + \mathrm{d_{TV}}(\mathcal{D}_{\pi x}, \mathcal{D}_{\pi_J \pi_K x})\right] \\
&= \mathop{\mathbf{E}}_{x}\left[\Pr_{\pi_J,\pi_K}[f(x) \neq f(\pi_J \pi_K x)] + \mathrm{d_{TV}}(\mathcal{H}_{|J \cup K|,|x_{J \cup K}|,|K \setminus J|}, \mathcal{H}_{|K|,|x_K|,|K \setminus J|})\right]
\end{aligned}$$

Let $t = \frac{1}{100\sqrt{\gamma}}$. We call x *strongly balanced* if $\left| \|x_{J \cup K}\| - \frac{|J \cup K|}{2} \right| \leq t\sqrt{n'}$ and $\left| \|x_K\| - \frac{|K|}{2} \right| \leq t\sqrt{|K|}$. From Chernoff bound and the union bound, x is strongly balanced except with probability at most $4\exp(-2t^2) \leq 4\exp\left(-\frac{1}{5000\gamma}\right) \leq c'\gamma$ for some constant c'.

Note that $|K \setminus J| \leq |\overline{J}| \leq \gamma n$ and $|J \cup K| - |K| = |J \setminus K| \leq |\overline{K}| \leq \gamma n$. Also, $t = \frac{1}{100\sqrt{\gamma}} \leq \frac{1}{100\gamma}$ holds. Thus, when x is strongly balanced, Lemma 4.2 implies that $\mathrm{d}_{\mathrm{TV}}(\mathcal{H}_{|J \cup K|, |x_{J \cup K}|, |K \setminus J|}, \mathcal{H}_{|K|, |x_K|, |K \setminus J|}) \leq c_{4.2}(1+t)\gamma$. Then, we have

$$\begin{aligned}
\mathrm{SymInf}(J \cup K) &\leq \mathop{\mathbf{E}}_{x}\Big[\Pr_{\pi_J, \pi_K}[f(x) \neq f(\pi_J \pi_K x)]\Big] + \Pr_{x}[x \text{ is not strongly balanced}] \\
&\quad + \mathop{\mathbf{E}}_{x}\Big[\mathbf{1}[x \text{ is strongly balanced}] \cdot c_{4.2}(1+t)\gamma\Big] \\
&\leq \Pr_{x, \pi_J, \pi_K}[f(x) \neq f(\pi_K x)] + \Pr_{x, \pi_J, \pi_K}[f(\pi_K x) \neq f(\pi_J \pi_K x)] \\
&\quad + c'\gamma + c_{4.2}(1+t)\gamma \\
&\leq \mathrm{SymInf}_f(J) + \mathrm{SymInf}_f(K) + c\sqrt{\gamma}
\end{aligned}$$

for some constant c. □

6.3 Analysis of the algorithm

This section is devoted to the analysis of the PARTIALSYMMETRYTEST algorithm. The broad outline of this analysis closely mirrors that of the analysis of the JUNTATEST algorithm in the last chapter.

6.3.1 Analysis of ASYMMETRICTEST

As a first step in the analysis of the partial symmetry tester, we show that the ASYMMETRICTEST accepts a set with probability equal to the symmetric influence of that set. The proof of this statement follows almost immediately from the definition of symmetric influence.

Lemma 6.7. *For any function $f : \{0,1\}^n \to \{0,1\}$ and any set $S \subseteq [n]$, a call to* ASYMMETRICTEST*(f, S) accepts with probability* $\mathrm{SymInf}_f(\overline{S})$.

Proof. The ASYMMETRICTEST algorithm accepts iff $f(x) \neq f(\pi x)$. When x is chosen uniformly at random from $\{0,1\}^n$ and π is a permutation on $[n]$ chosen uniformly at random conditioned on $\pi(i) = i$ for each $i \in \overline{S}$, the probability that $f(x)$ and $f(\pi x)$ are distinct is, by definition, $\mathrm{SymInf}_f(\overline{S})$. □

6.3.2 Analysis of FINDASYMMETRICPART

To establish the correctness of FINDASYMMETRICPART, we begin by establishing sufficient conditions for guaranteeing that the BALANCEDSPLIT algorithm will succeed in finding a balanced split of the input set.

Lemma 6.8. *Fix a set $S \subseteq [n]$ and constants $\tau_{\min}, \tau_{\max}$ that satisfy $\tau_{\min} \le 0 \le \tau_{\max}$. Let $a_1, \ldots, a_n$ satisfy $\sum_{i=1}^n a_i = 0$, $\tau_{\min} \le \sum_{i \in S} a_i \le \tau_{\max}$, and $\max_{i \in [n]} |a_i| < \frac{\tau_{\max} - \tau_{\min}}{2}$. Then* BALANCEDSPLIT *returns a set T of size $|T| = \lfloor \frac{|S|}{2} \rfloor$ that satisfies*

$$\tau_{\min} \le \sum_{i \in T} a_i \le \tau_{\max}.$$

Proof. To establish the lemma, it suffices to show that for any strict subset $T \subset S$ that satisfies $\tau_{\min} \le \sum_{i \in T} a_i \le \tau_{\max}$, we can always find an elements $j \in S \setminus T$ such that $\tau_{\min} \le a_j + \sum_{i \in T} a_i \le \tau_{\max}$.

Consider the case when $\sum_{i \in T} a_i \le \frac{\tau_{\min} + \tau_{\max}}{2}$. If there exists $j \in S \setminus T$ such that $a_j \ge 0$, then

$$\tau_{\min} \le \sum_{i \in T} a_i \le a_j + \sum_{i \in T} a_i < \frac{\tau_{\max} - \tau_{\min}}{2} + \frac{\tau_{\max} + \tau_{\min}}{2} = \tau_{\max}.$$

Conversely, if every $j \in S \setminus T$ satisfies $a_j \le 0$, for any such j we have

$$\tau_{\max} \ge \sum_{i \in T} a_i \ge a_j + \sum_{i \in T} a_i \ge \sum_{i \in S} a_i \ge \tau_{\min}.$$

In either case, we obtain the desired inequality.

The case when $\sum_{i \in T} a_i > \tau_{\text{mid}}$ is nearly identical. □

We can now complete the proof of correctness of FINDASYMMETRICPART.

Lemma 6.9. *Fix $f : \{0,1\}^n \to \{0,1\}$. Let $\mathcal{I} = \{I_1, \ldots, I_s\}$ be a partition of $[n]$ into s parts such that $|I_s| > 2 \max_{1 \le i \le s-1} |I_i|$ and I_s contains no asymmetric variables. Let $x, y \in \{0,1\}^n$ satisfy $|x| = |y|$, $x_{I_s} = y_{I_s}$, and $f(x) \ne f(y)$. Then a call to* FINDASYMMETRICPART*$(f, x, y, I_1, \ldots, I_s)$ returns an asymmetric part in $\mathcal{I}$. Furthermore, this call requires $O(\log s)$ queries to the function f.*

Proof. We claim that the sets J_x, J_y, and $J_?$ satisfy the following three properties at every invocation of the loop in step 5.

(i) J_x, J_y, and $J_?$ form a partition of $[s-1]$.

(ii) $f(x) = f(x_{J_x \cup J_?} \vee y_{J_y})$.

(iii) $f(y) = f(x_{J_x} \vee y_{J_y \cup J_?})$.

All three claims can be verified by direct examination of the algorithm. Claim (i) is clearly satisfied initially by the definition in Step 4 and is also clearly maintained by the updates in Steps 5.4.1 and 5.5.1 since every elements added to J_x or to J_y are simultaneously removed from $J_?$. Claim (ii) is (vacuously) true initially and is guaranteed to remain true since we only add elements to J_x when the condition in Step 5.4 holds. The proof of Claim (iii) is identical but for the observation that we add elements to J_y when the condition in Step 5.4 does *not* hold or, equivalently, when $f(y) = f(z)$.

The invariants in Claims (i)–(iii) guarantee that at every iteration of the loop, we have

$$f(x_{J_x} \vee x_{J_?} \vee y_{J_y} \vee w^{\|x\| - \|x_{J_x} \vee x_{J_?} \vee y_{J_y}\|}) \neq f(x_{J_x} \vee y_{J_?} \vee y_{J_y} \vee w^{\|x\| - \|x_{J_x} \vee y_{J_?} \vee y_{J_y}\|}).$$

When $J_? = \{j\}$ is a singleton, this inequality and the condition that I_s contains no asymmetric variable guarantees that I_j is an asymmetric part. Furthermore, the conditions of the lemma also guarantee that the conditions of Lemma 6.8 hold. This guarantees that every iteration of the loop cuts the size of $J_?$ in half, so that the algorithm is guaranteed to terminate in $\log s$ steps. Since the only queries to f occur in Step 5.4, this concludes the query complexity analysis of the algorithm. □

6.3.3 Main technical lemma

The key claim in the analysis of PARTIALSYMMETRYTEST is that if a function is far from being $(n-k)$-symmetric, then it is also far from being symmetric on any union of all but k parts of a sufficiently fine random partition. (C.f. Lemma 5.4.) We now prove this claim.

Lemma 6.10. *Let $f : \{0,1\}^n \to \{0,1\}$ be a function ϵ-far from $(n-k)$-symmetric and $\mathcal{I}$ be a random partition of $[n]$ into $r = c \cdot k^2/\epsilon^2$ parts, for some large enough constant c. Then with probability at least $\frac{8}{9}$, $\mathrm{SymInf}_f(\overline{J}) \geq \frac{\epsilon}{9}$ holds for any union J of k parts.*

Proof. We first note that when the number of parts r is bigger then n, we simply partition into the n single-element sets and the lemma trivially holds. For $0 \leq t \leq 1$, let $\mathcal{F}_t = \{J \subseteq [n] : \mathrm{SymInf}_f(\overline{J}) < t,\ |J| \leq 5kn/r\}$ be the family of all sets which are not too big and whose complement has symmetric influence of at most t. (Notice that with high probability, the union of any k sets in the partition would have size smaller than $5kn/r$, and therefore we assume this is the case from this point on.) Our first observation is that for small enough values of t, $\mathcal{F}_t$ is a $(k+1)$-intersecting family. Indeed, for any sets $J, K \in \mathcal{F}_{\frac{\epsilon}{3}}$,

$$\begin{aligned}\mathrm{SymInf}_f(\overline{J \cap K}) &= \mathrm{SymInf}_f(\overline{J} \cup \overline{K}) \\ &\leq \mathrm{SymInf}_f(\overline{J}) + \mathrm{SymInf}_f(\overline{K}) + c\sqrt{5k/r} < 2\tfrac{\epsilon}{3} + \tfrac{\epsilon}{9} < \epsilon.\end{aligned}$$

Since f is ϵ-far from $(n-k)$-symmetric, every set $S \subseteq [n]$ of size $|S| \leq k$ satisfies $\mathrm{SymInf}_f(\overline{S}) \geq \epsilon$. So $|J \cap K| > k$.

We consider two cases separately: when $\mathcal{F}_{\frac{\epsilon}{3}}$ contains a set of size less than $2k$; and when it does not. The first case is identical to the proof of Lemma 5.4 and hence we do not elaborate on it.

In the second case, which also resembles the proof of Lemma 5.4, we claim that $\mathcal{F}_{\frac{\epsilon}{9}}$ is a $2k$-intersecting family. If this was not the case, we could find sets $J, K \in \mathcal{F}_{\frac{\epsilon}{9}}$ such that $|J \cap K| < 2k$ and $\mathrm{SymInf}_f(\overline{J \cap K}) \leq \mathrm{SymInf}_f(\overline{J}) + \mathrm{SymInf}_f(\overline{K}) + \frac{\epsilon}{9} < \frac{\epsilon}{3}$, contradicting our assumption.

Let $J \subseteq [n]$ be the union of k parts in $\mathcal{I}$. Since $\mathcal{I}$ is a random partition, J is a random subset obtained by including each element of $[n]$ in J independently with probability $p = \frac{k}{r} < \frac{1}{2k+1}$. To bound the probability that J contains some element from $\mathcal{F}_{\frac{\epsilon}{9}}$, we define $\mathcal{F}'_{\frac{\epsilon}{9}}$ to be all the sets that contain a member from $\mathcal{F}_{\frac{\epsilon}{9}}$. Since $\mathcal{F}'_{\frac{\epsilon}{9}}$ is also a $2k$-intersecting family, by Theorem 4.10, for every such J of size at most $5kn/r$, $\Pr[\mathrm{SymInf}_f(\overline{J}) < \frac{\epsilon}{9}] = \Pr[J \in \mathcal{F}_{\frac{\epsilon}{9}}] \leq \mu_{k/r}(\mathcal{F}'_{\frac{\epsilon}{9}}) \leq \left(\frac{k}{r}\right)^{2k}$. Applying the union bound over all possible choices for k parts, f will not satisfy the condition of the lemma with probability at most $\binom{r}{k}\left(\frac{k}{r}\right)^{2k} = O(k^{-k})$. □

6.3.4 Proof of Theorem 6.1

We can now complete the proof that partial symmetry is efficiently testable.

Theorem 6.1 (Restated). *Fix $n > 0$ and $0 \leq k < n$. The property of being $(n-k)$-symmetric is ϵ-testable with $O(k \log k + k/\epsilon)$ queries.*

Proof. We consider different cases depending on the value of k. When $k = 0$, the problem of testing $(n-k)$-symmetry is simply the problem of testing symmetry. This can be done with $O(1/\epsilon)$ queries by calling ASYMMETRYTEST(f, $[n]$) a number of times and accepting iff every call to the function rejects.

Consider now the case where $1 \leq k = o(\epsilon\sqrt{\frac{n}{\log n}})$. We show that in this case the PARTIALSYMMETRYTEST is a valid tester for partial symmetry that satisfies the requirements of the theorem statement. Note first that $s = \Theta(k^2/\epsilon^2) < \frac{n}{12\log n} - 3$.

We claim that the failure event in Step 3 of PARTIALSYMMETRYTEST occurs with probability at most $\frac{1}{12}$. For any $i = 1, \ldots, s-1$, the expected size of I_i is $\mathbf{E}\,|I_i| = \frac{n}{s+3}$ so, by Chernoff's bound,

$$\Pr\left[|I_i| > \tfrac{3}{2}\tfrac{n}{s+3}\right] < e^{-\frac{n}{12(s+3)}}.$$

Similarly, $\mathbf{E}\,|I_s| = \frac{4n}{s+3}$ and Chernoff's bound implies that

$$\Pr\left[|I_s| < 3\tfrac{n}{s+3}\right] < e^{-\frac{n}{8(s+3)}}.$$

The inequality $|I_i| > \frac{1}{2}|I_s|$ can only hold when $|I_i| > \frac{3}{2}\frac{n}{s+3}$ for some $i \in [s-1]$ or when $|I_s| < 3\frac{n}{s+3}$. By the union bound and our bound on s, the probability that this event occurs is bounded above by

$$(s-1)e^{-\frac{n}{12(s+3)}} + e^{-\frac{n}{8(s+3)}} < se^{-\frac{n}{12(s+3)}} \leq \frac{n}{12\log n}e^{-\log n} = \frac{1}{12\log n} \ll \frac{1}{12}$$

for any $n \geq 2$.

When f is $(n-k)$-symmetric, the probability that I_s contains an asymmetric variable is at most $k\frac{4}{s+3} = \Theta(\epsilon^2/k) \ll \frac{1}{12}$. When I_s contains no asymmetric variable, then Lemma 6.9 guarantees that every part returned by FINDASYMMETRICPART contains an asymmetric variable. There are at most k such parts, so this means that, conditioned on I_s containing no asymmetric variable and the condition in Step 3 satisfied, f will always be accepted. In other words, every $(n-k)$-symmetric function is accepted with probability at least $1 - 2\frac{1}{12} = \frac{5}{6} > \frac{2}{3}$.

When f is ϵ-far from $(n-k)$-symmetric, Lemma 6.10 guarantees that with probability at least $\frac{8}{9}$ over the choice of the random partition, $\mathrm{SymInf}_f(\bar{J}) \geq \frac{\epsilon}{9}$ will hold for any set J obtained by taking the union of at most k parts. Then Lemma 6.7 and Chernoff's bounds proves that with probability at least $\frac{8}{9}$, the algorithm will identify at least $k+1$ parts with asymmetric variables. Therefore, the algorithm correctly rejects with probability at least $1 - \frac{1}{12} - \frac{2}{9} = 1 - \frac{11}{36} > \frac{2}{3}$.

The above argument shows that PARTIALSYMMETRYTEST is indeed a valid ϵ-tester for $(n-k)$-symmetry, as claimed. To complete the proof of the theorem for this range of values of k, it suffices to analyze the algorithm's query complexity. By inspection, we see immediately that it makes at most $O(r + k\log s) = O(k/\epsilon + k\log(k^2/\epsilon^2)) = O(k/\epsilon + k\log k)$, as required by the theorem statement.

Finally, we now consider the case where $\Omega(\epsilon\sqrt{\frac{n}{\log n}}) \leq k < n$. In this case, consider a simplification of the PARTIALSYMMETRYTEST in which we replace the random partition with a trivial partition into $n-3$ parts with 4 coordinates from $[n]$ chosen at random and put into the part I_{n-3} and all the other remaining parts containing a single element from $[n]$. We can then apply the same strategy as in the general algorithm where we run the ASYMMETRICTEST to find witnesses of asymmetry then run FINDASYMMETRICPART to identify the parts that contain asymmetric variables.

The analysis of correctness of the simplified algorithm is essentially identical to the one above. □

6.4 Notes and Discussion

History of partially symmetric functions. The class of partially symmetric functions was first considered by Shannon [92] in his research on the circuit complexity of boolean functions. He showed that while most functions $f : \{0,1\}^n \to \{0,1\}$ have circuit complexity exponential in n, the circuit complexity of partially symmetric functions is much smaller.

Shannon's result generated much further research into the interplay between the partial symmetry of functions and circuit complexity [36, 10] and the problem of determining when a function is partially symmetric (see, e.g., [43] and references therein).

Other definitions of symmetry have been considered in the computational complexity community. Clote and Kranakis [42] showed that all functions that obey a large amount of symmetry in the sense that the isomorphism class of those functions contain at most $\text{poly}(n)$ distinct functions are in $\mathbf{NC_1}$. This result was further generalized by Babai, Beals, and Takácsi-Nagy [12] and has been used, e.g., in proof complexity [85].

Functions that are partially symmetric under our definition also satisfy the Clote–Kranakis definition of symmetry. While this does not immediately imply testability of Clote–Kranakakis symmetry (since, obviously, an efficient test for a property $\mathcal{P}$ does not imply the existence of efficient testers for properties that contain $\mathcal{P}$), Chakraborty, Fischer, García Soriano, and Matsliah recently showed that the two definitions are in fact equivalent [37]. Thus, the theorem in this chapter shows that Clote–Kranakakis symmetry is also efficiently testable.

Concurrent work. Chakraborty, Fischer, García Soriano, and Matsliah [37], in independent and simultaneous research, obtained a different proof that partial symmetry can be tested efficiently. Interestingly, their result is obtained by a significantly different approach: instead of generalizing the JUNTATEST algorithm, they identify a clever reduction between testing partial symmetry and testing juntas to show that the JUNTATEST algorithm can be used as-is—along with a separate algorithm to construct a function g that is a junta iff f is partially symmetric—to also test partial symmetry. We refer the reader to their paper [37] for the details.

Influence on Schreier graphs. O'Donnell and Wimmer [82, 83] introduced a generalized notion of influence for functions defined over Schreier graphs. In this paragraph, we discuss a connection between this generalized notion of influence and symmetric influence. This connection was first pointed out to us by Dvir Falik.

For any set $U \subseteq \mathcal{S}_n$ of permutations that is (a) closed under inverses, and (b) generates $\mathcal{S}_n$, we can define the *Schreier graph* $\mathrm{Sch}(\{0,1\}^n, \mathcal{S}_n, U)$ to be the graph obtained by associating one vertex for each element of $\{0,1\}^n$ and adding an edge between $x, y \in \{0,1\}^n$ if there exists a permutation $\pi \in U$ such that $y = \pi x$. The *Schreier influence* of the permutation $\pi \in U$ in the function $f : \{0,1\}^n \to \{0,1\}$ is

$$\mathrm{Inf}_f^{(U)}(\pi) = \tfrac{1}{2} \Pr_x[f(x) \neq f(\pi x)].$$

Fix $U = \{\tau_{i,j}\}_{i \neq j}$ to be the set of transpositions on $[n]$. Then for any pair of distinct elements $i, j \in [n]$, we have

$$\begin{aligned}\mathrm{Inf}_f^{(U)}(\tau_{i,j}) &= \tfrac{1}{2} \Pr_x[f(x) \neq f(\tau_{i,j} x)] \\ &= \Pr_{x,\pi}\big[f(x) \neq f(\pi x) \mid \forall \ell \in [n] \setminus \{i,j\}, \pi(\ell) = \ell\big] = \mathrm{SymInf}_f(\{i,j\}).\end{aligned}$$

Thus, for sets of size 2, symmetric influence is a special case of Schreier influence.

More generally, the symmetric influence of larger sets of variables corresponds to the *average* Schreier influence of sets of permutations on the same function. Specifically, letting U be the set of permutations with at least $|\bar{J}|$ fixed points, we have that

$$\mathrm{SymInf}_f(J) = \mathop{\mathbf{E}}_{\pi : \forall \ell \in \bar{J}, \pi(\ell) = \ell}[\mathrm{Inf}_f^{(U)}(\pi)].$$

Invariance in property testing. Finally, we wish to mention that the research presented in this chapter fits into the general effort for understanding the role of invariance in property testing. This effort, launched by Kaufman and Sudan [72], has been widely successful in identifying the effect of different invariances of algebraic properties of functions on the number of queries required to test the same properties. We encourage the reader to consult the survey [95] and the references therein for more details on this line of research.

Chapter 7

Testing k-Linearity

Linearity testing is one of the earliest success stories in property testing. Recall that the function $f : \{0,1\}^n \to \{0,1\}$ is *linear* if it returns the parity of a subset of the variables. As we saw in Section 2.6, when f is a linear function, it satisfies the identity

$$f(x) \oplus f(y) = f(x \oplus y)$$

for every $x, y \in \{0,1\}^n$. Blum, Luby, and Rubinfeld [31] showed that, remarkably, linearity can be ϵ-tested with only $O(1/\epsilon)$ queries by simply verifying that the above identity holds for randomly selected pairs x, y. Linearity testing has since been studied extensively [16, 17, 14, 71] and its query complexity is well understood.

The class of k-*linear* functions—functions that return the parity of *exactly* k of the input variables—is closely related to the class of all linear functions. The query complexity of the k-linearity testing task, however, remained until very recently much less well-understood than that of linearity testing.

Previous work. Let us first review some folklore bounds on the query complexity for testing k-linearity. As we saw in Section 3.2, any proper learning algorithm for k-linear functions with query complexity q yields a k-linearity tester that makes $q + O(1/\epsilon)$ queries. There is a very simple n-query learning algorithm for k-linear functions: query the values $f(e^1), f(e^2), \ldots, f(e^n)$. When f is k-linear, $f(e^i) = 1$ for exactly k indices $i \in [n]$ and the function returns the parity of those variables. Thus, by Lemma 3.13, we can test k-linearity with $n + O(1/\epsilon)$ queries for every $0 \le k \le n$.

When $k < \frac{n}{\log n}$, a similar folkloric argument yields an even better bound on the query complexity for testing k-linearity. There are $\binom{n}{k}$ distinct k-linear functions, so by Corollary 3.14 we can test k-linearity with $O(\log \binom{n}{k}/\epsilon) = O(k \log(n)/\epsilon)$ queries.

There are a few values of k for which we have significantly better bounds. First, when $k = 0$ and $k = n$, then the function is either constant or the symmetric parity function; in both cases, testing k-linearity reduces to testing function identity and can be done with $O(1/\epsilon)$ queries. When $k = 1$, we have an interesting special case: the class of 1-linear functions is exactly the set of dictator functions. The dictator tests that require $O(1/\epsilon)$ queries [15, 84] imply that 1-linearity also can be tested with the same query complexity.

The first improvements on the folklore bound for the query complexity for testing k-linearity for general values of k were obtained by Fischer et al. [52]. One of their results on testing function isomorphism that we will discuss in more detail in Chapter 8 implies that k-linearity can be tested with $\mathrm{poly}(k, \epsilon)$ queries. Clearly, when $k \ll n$, this is much more efficient than the folklore test.

The first non-trivial lower bound on the query complexity for testing k-linearity was also established by Fischer et al. [52]. They showed that when $k = o(\sqrt{n})$, testing k-linearity non-adaptively requires $\Omega(\sqrt{k/\log k})$ queries. This implies a general lower bound of $\Omega(\log k)$ queries for general (i.e., adaptive) k-linearity testers. Their motivation for establishing this bound was not directly related to the study of k-linearity. Rather, they wanted to establish a lower bound for the query complexity of the junta testing problem. Since k-linear functions are k-juntas and $(k+2)$-linear functions are far from k-juntas, so to prove a lower bound on the query complexity for testing k-juntas, it suffices to establish a corresponding lower bound for the problem of distinguishing k-linear and $(k+2)$-linear functions.

In Chapter 10, we will derive a lower bound that is incomparable to the lower bound of Fischer et al. [52]: Theorem 10.1 gives a weaker bound of $\Omega(\log\log(\min\{k, n-k\}))$ queries for testing k-linearity, but that lower bound applies to all values of k. We will discuss the theorem in more detail in that chapter. For now, let us simply point out that the original motivation for this result was again not the study of k-linearity per se, but rather to understand another property testing problem—in this case, the function isomorphism testing problem.

A significant improvement on the best lower bound for the query complexity of testing k-linearity was obtained by Goldreich [57]. He showed that $\Omega(\min\{k, n-k\})$ queries are required to test k-linearity non-adaptively and that adaptive testers for k-linearity must make at least $\Omega(\min\{\sqrt{k}, \sqrt{n-k}\})$ queries.[1] Once again, this result was not motivated by

[1] More precisely, Goldreich considered the slightly different problem of testing $\leq k$-linearity—testing whether a function returns the parity of *at most* k variables. His arguments, however, apply equally well to the k-linearity testing problem.

the study of k-linearity itself. Instead, the lower bound was used to establish lower bounds on the number of queries required to test properties computable by width-2 ordered binary decision diagrams (OBDDs).[2]

The folklore tester described earlier in this section shows that Goldreich's lower bound for the query complexity of non-adaptive k-linearity testers is asymptotically optimal when $k \approx \frac{n}{2}$. The major question that was left open was whether the lower bound for adaptive testers of k-linearity could also be improved to match the trivial query complexity of the folklore tester in the same range. Indeed, Goldreich conjectured that it could—specifically, that testing $\frac{n}{2}$-linearity requires $\Omega(n)$ queries.

Our results. We confirm Goldreich's conjecture and show that testing k-linearity requires $\Omega(\min\{k, n-k\})$ queries. In fact, we show more. Instead of simply giving a lower bound that is asymptotically equivalent to $\min\{k, n-k\}$, we show a lower bound that nearly matches this amount.

Theorem 7.1. *For any* $0 < k < n$ *and any* $0 < \epsilon < \frac{1}{2}$, *$\epsilon$-testing k-linearity requires at least* $\min\{k, n-k\} \cdot (1 - o(1))$ *queries.*

In particular, for $k = \frac{n}{2}$, the theorem states that no tester for k-linearity can improve on the query complexity of the folklore testing algorithm by more than a factor of 2.

As we have seen in the overview of prior work, lower bounds on the query complexity for testing k-linearity (or for related problems) yield lower bounds for the query complexity of other property testing problems as well. As we show in Section 7.3, the lower bound on testing k-linearity in Theorem 7.1 indeed gives improvements on the best-known lower bounds for many other property testing problems.

The proof of Theorem 7.1 proceeds by reducing the problem of testing k-linearity to a purely geometric problem on the boolean hypercube. We describe that geometric problem and solve it in Section 7.1. We then complete the proof of the theorem in Section 7.2.

7.1 Affine subspaces and layers of the hypercube

We reduce the problem of testing k-linear functions to a purely geometric problem on the Hamming cube.

[2] See Section 7.3.7 for more details.

Namely, we obtain our testing lower bound by showing that affine subspaces of large dimension intersect roughly the same fraction of the middle layers of the cube. More precisely, let $W_k \subseteq \{0,1\}^n$ denote the set of vectors $x \in \{0,1\}^n$ of Hamming weight k. Our main technical contribution is the following result.

Lemma 7.2. *There is a constant $c > 0$ such that for any affine subspace $V \subseteq \{0,1\}^n$ of dimension $d \geq \frac{n}{2} + cn^{2/3}$,*

$$\left| \frac{|V \cap W_{\frac{n}{2}-1}|}{|W_{\frac{n}{2}-1}|} - \frac{|V \cap W_{\frac{n}{2}+1}|}{|W_{\frac{n}{2}+1}|} \right| \leq \tfrac{1}{36} 2^{-d}.$$

Proof. For any set $A \subseteq \{0,1\}^n$, define $I_A : \{0,1\}^n \to \{0,1\}$ to be the indicator function for A. For a given function $f : \{0,1\}^n \to \{0,1\}$, let us write $\mathbf{E}[f]$ as shorthand for $\mathbf{E}_x[f(x)]$ where the expectation is over the uniform distribution of $x \in \{0,1\}^n$. Similarly, for two functions f, g, we write $\mathbf{E}[f \cdot g]$ as short-hand for $\mathbf{E}_x[f(x) \cdot g(x)]$.

For any subsets $A, B \subseteq \{0,1\}^n$, $|A \cap B| = 2^n \cdot \mathbf{E}[I_A \cdot I_B]$. Since $|W_{\frac{n}{2}-1}| = |W_{\frac{n}{2}+1}| = \binom{n}{\frac{n}{2}-1}$,

$$\left| \frac{|V \cap W_{\frac{n}{2}-1}|}{|W_{\frac{n}{2}-1}|} - \frac{|V \cap W_{\frac{n}{2}+1}|}{|W_{\frac{n}{2}+1}|} \right| = \frac{2^n}{\binom{n}{\frac{n}{2}-1}} \cdot \mathbf{E}\left[I_V \cdot (I_{W_{\frac{n}{2}-1}} - I_{W_{\frac{n}{2}+1}}) \right].$$

The subspace V can be defined by a set $S \subseteq [n]$ of size $|S| = d$ and an affine-linear function $f : \{0,1\}^{n-d} \to \{0,1\}^d$, where $x \in V$ iff $x_S = f(x_{\bar{S}})$. Define I^S_m and $I^{\bar{S}}_m$ to be indicator functions for $|x_S| = m$ and $|x_{\bar{S}}| = m$, respectively. Then

$$\mathbf{E}[I_V \cdot (I_{W_{\frac{n}{2}-1}} - I_{W_{\frac{n}{2}+1}})] = \sum_{m=0}^{d} \mathbf{E}\left[I_V \cdot I^S_m \cdot (I^{\bar{S}}_{\frac{n}{2}-m-1} - I^{\bar{S}}_{\frac{n}{2}-m+1}) \right].$$

Let $U \subseteq \{0,1\}^S$ be the image of f. Let $d' = \dim(U)$. Define $h_m : \{0,1\}^S \to [-1,1]$ by setting $h_m(u) = \mathbf{E}_{x \in \{0,1\}^{\bar{S}}}[I_V(x,u) \cdot (I^{\bar{S}}_{\frac{n}{2}-m-1}(x) - I^{\bar{S}}_{\frac{n}{2}-m+1}(x))]$. Note that $h_m = f_* \left(I^{\bar{S}}_{\frac{n}{2}-m-1} - I^{\bar{S}}_{\frac{n}{2}-m+1} \right)$. Notice also that h_m is supported on U. We have

$$\mathbf{E}[I_V \cdot (I_{W_{\frac{n}{2}-1}} - I_{W_{\frac{n}{2}+1}})] = \sum_{m=0}^{d} \mathbf{E}\left[I^S_m \cdot h_m \right] = \sum_{m=0}^{d} \mathbf{E}\left[I^S_m \cdot \mathbf{1}_U \cdot h_m \right]. \tag{7.1}$$

Two applications of the Cauchy-Schwarz inequality yield

$$\sum_{m=0}^{d} \mathbf{E}\left[I^S_m \cdot \mathbf{1}_U \cdot h_m \right] \leq \sum_{m=0}^{d} \|I^S_m \cdot \mathbf{1}_U\|_2 \cdot \|h_m\|_2 \leq \sqrt{\sum_{m=0}^{d} \|I^S_m \cdot \mathbf{1}_U\|_2^2} \cdot \sqrt{\sum_{m=0}^{d} \|h_m\|_2^2}. \tag{7.2}$$

We now bound the two terms on the right-hand side. For the first term, we have

$$\sum_{m=0}^{d} \|I_m^S \cdot \mathbf{1}_U\|_2^2 = \sum_m \mathop{\mathbf{E}}_x [I_m^S(x)^2 \cdot \mathbf{1}_U] = \mathop{\mathbf{E}}_x \left[\mathbf{1}_U \sum_m I_m^S(x)^2 \right] = 2^{d'-d}, \tag{7.3}$$

where the last equality follows from the fact that for every $x \in \{0,1\}^n$, there is exactly one m for which $I_m^S(x) = 1$.

We now examine the second term. By Parseval's Identity, $\|h_m\|_2^2 = \sum_{\alpha \in \{0,1\}^S} \hat{h}_m(\chi_\alpha)^2$. Suppose that the image of f has dimension $d' \leq d$. Then, since h_m is a pushforward,

$$\hat{h}_m(\chi) = 2^{-d} \left(\widehat{I}^{\bar{S}}_{\frac{n}{2}-m-1}(\chi \circ f) - \widehat{I}^{\bar{S}}_{\frac{n}{2}-m+1}(\chi \circ f) \right).$$

The characters $\chi \circ f$ depend only on the restriction of χ to $f(\{0,1\}^{\bar{S}})$. Thus these characters all lie in some subspace $W \subseteq \widehat{\{0,1\}}^{\bar{S}}$ of dimension d', with each character appearing $2^{d-d'}$ times. Thus, we have that

$$\|h_m\|_2^2 = 2^{-d-d'} \sum_{\chi \in W} \left(\widehat{I}^{\bar{S}}_{\frac{n}{2}-m-1}(\chi) - \widehat{I}^{\bar{S}}_{\frac{n}{2}-m+1}(\chi) \right)^2.$$

For any set $\chi \subseteq \bar{S}$, we can apply Facts 4.13 and 4.12(i) to obtain

$$\widehat{I}^{\bar{S}}_{\frac{n}{2}-m+1}(\chi) - \widehat{I}^{\bar{S}}_{\frac{n}{2}-m-1}(\chi) = 2^{-(n-d)} K^{n-d+2}_{\frac{n}{2}-m+1}(|\chi|+1).$$

Therefore, $\sum_{m=0}^{d} \|h_m\|_2^2 = 2^{-2n+d-d'} \sum_m \sum_{\chi \in W} K^{n-d+2}_{\frac{n}{2}-m+1}(|\chi|+1)^2$ and by Fact 4.12(ii),

$$\sum_{m=0}^{d} \|h_m\|_2^2 \leq 2^{-2n+d-d'} \sum_{\chi \in W} (-1)^{|\chi|+1} K^{2(n-d+1)}_{n-d+1}(2|\chi|+2). \tag{7.4}$$

There exist some d' coordinates such that the projection of W onto those coordinates is surjective. Therefore the number of elements of W with weight at most ℓ is at most $\sum_{j=1}^{\ell} \binom{d'}{\ell}$. We also have a similar bound on the number of elements of W of size at least $n-d-\ell$. Therefore, since by Fact 4.12(v) the summand in (7.4) is decreasing in $\min(|\chi|, n-d-|\chi|)$, we have

$$\sum_{m=0}^{d} \|h_m\|_2^2 \leq 2^{-2n+d-d'+1} \sum_{j=0}^{d'} \binom{d'}{j} (-1)^{j+1} K^{2(n-d+1)}_{n-d+1}(2j+2).$$

By Fact 4.12(iii), the sum on the right-hand side evaluates to $-K_{n-d-d'+1}^{2(n-d-d'+1)}(2)$. We can then apply the generating function representation of Krawtchouk polynomials to obtain

$$\begin{aligned}\sum_{m=0}^{d} \|h_m\|_2^2 &\leq -2^{-2n+d+d'+1}[x^{n-d-d'+1}](1-x)^2(1+x)^{2(n-d-d')} \\ &= 2^{-2n+d+d'+2}\left(\binom{2(n-d-d')}{n-d-d'} - \binom{2(n-d-d')}{n-d-d'-1}\right) \\ &= 2^{-d-d'}\Theta(n-d-d')^{-3/2} = 2^{-d-d'}O\left((n-2d)^{-3/2}\right).\end{aligned}$$

Thus we have that

$$\mathbf{E}[I_V \cdot (I_{W_{\frac{n}{2}+1}} - I_{W_{\frac{n}{2}-1}})] \leq \sqrt{2^{d'-d}}\sqrt{2^{-d-d'}O\left((n-2d)^{-3/2}\right)} = 2^{-d}O\left((n-2d)^{-3/4}\right).$$

When $d = \frac{n}{2} - cn^{2/3}$ for some large enough constant $c > 0$, we therefore have $\mathbf{E}[I_V \cdot (I_{W_{\frac{n}{2}+1}} - I_{W_{\frac{n}{2}-1}})] < \frac{1}{36}\binom{n}{\frac{n}{2}-1}2^{-n-d}$ and the lemma follows. □

7.2 Proof of Theorem 7.1

We are now ready to complete the proof of Theorem 7.1 by using the lemma from the last section.

Theorem 7.1 (Restated). *For any $0 < k < n$ and any $0 < \epsilon < \frac{1}{2}$, ϵ-testing k-linearity requires at least $\min\{k, n-k\} \cdot (1 - o(1))$ queries.*

Proof. We first prove the special case where $k = \frac{n}{2} - 1$. There is a natural bijection between linear functions $\{0,1\}^n \to \{0,1\}$ and vectors in $\{0,1\}^n$: associate $f(x) = \sum_{i \in S} x_i$ with the vector $\alpha \in \{0,1\}^n$ whose coordinates satisfy $\alpha_i = \mathbf{1}[i \in S]$. Note that $f(x) = \alpha \cdot x$.

For $0 \leq \ell \leq n$, let $W_\ell \subseteq \{0,1\}^n$ denote the set of elements of Hamming weight ℓ. Fix any set $X \subseteq \{0,1\}^n$ of $q < \frac{n}{2} - O(n^{2/3})$ queries and any response vector $r \in \{0,1\}^q$. The set of linear functions that return the response vector r to the queries in X corresponds in our bijection to an affine subspace $V \subseteq \{0,1\}^n$ of dimension $n - q$. This is because for each $x \in X$, the requirement that $f(x) = r_i$ imposes an affine linear relation on f. By Lemma 7.2, this subspace satisfies the inequality

$$\left| \frac{|V \cap W_{\frac{n}{2}-1}|}{|W_{\frac{n}{2}-1}|} - \frac{|V \cap W_{\frac{n}{2}+1}|}{|W_{\frac{n}{2}+1}|} \right| \leq \tfrac{1}{36}2^{-q}. \tag{7.5}$$

Define $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$ to be the uniform distributions over $(\frac{n}{2} - 1)$-linear and $(\frac{n}{2} + 1)$-linear functions, respectively. By our bijection, $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$ correspond to the uniform distributions over $W_{\frac{n}{2}-1}$ and $W_{\frac{n}{2}+1}$. As a result, the probability that a function drawn from $\mathcal{D}_{\text{yes}}$ or from $\mathcal{D}_{\text{no}}$ returns the response r to the set of queries X is

$$\Pr_{f\sim\mathcal{D}_{\text{yes}}}[f(X) = r] = \frac{|V \cap W_{\frac{n}{2}-1}|}{|W_{\frac{n}{2}-1}|} \quad \text{and} \quad \Pr_{f\sim\mathcal{D}_{\text{no}}}[f(X) = r] = \frac{|V \cap W_{\frac{n}{2}+1}|}{|W_{\frac{n}{2}+1}|}.$$

So (7.5) and Lemma 3.15 imply that at least $\frac{n}{2} - O(n^{2/3})$ queries are required to distinguish $(\frac{n}{2} - 1)$-linear and $(\frac{n}{2} + 1)$-linear functions. All $(\frac{n}{2} + 1)$-linear functions are $\frac{1}{2}$-far from $(\frac{n}{2} - 1)$-linear functions, so this completes the proof of the theorem for $k = \frac{n}{2} - 1$.

For other values of k, we apply a simple padding argument. When $k < \frac{n}{2} - 1$, modify $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$ to be uniform distributions over k-linear and $(k + 2)$-linear functions, respectively, under the restriction that all coordinates in the sum taken from the set $[2k + 2]$. This modification with $k = \frac{n}{2} - 2$ shows that $\frac{n}{2} - O(n^{\frac{2}{3}})$ queries are required to distinguish $(\frac{n}{2} - 2)$- and $\frac{n}{2}$-linear functions; this implies the lower bound in the theorem for the case $k = \frac{n}{2}$. □

7.3 Implications

By examining the proof of Theorem 7.1, we see that the proof actually establishes a stronger result: even if we are promised that the input function is either a k-linear function or a $(k+2)$-linear function, we still need $\min(k, n-k) \cdot (1 - o(1))$ queries to distinguish between the two cases. In other words, we proved the following lemma.

Lemma 7.3. *For any* $0 < k < n$ *and any* $0 < \epsilon < \frac{1}{2}$*, at least* $\min\{k, n-k\} \cdot (1 - o(1))$ *queries are required to distinguish* k*-linear and* $(k + 2)$*-linear functions with probability at least* $\frac{2}{3}$*.*

This lemma is particularly helpful for establishing other lower bounds in property testing. Given a property $\mathcal{P}$, if we can identify some $0 < k < n$ such that k-linear functions are contained in $\mathcal{P}$ and $(k + 2)$-linear functions are ϵ-far from $\mathcal{P}$, then we immediately obtain a lower bound of $\min(k, n - k) \cdot (1 - o(1))$ queries on the query complexity for ϵ-testing $\mathcal{P}$. In the rest of this section, we apply this method to a number of different properties. A summary of the results obtained in this section is presented in Table 7.1.

Property	**Lower bound**	**Previous lower bound**	**Upper bounds**
k-linearity	$\boldsymbol{k - o(k)}$	$\Omega(\sqrt{k})$ [57] $\Omega(k)$ (n.a.) [57]	$O(k \log k)$ [40] $O(n)$ (folklore)
$\leq k$-linearity	$\boldsymbol{k - o(k)}$	$\Omega(k/\log k)$ [57, 38] $\Omega(k)$ (n.a.) [57]	$O(k \log k)$ [40] $O(n)$ (folklore)
k-juntas	$k - o(k)$	$\Omega(k)$ [41]	$O(k \log k)$ [20]
Fourier degree $\leq d$	$d - o(d)$	$\Omega(d)$ [40]	$2^{O(d)}$ [46, 40]
s-sparse polynomials	$\boldsymbol{s - o(s)}$	$\Omega(\sqrt{s})$ [39]	$\tilde{O}(s)$ [39]
size-s branching programs	$\boldsymbol{s - (s)}$	$s^{\Omega(1)}$ [39]	$\tilde{O}(s)$ [39]
size-s decision trees	$\log s - o(\log s)$	$\Omega(\log s)$ [39]	$\tilde{O}(s)$ [39]

Table 7.1: Results implied by Lemma 7.3. All bounds are stated under the assumption that k, d, and s are at most $\frac{n}{2}$. Bold font indicates an asymptotic improvement over the previous bounds. Bounds labeled with (n.a.) apply only to non-adaptive testers.

7.3.1 $\leq k$-linearity

We begin with an easy corollary. The function $f : \{0,1\}^n \to \{0,1\}$ is *$\leq k$-linear* iff it is ℓ-linear for some $\ell \leq k$. Lemma 7.3 gives a strong lower bound for the query complexity of testing this property.

Corollary 7.4. *For any $0 < k < n$ and any $0 < \epsilon < \frac{1}{2}$, ϵ-testing $\leq k$-linearity requires at least $\min\{k, n-k\} \cdot (1 - o(1))$ queries.*

Proof. By definition, k-linear functions are also $\leq k$-linear. Proposition 2.24 implies that $(k+2)$-linear functions are $\frac{1}{2}$-far from every $\leq k$-linear function. Therefore, any $\leq k$-linearity tester must be able to distinguish k-linear and $(k+2)$-linear functions with probability at least $\frac{2}{3}$ and the corollary follows from Lemma 7.3. □

7.3.2 Fourier degree

Recall that the function $f : \{0,1\}^n \to \{0,1\}$ has *Fourier degree at most d* if $\hat{f}(\alpha) = 0$ for every element $\alpha \in \{0,1\}^n$ of Hamming weight $\|\alpha\| > d$.

Corollary 7.5. *For any $0 < k < n$ and any $0 < \epsilon < \frac{1}{2}$, ϵ-testing $f : \{0,1\}^n \to \{0,1\}$ for the property of having Fourier degree at most d requires at least $\min\{k, n-k\} \cdot (1 - o(1))$ queries.*

Proof. By Proposition 2.23, k-linear functions have Fourier degree k and by Proposition 2.24, $(k+2)$-linear functions are $\frac{1}{2}$-far from all functions with Fourier degree at most k. So the corollary again follows directly from Lemma 7.3. □

7.3.3 Juntas

Another easy corollary of Lemma 7.3 gives a good lower bound on the query complexity for testing juntas.

Corollary 7.6. *For any $0 < k < n$ and any $0 < \epsilon < \frac{1}{2}$, ϵ-testing k-juntas requires at least $\min\{k, n-k\} \cdot (1 - o(1))$ queries.*

Proof. When $f : \{0,1\}^n \to \{0,1\}$ is k-linear, it is clearly also a k-junta. When f is $(k+2)$-linear, then Proposition 2.24 implies that f is $\frac{1}{2}$-far from all k-junta functions. The corollary follows immediately from Lemma 7.3. □

7.3.4 Sparse polynomials

The function $f : \{0,1\}^n \to \{0,1\}$ is an *s-sparse polynomial* if the corresponding function $f_{\mathbb{F}_2} : \mathbb{F}_2^n \to \mathbb{F}_2$ is a polynomial with at most s monomials. We can again use Lemma 7.3 to prove a lower bound on the query complexity for testing s-sparse polynomials. In order to do so, however, we need to show that $(s+2)$-linear functions are far from all s-sparse polynomials. This was first done by Diakonikolas et al. [46, Thm. 36].

Lemma 7.7 (Diakonikolas et al. [46])**.** *Fix $0 < k \leq n-2$. Let $f : \{0,1\}^n \to \{0,1\}$ be an $(k+2)$-linear function. Then f is $\frac{1}{20}$-far from every k-sparse polynomial.*

Proof. Without loss of generality, let $f : x \mapsto x_1 \oplus \cdots \oplus x_{k+2}$. Let g be an k-sparse polynomial, i.e. $g = T_1 \oplus \cdots \oplus T_k$ where each T_i is a monomial. We want to show that f and g are far. We can assume without loss of generality that g does not contain any length-1 terms, since if it did we could just subtract those terms off of both f and g to create f' and g', which have the same distance from each other. We could then prove the theorem for f', g', and a smaller value of k.

Define the *influence* of a variable x_i in f, denoted $\mathrm{Inf}_i(f)$, in the standard way- i.e. $\mathrm{Inf}_i(f) = \Pr_x[f(x) \neq f(x^{\oplus i})]$ where $x^{\oplus i}$ denotes x with the ith bit flipped. Define the *total influence* of f to be $\sum_i \mathrm{Inf}_i(f)$.

For any f and g, it is straightforward to show that if for some i the difference $|\mathrm{Inf}_i(f) - \mathrm{Inf}_i(g)|$ is at least δ, then f and g must have distance at least $\delta/2$. When f is the $(k+2)$-linear function defined above, each variable x_1 through x_{k+2} has influence 1. Thus, to complete the proof, we will show that in g one of these variables must have influence at most 0.9.

If the total influence of x_1 through x_{k+2} in g is less than $0.9(k+2)$, then we are done, since the pigeonhole principle implies the existence of a variable x_i with influence at most 0.9. Thus, in what follows, we assume

$$\sum_{i}^{k+2} \mathrm{Inf}_i(g) \geq 0.9(k+2) \ . \tag{7.6}$$

We can bound the total influence of x_1 through x_{k+2} in g as follows. First, we write $g = g_2 \oplus g_3$ where g_2 is the collection of terms in g that have length 2, and g_3 is the collection of terms in g that have length at least 3. Now note:

- Each variable x_i that appears in g_2 has $\mathrm{Inf}_i(g_2) = 1/2$. The reason is because since every term of g_2 has length 2, x_i is influential exactly when the other variables it appears with have parity 1, which happens exactly half the time.

- For each term in g_3, the total contribution of that term to the influences of all the variables is at most 3/4. To see why, suppose the term has length m, then on a random assignment the probability that a variable is relevant to that term is $\frac{1}{2^{m-1}}$, so the total effect the term can have on all the influences is at most $m \cdot \frac{1}{2^{m-1}}$. If $m \geq 3$, this is at most $3/4$.

Let R_2 be the number of terms of g_2, and R_3 be the number of terms in g_3. By hypothesis, $R_2 + R_3 \leq k$. Since each term of g_2 contributes at most 1 to the total influence of g, and each term of g_3 contributes at most $3/4$ to the total influence of g, we have that

$$\sum_{i}^{k+2} \mathrm{Inf}_i(g) \leq R_2 + (3/4)R_3 \ . \tag{7.7}$$

Combining equations 7.6 and 7.7 we get that $R_2 + (3/4)R_3 \geq (9/10)k$. Using the fact that $R_2 + R_3 \leq k$, this implies that $R_3 \leq (4/10)k$, in other words there cannot be too many terms of length 3 or more in g. Now we can bound the influence of variables x_1 through

x_{k+2} in g.

$$\begin{aligned}\sum_i^{k+2} \mathrm{Inf}_i(g) &\leq \sum_i^{k+2} [\mathrm{Inf}_i(g_2) + \mathrm{Inf}_i(g_3)] \\ &\leq \sum_i^{k+2} \mathrm{Inf}_i(g_2) + \sum_i^{n} \mathrm{Inf}_i(g_3) \\ &\leq \frac{1}{2}(k+2) + \frac{3}{4} \cdot R_3 \\ &\leq \frac{1}{2}(k+2) + \frac{3}{4} \cdot \frac{4}{10} \cdot k \\ &< 0.9(k+2)\ .\end{aligned}$$

By the pigeonhole principle, there must exist a variable x_i with influence at most 0.9 in g. □

Theorem 7.8. *For any $0 < s < n$ and any $0 < \epsilon < \frac{1}{20}$, ϵ-testing s-sparse polynomials requires at least $\min\{s, n-s\} \cdot (1 - o(1))$ queries.*

Proof. When $f : \{0,1\}^n \to \{0,1\}$ is s-linear, it is an s-sparse polynomial. When f is $(s+2)$-linear, then Lemma 7.7 implies that f is $\frac{1}{20}$-far from all s-sparse polynomials. The corollary follows immediately from Lemma 7.3. □

7.3.5 Decision trees

We can also use Lemma 7.3 to prove lower bounds for properties related to the computational complexity of a boolean function. We first show how it implies a lower bound for testing whether a function can be computed by a small decision tree.

Lemma 7.9. *Fix $s \geq 1$ and $0 < \alpha < 1$. Let $f : \{0,1\}^n \to \{0,1\}$ be an s-linear function. Then f can be computed by a decision tree of size 2^s and is $\frac{1-\alpha}{2}$-far from all functions that are computable by decision trees of size at most $\alpha\, 2^s$.*

Proof. To construct a decision tree of size 2^s that computes the function $f : x \mapsto x_{i_1} \oplus \cdots \oplus x_{i_s}$, create a complete tree of depth s where each node at level j of the tree queries x_{i_j}. This tree has 2^s leaves and, by setting the value of each leaf appropriately, computes the function f exactly.

Consider now a decision tree T of size at most $\alpha\, 2^s$, and let $g : \{0,1\}^n \to \{0,1\}$ be the function computed by this tree. We want to show that $\Pr[f(x) \neq g(x)] \geq \frac{1-\alpha}{2}$ when the probability is over the uniform distribution of $x \in \{0,1\}^n$. For each leaf ℓ of T, let $\mathrm{depth}(\ell)$ denote the number of unique variables queried by the nodes in the path from the root of T to ℓ and let $R_\ell \subseteq \{0,1\}^n$ represent the set of inputs $x \in \{0,1\}^n$ that define a path in T that reaches ℓ. (Note that the sets R_ℓ form a partition of $\{0,1\}^n$.) Define $B := \bigcup_{\ell \,:\, \mathrm{depth}(\ell) < s} R_\ell$ to be the union of the sets R_ℓ for all the leaves in T of depth strictly less than s. Then

$$\Pr[f(x) \neq g(x)] \geq \Pr[f(x) \neq g(x) \cap x \in B] = \Pr[x \in B] \cdot \Pr[f(x) \neq g(x) \mid x \in B].$$

For any leaf ℓ of T, the probability that an input x chosen uniformly at random from $\{0,1\}^n$ reaches ℓ is $2^{-\mathrm{depth}(\ell)}$. By the union bound, the probability that x reaches a leaf of depth at least s in T is at most $\alpha\, 2^s \cdot 2^{-s} = \alpha$, so $\Pr[x \in B] \geq 1 - \alpha$.

Let ℓ be a leaf in T of depth at most $s-1$. Then there is some index $i \in \{i_1, \ldots, i_s\}$ that is not queried in the path from the root of T to ℓ. We can partition R_ℓ into pairs $(x, x^{(i)})$ where each pair is identical in all but the i-th coordinate. For each such pair, $f(x) \neq f(x^{(i)})$ so no matter what label is attached to the leaf ℓ, we have $\Pr[f(x) \neq g(x) \mid x \in R_\ell] = \frac{1}{2}$. This also implies that $\Pr[f(x) \neq g(x) \mid x \in B] = \frac{1}{2}$ and, therefore, $\Pr[f(x) \neq g(x)] \geq (1-\alpha) \cdot \frac{1}{2} = \frac{1-\alpha}{2}$, as we wanted to show. □

Theorem 7.10. *For any $0 < s < n$ and any $0 < \epsilon < \frac{3}{8}$, ϵ-testing size-2^s decision trees requires at least $\min\{s, n-s\} \cdot (1 - o(1))$ queries.*

Proof. By Lemma 7.9, when $f : \{0,1\}^n \to \{0,1\}$ is s-linear it can be computed by a decision tree of size 2^s and when f is $(s+2)$-linear it is $\frac{3}{8}$-far from all decision trees of size 2^s.. The corollary follows immediately from Lemma 7.3. □

7.3.6 Branching programs

We show that Lemma 7.9 also implies a nearly-optimal lower bound on the query complexity for testing whether a function can be computed by a small-size branching program.

Lemma 7.11. *Let $\mathcal{P}$ be the class of all boolean functions computable by branching programs of size $2s$. Then every s-linear function is in $\mathcal{P}$ while every $(s+2)$-linear function is $\frac{1}{6}$-far from $\mathcal{P}$.*

Proof. Again, the first assertion is almost immediate: consider a branching program of width 2 that queries x_{i_1} at the start node and queries x_{i_j} on both nodes at level $1 < j \leq s$. We can arrange the edges of this branching program so that the left (resp., right) node at level j is reached when $x_{i_1} \oplus \cdots \oplus x_{i_{j-1}}$ equals 0 (resp., equals 1). This branching program has size $2s - 1$ and computes the s-linear function.

For the second assertion, let P be a branching program of size $2s$, and suppose it is close to some $(s+2)$-linear function h. Note that if one of the $s+2$ variables in h does not appear in P, then h and P are $\frac{1}{2}$-far, since for every input there is a variable whose value we can flip to change the value of h without changing the output of P.

Thus, we assume that every variable in h appears in P. Moreover, since P has only $2s$ nodes, there must be at least two variables in h that are queried only once in P. Let x_1 and x_2 denote two such variables, and let u_1 and u_2 denote the corresponding nodes in P. The graph of P is directed and acyclic, so we can assume without loss of generality that no path reaches the node u_1 after reaching u_2.

Consider the paths in P generated by strings $x, x^{(1)} \in \{0,1\}^n$, where x is generated uniformly at random and $x^{(1)}$ is generated from x by flipping x_1. Note that $x^{(1)}$ is also uniform. If the random path generated by x reaches u_2 with probability less than $\frac{2}{3}$, then with probability at least $\frac{1}{3}$, flipping the value of x_2 changes the value of h without changing the output of P; hence, P is $\frac{1}{6}$-far from h. On the other hand, if this random path reaches u_2 with probability at least $\frac{2}{3}$, then the path generated by $x^{(1)}$ also reaches u_2 with probability $\frac{2}{3}$. By the union bound, the probability that *both* x and $x^{(1)}$ describe paths in P reaching u_2 is at least $\frac{1}{3}$. But since u_1 cannot be reached after u_2, this means that both x and $x^{(1)}$ describe paths to the same terminal in P even though they have different values in h. Therefore, P is $\frac{1}{6}$-far from h in this case too. □

Theorem 7.12. *For any $0 < s < n$ and any $0 < \epsilon < \frac{1}{6}$, ϵ-testing size-s branching programs requires at least $\min\{s, n-s\} \cdot (1 - o(1))$ queries.*

Proof. By Lemma 7.11, when $f : \{0,1\}^n \to \{0,1\}$ is s-linear it can be computed by a branching program of size s and when f is $(s+2)$-linear it is $\frac{1}{6}$-far from all functions computable by branching programs of size s.. The corollary follows immediately from Lemma 7.3. □

7.3.7 Small-width OBDDs

An *ordered binary decision diagram* (or OBDD) is an acyclic graph with a single root and at most $n+1$ levels of nodes. There are two nodes in the last level. These are called *sink* nodes; one is labeled 0 and the other one is labeled 1. The nodes in every other level have out-degree two, with one edge labeled 0 and the other one labeled 1. All edges leaving a node from the ℓth level go to a node in the $(\ell+1)$st level. Each level is labeled with some index $i \in [n]$. The first level of any OBDD contains a single node. The *width* of an OBDD is the maximum number of nodes at any level.

We say that an OBDD *computes* the function $f : \{0,1\}^n \to \{0,1\}$ if for every $x \in \{0,1\}^n$, the path through the OBDD defined by following the edge labeled x_i from the current node, where i is the label associated with the current node's level, leads to the sink node labeled with $f(x)$.

There are two natural question that we may ask related to property testing and small-width OBDDs—or, for that matter, any other complexity class. How many queries do we need to test if a function is computable by OBDDs of width w? And are there properties that contain *only* functions computable by OBDDs of width w that are much harder to test?

Ron and Tsur [88] showed that testing whether a function is computable by width-2 OBDDs can be done with $O(\log n)$ queries. Goldreich [57] showed that there are properties that contain only functions computable by width-2 OBDDs that are much harder to test: they require $\Omega(n)$ queries to test. Theorem 7.1 immediately yields a slight sharpening of this result.

Corollary 7.13. *There is a property $\mathcal{P}$ containing only functions computable by width-2 OBDDs that requires $\frac{n}{2} - o(n)$ queries to test.*

Proof. Note that all linear functions are computable by width-2 OBDDs. To see this, consider $f(x) = x_{i_1} \oplus \cdots \oplus x_{i_k}$. We can build a width-2 OBDD with $k+1$ levels where the levels are labeled with $i_1, \ldots, i_k$. For the levels $2, \ldots, k+1$, we associate one node with the value 1 and the other with the value 0. Let the edges labeled 0 (resp., 1) go to the node of the next level with the same (resp., different) value as the current node. Then the value of the node on level ℓ represents the parity of the variables $i_1, \ldots, i_{\ell-1}$.

As a result, all $\frac{n}{2}$-linear functions are also computable by width-2 OBDDs and the corollary follows directly from Theorem 7.1. □

7.4 Notes and Discussion

Alternative proof. The proof of the lower bound for testing k-linearity presented in this section was obtained in joint work with Daniel Kane [24]. In a simultaneous but separate project with Joshua Brody and Kevin Matulef [22, 23], we obtained a different proof of the same (asymptotic) lower bound. That proof is presented in Chapter 11.

Non-adaptive testing of k-linearity. The manuscript [24] includes the result presented in this chapter as well as one more lower bound. The second lower bound shows that testing k-linearity non-adaptively requires $2 \cdot \min\{k, n-k\} - O(1)$ queries. (This lower bound is twice as large as the lower bound for adaptive algorithms.)

Asymptotically, the two lower bounds are equivalent. The non-adaptive lower bound, however, says something much stronger about the problem of testing $\frac{n}{2}$-linearity: it says that no non-adaptive tester can improve on the query complexity of the folklore tester by more than an *additive* constant.

One may wonder if the same strong statement also applies to adaptive algorithms: is it possible to show that all adaptive testers for $\frac{n}{2}$-linearity also require $n - O(1)$ queries? The answer is no. In [24], we show that there is an adaptive algorithm for testing $\frac{n}{2}$-linearity with $\frac{2}{3}n + o(n)$ queries. This shows that it is indeed possible to beat the folklore tester by more than an additive constant if we can choose the queries adaptively. It also gives a gap—albeit a small one—between the query complexities of testing $\frac{n}{2}$-linearity adaptively and non-adaptively.

Part II

Testing Function Isomorphism

Chapter 8

Testing Isomorphism to Partially Symmetric Functions

The first part of this thesis was mainly concerned with determining the exact query complexity for testing various properties of boolean functions. The current part aims to understand *why* different properties of boolean functions can be tested with very few queries, while other properties cannot.

In the last few years, there has been great progress in understanding the testability of graph properties—properties of graphs $G = (V, E)$ that are invariant under relabeling of the vertices—and, to a large extent, the problem of characterizing the set of graph properties that are testable with a constant number of queries has been largely solved [5, 6, 7]. Recently, there has also been much progress in understanding the testability of hypergraph properties [11] and the testability of algebraic properties—that is, properties that are invariant under linear or affine transformations—of functions [18, 72, 95].

Our understanding of the testability of (non-algebraic) properties of boolean functions, however, remains largely incomplete. One approach for remedying this situation, first proposed by Fischer et al. [52], is to study the function isomorphism testing problem. Two boolean functions $f, g : \{0, 1\}^n \to \{0, 1\}$ are said to be *isomorphic* if they are identical up to permutation of the input labels. The *f-isomorphism* ϵ-testing problem asks a tester to determine whether a function g is isomorphic to f or whether it is ϵ-far from being so with as few queries as possible. When it is possible to ϵ-test f-isomorphism with a number of queries that depends on ϵ but not on n, we say that f is *efficiently isomorphism-testable*. The *function isomorphism testing (characterization) problem* asks us to determine the set of functions that are efficiently isomorphism-testable.

The current chapter's focus is on functions that are efficiently isomorphism-testable. That is, we seek *upper bounds* on the query complexity for testing f-isomorphism for different functions f. Lower bounds on the query complexity for testing function isomorphism will be established in the next two chapters.

Previous work. Let's begin with some folklore observations. When $f : \{0,1\}^n \to \{0,1\}$ is a (fully) symmetric function, then testing isomorphism to f is equivalent to testing *identity* to f. (Since the only function isomorphic to f is f itself.) So in this case ϵ-testing f-isomorphism can be done with $O(1/\epsilon)$ queries.

Another folklore result is obtained with a similar argument. For *any* function $f : \{0,1\}^n \to \{0,1\}$, there are at most $n!$ functions g that are isomorphic to f. As a result, by Corollary 3.14, we can ϵ-test f-isomorphism with $O(\log(n!)/\epsilon) = O(n\log(n)/\epsilon)$ queries.

The two above observations can be unified and generalized in the following way. For the function $f : \{0,1\}^n \to \{0,1\}$, let $\kappa(f) = |\{f^\pi : \pi \in \mathcal{S}_n\}|$ denote the number of distinct functions that are isomorphic to f. Then by Corollary 3.14 we can test f-isomorphism with $O(\log(\kappa_f)/\epsilon)$ queries.

The generalized folklore observation implies that when f is a k-junta, it is possible to ϵ-test f-isomorphism with $O(\log(\binom{n}{k}k!)/\epsilon) = O(k\log(n)/\epsilon)$ queries. Fischer et al. [52] improved on this result by showing that when f is a k-junta, it is possible to ϵ-test f-isomorphism with only $\mathrm{poly}(k,\epsilon)$ queries. In other words, when f is a junta (with a constant number of relevant variables), then f is efficiently isomorphism-testable.

The query complexity for testing juntas was sharpened by Chakraborty, García Soriano, and Matsliah [40]. By building on the junta test presented in Chapter 5, they showed that it is possible to test k-juntas with $O(k\log k)$ queries.

At a qualitative level, the folklore observations and the results in [52, 40] show that symmetric functions and junta functions are efficiently isomorphism-testable. When we began the research project presented in this chapter, those were the essentially the only functions that were known to be efficiently isomorphism-testable.

Our result. It is not too hard to see that juntas and symmetric functions cannot be the only functions that are efficiently isomorphism-testable. For example, consider the function defined by $f(x) = x_1 \oplus \cdots \oplus x_{n-1}$. This function is not symmetric and has $n-1$ relevant variables. This function, however, can be represented as the combination of a symmetric function and a junta function:

$$f(x) = (x_1 \oplus \cdots \oplus x_n) \oplus x_n.$$

We can use this observation to build an f-isomorphism tester that requires only $O(1/\epsilon)$ queries: given some function $g : \{0,1\}^n \to \{0,1\}$, we test whether the function g' obtained by setting

$$g'(x) = (x_1 \oplus \cdots \oplus x_n) \oplus g(x)$$

is isomorphic to the function $f'(x) = x_n$. We leave it to the reader to verify that when g is isomorphic to f, then g' is isomorphic to f' and that when g is ϵ-far from isomorphic to f then g' is also ϵ-far from isomorphic to f'.

When examining why the function f defined in the last paragraph is efficiently isomorphism-testable, one may notice that it is a partially symmetric function. This leads to the natural question: are all partially symmetric functions efficiently isomorphism-testable? The main result of this chapter gives an affirmative answer to this question.

Theorem 8.1. *For every $(n-k)$-symmetric function $f : \{0,1\}^n \to \{0,1\}$, there exists an ϵ-tester for f-isomorphism that requires only $O(k \log(k)/\epsilon^2)$ queries.*

This result presents a unified explanation for the efficient isomorphism-testability of juntas and of symmetric functions. Since many partially symmetric functions—such as the one in the example above—are not juntas or fully symmetric, it also significantly extends the set of functions that we know to be efficiently isomorphism-testable.

One might wonder if, in turn, the set of partially symmetric functions is only a subset of all the functions that are efficiently isomorphism-testable. In [26], we conjecture that the answer to this question is essentially "no": that partial symmetry is effectively the characteristic that determines whether functions are efficiently isomorphism-testable or not. This conjecture is still open; see [26] for the details.

The proof of Theorem 8.1 is constructive. In the next section, we introduce an algorithm for efficiently testing isomorphism to partially symmetric functions. The analysis of this algorithm, and thus the proof of the main theorem, follows in Section 8.2.

8.1 The Algorithm

The algorithm we introduce for testing isomorphism to partially symmetric functions follows the general outline of the isomorphism tester for juntas introduced by Fischer et al. [52] and refined by Chakraborty, García Soriano, and Matsliah [40]. The algorithm proceeds in two stages.

The first stage tests whether the given function is partially symmetric. We use the PARTIALLYSYMMETRICTEST algorithm from Chapter 6 for this task, with one minor modification: when that test accepts, it also returns the partition of $[n]$ that it defined and the asymmetric parts that it identified.

The second stage of the algorithm verifies that, if the function is indeed partially symmetric, it is consistent with the target function (up to relabeling of the input variables). This second stage relies on an efficient "core sampler" to reduce the number of queries.

8.1.1 Core sampler for partially symmetric functions

Let $f : \{0,1\}^n \to \{0,1\}$ be J-symmetric for some set $J \subseteq [n]$ of size $|J| = n - k$. This $(n-k)$-symmetric function can be represented in a concise manner as the function $f_{\text{core}} : \{0,1\}^k \times \{0,1,\ldots,n-k\}$. We call the function f_{core} the *core* of f. Two $(n-k)$-symmetric functions $f, g : \{0,1\}^n \to \{0,1\}$ are isomorphic iff their core functions are isomorphic. The latter task can be done with an efficient sample extractor.

In the following definition, let $\mathcal{D}^*_{n,k}$ be the distribution on pairs $(x, w) \in \{0,1\}^k \times \{0,1,\ldots,n-k\}$ where x is drawn from the uniform distribution over $\{0,1\}^k$ and w is drawn independently from the binomial distribution $\text{Bin}(n-k, \frac{1}{2})$. A perfect sample extractor for a partially symmetric function draws an input from $\mathcal{D}^*_{n,k}$ and returns the value of the core of that function on this input.

Definition 8.2. A *perfect sampler* for the $(n-k)$-symmetric function $f : \{0,1\}^n \to \{0,1\}$ is a randomized algorithm that queries f on a single input and returns a triplet $(x, w, z) \in \{0,1\}^k \times \{0,1,\ldots,n-k\} \times \{0,1\}$ where

1. $(x, w) \sim \mathcal{D}^*_{n,k}$; and
2. $f_{\text{core}}(x, w) = z$.

If we know the identity of the set J of k asymmetric variables in the $(n-k)$-symmetric function f, it is easy to design a perfect sampler for this function: draw $y \in \{0,1\}^n$ uniformly at random and return the triplet $(y_J, \|y_{\bar{J}}\|, f(y))$. If we do not know the exact set J of asymmetric variables, however, it is much easier to design an *approximate* sample extractor for the function

Definition 8.3. A δ-*sampler* for the $(n-k)$-symmetric function $f : \{0,1\}^n \to \{0,1\}$ is a randomized algorithm that queries f on a single input and returns a triplet $(x, w, z) \in \{0,1\}^k \times \{0,1,\ldots,n-k\} \times \{0,1\}$ where

1. The distribution $\mathcal{D}$ of (x, w) satisfies $\mathrm{d_{TV}}(\mathcal{D}, \mathcal{D}^*_{n,k}) \le \delta$; and

2. $z = f_{\text{core}}(x, w)$ with probability at least $1 - \delta$.

Given a partition $\mathcal{I} = \{I_1, \ldots, I_s\}$ of $[n]$, we say that $x \in \{0,1\}^n$ *respects* the partition $\mathcal{I}$ if for every $1 \le i \le s-1$, each coordinate in I_i has the same value in x (i.e., if for every part I_i we have either $x_I = e^I$ or $x_I = \mathbf{0}$). We define $\mathcal{D}_{\mathcal{I}}$ to be the distribution over $\{0,1\}^n$ obtained by the following procedure. First, we sample $w \sim \mathrm{Bin}(n, \frac{1}{2})$. If there exist some elements $x \in \{0,1\}^n$ of Hamming weight $\|x\| = w$ that respects the partition $\mathcal{I}$, we choose one of those elements uniformly at random and return it. If no such element exists, we return $\mathbf{0}$.

The following algorithm is an efficient (approximate) sampler for the core of a partially symmetric function when it is given a partition that splits the asymmetric variables among the sets $I_1, \ldots, I_{s-1}$.

```
SAMPLEPSF($f, I_1, \ldots, I_s, J$)
  1. Draw $y \sim \mathcal{D}_{\mathcal{I}}$.
  2. Let $x \in \{0,1\}^k$ be the value assigned to the parts in $J$.
  3. Return the triplet $(x, \|y\| - \|x\|, f(y))$.
```

8.1.2 The isomorphism-testing algorithm

We are now ready to describe the algorithm for isomorphism testing of $(n-k)$-symmetric functions. Given an $(n-k)$-symmetric function f, the following algorithm tests whether the input function g is isomorphic to f or ϵ-far from being so.

```
PSFISOTEST($f, k, g, \epsilon$)
  1. If PARTIALSYMMETRYTEST($g, k, \frac{\epsilon}{1000}$) does not accept, **reject**.
  2. Let $\{I_1, \ldots, I_s\}$ and $J \subseteq [s-1]$ be the partition and asymmetric
     parts defined by PARTIALSYMMETRYTEST.
  3. For $i = 1, \ldots, \Theta(k \log(k)/\epsilon^2)$,
     3.1. Draw $(x^{(i)}, w^{(i)}, z^{(i)}) \leftarrow$ SAMPLEPSF($f, I_1, \ldots, I_s, J$).
  4. **Accept** iff at least a $1 - \frac{\epsilon}{2}$ fraction of the triplets $(x^{(i)}, w^{(i)}, z^{(i)})$
     are consistent with the core function of some isomorphism of $f$.
```

8.2 Analysis of the Algorithm

We begin by establishing some technical properties about the distributions defined in the last section.

Proposition 8.4. *Let $J = \{j_1, \ldots, j_k\} \subseteq [n]$ be a set of size k, and $r = \Omega(k^2)$ be odd. If $x \sim \mathcal{D}_{\mathcal{I}}^{W}$ for a random partition $\mathcal{I}$ of $[n]$ into r parts and a random workspace $W \in \mathcal{I}$, then*

- *x is $o(1/n)$-close to being uniform over $\{0,1\}^n$, and*
- *$(x_J, \|x_{\overline{J}}\|)$ is c/k-close to being distributed according to $\mathcal{D}^*_{k,n}$, for our choice of $0 < c < 1$.*

Proof. We start with the first part of the proposition, showing x is almost uniform. Consider the following procedure to generate a random $\mathcal{I}, W$ and x. We draw a random Hamming weight $w \sim \mathcal{B}_{n,1/2}$ and define x' to be the input consisting of w ones followed by $n - w$ zeros. We choose a random partition $\mathcal{I}'$ of $[n]$ into r *consecutive* parts $I_1, \ldots, I_r$ (i.e., $I_1 = \{1, 2, \ldots, |I_1|\}, \ldots, I_r = \{n - |I_r| + 1, \ldots, n\}$) according to the typical distribution of sizes in a random partition. Let the workspace W' be the only part which contains the coordinate w (or I_1 if $w = 0$). We now apply a random permutation over x', $\mathcal{I}'$ and W' to get $x, \mathcal{I}$ and W.

The above procedure outputs a random element x that is uniformly distributed over $\{0,1\}^n$. The choice of $\mathcal{I}$ was also done at random, considering the applied permutation over $\mathcal{I}'$. The only difference is then in the choice of the workspace W, which can only be reflected in its size. However, when $r = o(\sqrt{n})$ we will choose the middle part as the workspace with probability $1 - o(1)$, regardless of its size. In the remaining cases, since there are $n/r = \Omega(\sqrt{n})$ parts, the possible parts to be chosen as workspace are a small fraction among all parts, and therefore W would be $o(1)$-close to being a random part.

Proving the second property of the proposition, we also consider two cases. When $r = o(\sqrt{n})$, with probability $1 - o(1)$, the workspace would have size $\omega(\sqrt{n})$ and also $w = n/2 + O(\sqrt{n})$. In such a case, the $r - 1$ parts (excluding the workspace) would be half zeros and half ones, and the marginal distribution over the number of ones in J would be $\mathcal{H}_{r-1,(r-1)/2,k}$ (assuming the elements of J are separated by $\mathcal{I}$, which happens with probability $1 - o(1)$). By Lemma 4.3, the distance between this distribution and $\mathcal{B}_{k,1/2}$ is bounded by $k/r < c/k$ for our choice of $0 < c < 1$. Since there is no restriction on the ordering of the sets, this is also the distance from uniform over $\{0,1\}^k$ as required.

In the remaining case where $r = \Omega(\sqrt{n})$, we can use the same arguments and also apply Lemma 4.4 with the distributions $\mathcal{B}_{k,1/2}$ and $\mathcal{B}_{k,1/2+\delta}$ for $\delta = O(1/\sqrt{n})$, implying the distance between these two distributions is at most $o(1)$. Combining this with the distance to $\mathcal{H}_{r-1,(r-1)(1/2+\delta),k}$ we get again a total distance of $k/r + o(1) < c/k$ for our choice of $0 < c < 1$. □

We now establish a basic fact regarding the functions that are accepted by the PARTIALLYSYMMETRICTEST.

Lemma 8.5. *Let g be a function ϵ-close to being $(n-k)$-symmetric which passed the* PARTIALLYSYMMETRICTEST(g, k, ϵ)*. In addition, let $\mathcal{I}, W$ and J be the partition, workspace and identified parts used by the algorithm. With probability at least $9/10$, there exists a function h which satisfies the following properties.*

- *h is 4ϵ-close to g, and*
- *h is $(n-k)$-symmetric whose asymmetric variables are contained in J and separated by $\mathcal{I}$.*

Proof. Let g^* be the $(n-k)$-symmetric function closest to g (which can be f itself, up-to some isomorphism) and R be the set of (at most) k asymmetric variables of g^*. By Lemma 6.3 and our assumption over g,

$$\mathrm{SymInf}_g(\overline{R}) \leq 2 \cdot \mathrm{dist}(g, g^*) \leq 2\epsilon \ .$$

Notice however that R is not necessarily contained in J and therefore g^* is not a good enough candidate for h. Let $U = R \cap J$ be the intersection of the asymmetric variables of g^* and the sets identified by the algorithm. In order to show that g is also close to being $\overline{U}$-symmetric, we bound $\mathrm{SymInf}_g(\overline{U})$ using Lemma 6.6 with the sets $\overline{R}$ and $\overline{J}$. Notice that since $|R| \leq k$ and $|J| \leq 2kn/r \leq \epsilon^2 n/c'$ for our choice of c', we can bound the error term (in the notation of Lemma 6.6) by $c\sqrt{\gamma} \leq c\sqrt{\epsilon^2/c'} \leq \epsilon$. We therefore have

$$\mathrm{SymInf}_g(\overline{U}) \leq \mathrm{SymInf}_g(\overline{R}) + \mathrm{SymInf}_g(\overline{J}) + \epsilon \leq 2\epsilon + \epsilon + \epsilon = 4\epsilon$$

where we know $\mathrm{SymInf}_g(\overline{J}) \leq \epsilon$ with probability at least $19/20$ as the algorithm did not reject.

By applying Lemma 6.3 again, we know there exists a $\overline{U}$-symmetric function h, whose distance to g is bounded by $\mathrm{dist}(g, h) \leq 4\epsilon$. Moreover, with probability at least $19/20$, all its asymmetric variables are completely separated by the partition $\mathcal{I}$ (and they were all identified as part of J). □

We now generalize a recent result of Chakraborty et al. [39]'s result concerning efficient algorithms for sampling the core of juntas.

Theorem 8.6. *Let $f : \{0,1\}^n \to \{0,1\}$ be $(n-k)$-symmetric with $k < n/10$. There is an algorithm that queries f on $O(\frac{k}{\eta\delta} \log \frac{k}{\eta\delta})$ inputs and with probability at least $1-\eta$ outputs a δ-sampler for f.*

Proof of Theorem 8.6. The algorithm for generating the sampler is described by PARTIALLYSYMMETRICSAMPLER, which performs $O(\frac{k}{\eta\delta} \log \frac{k}{\eta\delta})$ preprocessing queries to the function. What remains to be proved is that indeed with good probability, the algorithm returns a valid sampler.

Let h be the function defined in the analysis of Theorem 8.1, which satisfies the conditions of Lemma 8.5. Recall that its asymmetric variables were separated by $\mathcal{I}$ and appear in J. Following this analysis and that of PARTIALLYSYMMETRICTEST, one can see that with probability at least $1-\eta$ we would not reject f when calling PARTIALLYSYMMETRICTEST. Moreover, the samples would be $\delta/2$-close to sampling the core of h, which is by itself $\delta/2$-close to f. Therefore, overall our samples would be δ-close to sampling the core of f.

The last part in completing the proof of the theorem is showing that we sample the core with distribution δ-close to $\mathcal{D}^*_{k,n}$. By Proposition 8.4, the total variation distance between sampling the core according to $\mathcal{D}^*_{k,n}$ and sampling it according to $\mathcal{D}^W_{\mathcal{I}}$ is at most c/k for our choice of $0 < c < 1$, which we can choose it to be at most δ. □

Notice that if the function f is not $(n-k)$-symmetric but still very close (say $(k/\eta\delta)^2$-close), applying the same algorithm will provide a good sampler for an $(n-k)$-symmetric function f' close to f. The main reason is that most likely, we will not query any location of the function where it does not agree with f'.

Finally, we are ready to complete the analysis of PSFISOTEST.

Proof of Theorem 8.1. Before analyzing the algorithm we just described, we consider the case where $k > n/10$. Since Theorem 6.1 does not hold for such k's, we apply the basic algorithm of $O(n \log n/\epsilon)$ random queries, which is applicable testing isomorphism of any given function (since there are $n!$ possible isomorphisms, the random queries will rule out all of them with good probability, assuming we should reject). Since $k = \Omega(n)$, the complexity of this algorithm fits the statement of our theorem.

We start by analyzing the query complexity of the algorithm. The step of PARTIALLYSYMMETRICTEST performs $O(\frac{k}{\epsilon} \log \frac{k}{\epsilon})$ queries, and therefore the majority of the queries are performed at the sampling stage, resulting in $O(k \log k/\epsilon^2)$ queries as required. In order to prove the correctness of the algorithm, we consider the following cases.

- g is ϵ-far from being isomorphic to f and $\epsilon/1000$-far from being $(n-k)$-symmetric.
- g is ϵ-far from being isomorphic to f but $\epsilon/1000$-close to being $(n-k)$-symmetric.
- g is isomorphic to f.

In the first case, with probability at least $9/10$, PartiallySymmetricTest will reject and so will we, as required. We assume from this point on that PartiallySymmetricTest did not reject, as it will only reject g which is isomorphic to f with probability at most $1/10$, and that we are not in the first case. Notice that these cases match the conditions of Lemma 8.5, and therefore from this point onward we assume there exists an h satisfying the lemma's properties (remembering we applied the algorithm with $\epsilon/1000$).

In order to bound the distance between h and g in our samples, we use Proposition 8.4, indicating

$$\Pr_{\mathcal{I},W\in\mathcal{I},x\sim\mathcal{D}_{\mathcal{I}}^{W}}[g(x)\neq h(x)] = \mathrm{dist}(g,h)+o(1/n) \ .$$

By Markov's inequality, with probability at least $9/10$, the partition $\mathcal{I}$ and the workspace W satisfy

$$\Pr_{x\sim\mathcal{D}_{\mathcal{I}}^{W}}[g(x)\neq h(x)] \leq 10\cdot\mathrm{dist}(g,h)+o(1/n) \leq 10\cdot 4\epsilon/1000+o(1/n) < \epsilon/20 \ .$$

By Proposition 8.4, if we were to sample h according to $\mathcal{D}_{\mathcal{I}}^{W}$, it should be $\epsilon/20$-close to sampling its core (assuming the partition size is large enough). Combined with the distance between g and h in our samples, we expect our samples to be $\epsilon/20+\epsilon/20=\epsilon/10$ close to sampling h's core.

The last part of the proof is showing that there would be an almost consistent isomorphism of f only when g is isomorphic to f. Notice however that we care only for isomorphisms which map the asymmetric variables of f to the k sets of J. Therefore, the number of different isomorphisms we need to consider is $k!$.

Assume we are in the second case and g is ϵ-far from being isomorphic to f. Let f_π be some isomorphism of f. By our assumptions and Lemma 8.5,

$$\mathrm{dist}(f_\pi,h)\geq\mathrm{dist}(f_\pi,g)-\mathrm{dist}(g,h)\geq\epsilon-\epsilon/250 \ .$$

Each sample we perform would be inconsistent with f_π with probability at least $\epsilon-\epsilon/250-\epsilon/10 > 8\epsilon/9$. By the Chernoff bounds and the union bound, if we would perform $q=O(k\log k/\epsilon^2)$ queries, we would rule out all $k!$ possible isomorphisms with probability at least $9/10$ and reject the function as required.

On the other hand, if g is isomorphic to f, then we know there exists with probability at least $9/10$ some isomorphism f_π which maps the asymmetric variables of f into the sets of J, such that

$$\mathrm{dist}(f_\pi, h) \leq \mathrm{dist}(f_\pi, g) + \mathrm{dist}(g, h) \leq \epsilon/500 + \epsilon/250 \; .$$

For this isomorphism, with high probability much more than $(1 - \epsilon/2)$-fraction of the queries would be consistent and we would therefore accept g as we should. □

Chapter 9

Nearly Universal Lower Bound for Testing Isomorphism

In the last chapter, we saw that for every function $f : \{0,1\}^n \to \{0,1\}$, it is possible to ϵ-test f-isomorphism with $O(n \log(n)/\epsilon)$ queries. We also saw that for some functions—specifically, for partially symmetric functions—that universal upper bound is far from tight. The goal of this chapter is to see if this universal upper bound is tight—or nearly tight—for *any* functions, or whether f-isomorphism can be tested much more efficiently for every function f.

In the most extreme case, we might ask whether f-isomorphism can be tested with a *constant* number of queries for every function f. Fischer et al. [52] showed that this is not the case. Specifically, they showed that for every $k = o(\sqrt{n})$, testing isomorphism to the k-linear function $f(x) = x_1 \oplus \cdots \oplus x_k$ requires a number of queries that depends on k. So when $\omega(1) \leq k \leq o(\sqrt{k})$, testing isomorphism to k-linear functions requires a super-constant number of queries.

Until recently, that was the only lower bound known for the problem of testing function isomorphism. In this chapter, we provide a significantly stronger lower bound. We show that the universal upper bound of $O(n \log n)$ queries is tight, up to logarithmic factors, for *almost every* boolean function. More precisely, we establish the following lower bound.

Theorem 9.1. *Fix $0 < \epsilon < \frac{1}{2}$. For a $1-o(1)$ fraction of the functions $f : \{0,1\}^n \to \{0,1\}$, any non-adaptive algorithm for ϵ-testing isomorphism to f must make at least $\frac{n}{100}$ queries.*

The proof of the theorem that we present in this chapter is non-constructive: we show that if we pick a boolean function uniformly at random, then with probability $1 - o(1)$, testing isomorphism to that function requires at least $\frac{n}{100}$ queries.

9.1 The Lower Bound

The proof of Theorem 9.1 uses Yao's Minimax Principle [100]. For a fixed function f we introduce two distributions $\mathcal{F}_{\text{yes}}$ and $\mathcal{F}_{\text{no}}$ such that a function $g \sim \mathcal{F}_{\text{yes}}$ is isomorphic to f and a function $g \sim \mathcal{F}_{\text{no}}$ is ϵ-far from isomorphic to f with high probability. We then show that for most choices of f, deterministic non-adaptive testing algorithms cannot distinguish functions drawn from either of these distributions with only $\frac{n}{100}$ queries.

We define $\mathcal{F}_{\text{yes}}$ to be the uniform distribution over functions isomorphic to f. In other words, we draw a function $g \sim \mathcal{F}_{\text{yes}}$ by choosing $\pi \in \mathcal{S}_n$ uniformly at random and setting $g = f_\pi$.

A first idea for $\mathcal{F}_{\text{no}}$ may be to make it the uniform distribution over all boolean functions $\{0,1\}^n \to \{0,1\}$. This idea does not quite work, since, for example, a random function differs from f and all functions isomorphic to it on the all 0 input or the all 1 input with probability at least $3/4$. However, a simple modification of this idea does work: to draw a function $g \sim \mathcal{F}_{\text{no}}$, we choose a permutation $\pi \in \mathcal{S}_n$ uniformly at random and we choose a function g_{rand} uniformly at random from all boolean functions on n variables. We then let g be the function defined by

$$g(x) = \begin{cases} g_{\text{rand}}(x) & \text{if } \frac{n}{3} \le \|x\| \le \frac{2n}{3}, \\ f_\pi(x) & \text{otherwise.} \end{cases}$$

With high probability, a function $g \sim \mathcal{F}_{\text{no}}$ is far from isomorphic to f.

Proposition 9.2. *Fix $0 < \epsilon < \frac{1}{2}$. For any function $f : \{0,1\}^n \to \{0,1\}$, the function $g \sim \mathcal{F}_{\text{no}}$ is ϵ-close to isomorphic to f with probability at most $o(1)$.*

Proof. Fix any permutation $\pi \in \mathcal{S}_n$. Let g_{rand} be the random function generated in the draw of $g \sim \mathcal{F}_{\text{no}}$. By the triangle inequality,

$$\operatorname{dist}(g, f_\pi) \ge \operatorname{dist}(g_{\text{rand}}, f_\pi) - \operatorname{dist}(g, g_{\text{rand}}).$$

Since $\operatorname{dist}(g, g_{\text{rand}}) \le 2\sum_{i=0}^{n/3} \binom{n}{i}/2^n \le o(1)$, to complete the proof it suffices to fix $\epsilon < \epsilon' < \frac{1}{2}$ and show that $\operatorname{dist}(g_{\text{rand}}, f_\pi) > \epsilon'$ with high probability.

Let $\eta = 1 - 2\epsilon'$. For any $x \in \{0,1\}^n$, $g_{\text{rand}}(x) = f_\pi(x)$ with probability $\frac{1}{2}$, so $\mathbf{E}[\operatorname{dist}(g_{\text{rand}}, f_\pi)] = \frac{1}{2}$. By Chernoff's bound (see, e.g., Appendix A in [8]),

$$\Pr[\operatorname{dist}(g_{\text{rand}}, f_\pi) < \epsilon'] = \Pr[\operatorname{dist}(g_{\text{rand}}, f_\pi) < (1-\eta)\tfrac{1}{2}] \le e^{-2^n \eta^2/6} \le o(\tfrac{1}{n!}).$$

Taking the union bound over all choices of $\pi \in \mathcal{S}_n$ completes the proof. □

Let $\mathcal{T}$ be any deterministic non-adaptive algorithm that attempts to test f-isomorphism with at most $\frac{n}{100}$ queries to an unknown function g. We will show that $\mathcal{T}$ cannot reliably distinguish between the cases where g was drawn from $\mathcal{F}_{\text{yes}}$ or from $\mathcal{F}_{\text{no}}$.

Let $Q \subseteq \{0,1\}^n$ be the set of queries performed by $\mathcal{T}$ on g. We partition the queries in Q in two: the set $Q_b = \{q \in Q : \frac{n}{3} \leq |q| \leq \frac{2n}{3}\}$ of *balanced* queries, and the set $Q_u = Q \setminus Q_b$ of *unbalanced* queries.

When g is drawn from $\mathcal{F}_{\text{yes}}$ or from $\mathcal{F}_{\text{no}}$, the responses to the unbalanced queries Q_u are consistent with some function f_π isomorphic to f. Our next proposition shows that when $\mathcal{T}$ makes only $\frac{n}{100}$ queries to g, then in fact the responses to the unbalanced queries will be consistent with *many* functions isomorphic to f. More precisely, define

$$\Pi_f(g, Q_u) = \{\pi \in \mathcal{S}_n : f_\pi(Q_u) = g(Q_u)\}$$

to be the set of permutations π for which f_π is consistent with the responses to the queries Q_u. The following proposition shows that when the unknown function is drawn from $\mathcal{F}_{\text{yes}}$ or from $\mathcal{F}_{\text{no}}$, then with high probability the set $\Pi_f(g, Q_u)$ is large.

Proposition 9.3. *Let Q_u be any set of unbalanced queries and let g be a function drawn from $\mathcal{F}_{\text{yes}}$ or from $\mathcal{F}_{\text{no}}$. Then for any $0 < t < 1$,*

$$\Pr_g \left[|\Pi_f(g, Q_u)| < t \cdot \tfrac{n!}{2^{|Q_u|}}\right] \leq t.$$

Proof. When $g \sim \mathcal{F}_{\text{yes}}$ or $g \sim \mathcal{F}_{\text{no}}$, then $g(x) = f_\pi(x)$ for every unbalanced input x, where π is chosen uniformly at random from $\mathcal{S}_n$. So it suffices to show that $\Pr_\pi[|\Pi_f(f_\pi, Q_u)| < t \cdot \frac{n!}{2^{|Q_u|}}] \leq t$.

For every $r \in \{0,1\}^{|Q_u|}$, let $S_r \subseteq \mathcal{S}_n$ be the set of permutations σ for which $f_\sigma(Q_u) = r$. A set S_r is *small* if $|S_r| \leq t\frac{n!}{2^{|Q_u|}}$. The union of all small sets covers at most $2^{|Q_u|} \cdot t\frac{n!}{2^{|Q_u|}} = tn!$ permutations, so the probability that a randomly chosen permutation π belongs to a small set is at most t. □

The last proposition showed that when g is drawn from $\mathcal{F}_{\text{yes}}$ or from $\mathcal{F}_{\text{no}}$, then with high probability $\Pi_f(g, Q_u)$ is large; the next lemma shows that conditioned on $\Pi_f(g, Q_u)$ being large, the distribution on the responses to the balanced queries is nearly uniform, even when $g \sim \mathcal{F}_{\text{yes}}$. Specifically, given a function f and a set S of permutations, we define the *discrepancy of f on S* to be

$$\Delta_S(f) = \max_{\substack{Q_b : |Q_b| = \frac{n}{100} \\ r \in \{0,1\}^{|Q_b|}}} \left| \Pr_{\pi \in S}[f_\pi(Q_b) = r] - 2^{-\frac{n}{100}} \right|.$$

We then define the *discrepancy* of f to be

$$\Delta(f) = \max_{\substack{Q_u : |Q_u| = \frac{n}{100} \\ \pi : |\Pi_f(f_\pi, Q_u)| \geq n!/2^{n/50}}} \Delta_{\Pi_f(f_\pi, Q_u)}(f).$$

The following lemma shows that $\Delta(f)$ is small for almost all functions f.

Lemma 9.4. *When f is drawn uniformly at random from the set of functions $\{0,1\}^n \to \{0,1\}$,*

$$\Pr_f \left[\Delta(f) > \tfrac{1}{3} \cdot 2^{-\frac{n}{100}}\right] \leq 2^{-\Omega(2^{n/25})}.$$

We prove Lemma 9.4 in the next section, but first we show how it implies Theorem 9.1.

Theorem 9.1 (Restated)**.** *Fix $0 < \epsilon < \frac{1}{2}$. For a $1 - o(1)$ fraction of the functions $f : \{0,1\}^n \to \{0,1\}$, any non-adaptive algorithm for ϵ-testing isomorphism to f must make at least $\frac{n}{100}$ queries.*

Proof. By Lemma 9.4, with probability at least $1 - 2^{-\Omega(2^{n/25})} = 1 - o(1)$, the discrepancy of a randomly drawn function $f : \{0,1\}^n \to \{0,1\}$ is $\Delta(f) \leq \frac{1}{3} 2^{-\frac{n}{100}}$. Fix f to be any function that satisfies this condition. We will show that testing isomorphism to f requires at least $\frac{n}{100}$ queries.

As discussed earlier, we complete the proof with Yao's Minimax Principle, with the distributions $\mathcal{F}_{\text{yes}}$ and $\mathcal{F}_{\text{no}}$ as defined at the beginning of the section. Let $\mathcal{T}$ be any deterministic non-adaptive algorithm that makes at most $\frac{n}{100}$ queries to the input function g, and let $Q = Q_u \cup Q_b$ represent the queries made by $\mathcal{T}$. Without loss of generality, we can assume $|Q_u| = |Q_b| = \frac{n}{100}$. (If $|Q_b| < \frac{n}{100}$, simply add extra balanced queries to Q_b; this can only help $\mathcal{T}$ determine whether g was drawn from $\mathcal{F}_{\text{yes}}$ or from $\mathcal{F}_{\text{no}}$. Similarly, adding unbalanced queries to Q_u can only help $\mathcal{T}$.)

By Proposition 9.3, the probability that $|\Pi_f(g, Q_u)| < \frac{n!}{2^{n/50}}$ is at most $\frac{1}{2^{n/100}} = o(1)$. Assume, thus, that this event does not happen. Let $\mathcal{R}_{\text{yes}}$ and $\mathcal{R}_{\text{no}}$ be the distribution of the responses to the balanced queries Q_b. Then the total variation distance between $\mathcal{R}_{\text{yes}}$ and $\mathcal{R}_{\text{no}}$ is bounded by

$$\begin{aligned} \mathrm{d}_{\mathrm{TV}}(\mathcal{R}_{\text{yes}}, \mathcal{R}_{\text{no}}) &= \frac{1}{2} \sum_{r \in \{0,1\}^{\frac{n}{100}}} \left| \Pr_{\pi \in \Pi_f(g, Q_u)} [f_\pi(Q_b) = r] - 2^{-\frac{n}{100}} \right| \\ &\leq \tfrac{1}{2} \cdot 2^{\frac{n}{100}} \, \Delta(f) \leq \frac{1}{6}. \end{aligned} \tag{9.1}$$

Therefore, if $\mathcal{T}$ accepts functions drawn from $\mathcal{F}_{\text{yes}}$ with probability at least $\frac{2}{3}$, (9.1) implies that $\mathcal{T}$ also accepts functions drawn from $\mathcal{F}_{\text{no}}$ with probability at least $\frac{2}{3} - \frac{1}{6} = \frac{1}{2}$. But by Proposition 9.2, a function drawn from $\mathcal{F}_{\text{no}}$ is ϵ-far from isomorphic to f with probability $1 - o(1)$, so $\mathcal{T}$ can't be a valid ϵ-tester for isomorphism to f. □

9.2 Proof of Lemma 9.4

The first step in the proof of Lemma 9.4 is to show that for any sufficiently small set Q of balanced queries and sufficiently large set S of permutations, the set $\{\pi(Q)\}_{\pi \in S}$ can be partitioned into a number of large pairwise disjoint sets. The proof of this claim uses the celebrated theorem of Hajnal and Szemerédi [60] introduced in Section 4.2.2.

Lemma 9.5. *Let S be a set of at least $\frac{n!}{2^{n/50}}$ permutations on $[n]$, and let Q_b be a set of at most $\frac{n}{100}$ balanced queries. Then there exists a partition $S_1 \dot\cup \cdots \dot\cup S_k$ of the permutations in S such that for $i = 1, 2, \ldots, k$,*

(i) $|S_i| \geq 2^{n/20}$, and

(ii) The sets $\{\pi(Q_b)\}_{\pi \in S_i}$ are pairwise disjoint.

Proof. Construct a graph G on S where two permutations σ, τ are adjacent iff there exist $u, v \in Q_b$ such that $\sigma(u) = \tau(v)$. By this construction, when T is a set of permutations that form an independent set in G, then $\{\pi(Q_b)\}_{\pi \in T}$ are pairwise disjoint.

Consider a fixed permutation $\sigma \in S$. A second permutation τ is adjacent to σ in G iff there are two vectors u, v in Q_b such that the permutation $\tau\sigma^{-1}$ maps the indices where u has value 1 to the indices where v has value 1 as well. There are $\binom{|Q_b|}{2} \leq (\frac{n}{100})^2$ ways to choose $u, v \in Q_b$ and at most $|u|!(n - |u|)!$ ways to satisfy the mapping condition, so the graph has degree at most

$$\max_{\frac{n}{3} \leq k \leq \frac{2n}{3}} \left(\tfrac{n}{100}\right)^2 \cdot k!\,(n-k)! = \left(\frac{n}{100}\right)^2 \cdot \left(\frac{n}{3}\right)! \left(\frac{2n}{3}\right)! = \left(\frac{n}{100}\right)^2 \cdot \frac{n!}{\binom{n}{n/3}} \leq \frac{n!}{2^{cn}} - 1$$

for a constant $c = 1 - H_2(\frac{1}{3}) - o(1) \geq 0.07$.[1] Therefore, by the Hajnal-Szemerédi Theorem, G can be colored with $n!/2^{0.07n}$ colors, with each color class having size at least $\frac{n!/2^{n/50}}{n!/2^{0.07n}} = 2^{n/20}$. □

[1] $H_2(p)$ represents the binary entropy of p. $H_2(\frac{1}{3}) \approx 0.918$.

Lemma 9.5 is useful because most functions f have low discrepancy on large pairwise disjoint sets.

Lemma 9.6. *Fix Q_b to be a set of $\frac{n}{100}$ balanced queries and fix $r \in \{0,1\}^{\frac{n}{100}}$. Let S be a fixed set of at least $2^{\frac{n}{20}}$ permutations such that the sets $\{\pi(Q_b)\}_{\pi \in S}$ are pairwise disjoint. Then*

$$\Pr_f\left[\left|\Pr_{\pi\in S}[f_\pi(Q_b) = r] - 2^{-\frac{n}{100}}\right| > \tfrac{1}{3}\cdot 2^{-\frac{n}{100}}\right] < 2^{-\Omega(2^{n/25})}.$$

Proof. For every function $f : \{0,1\}^n \to \{0,1\}$ and every permutation π of $[n]$, define the indicator random variable

$$X_{f,\pi} = \begin{cases} 1 & \text{if } f_\pi(Q_b) = r, \\ 0 & \text{otherwise.} \end{cases}$$

When f is chosen uniformly at random from the set of all boolean functions $\{0,1\}^n \to \{0,1\}$, $\mathbf{E}_f[X_{f,\pi}] = \Pr_f[f_\pi(Q_b) = r] = 2^{-\frac{n}{100}}$, so

$$\mathop{\mathbf{E}}_f\left[\Pr_{\pi\in S}[f_\pi(Q_b) = r]\right] = \frac{1}{|S|}\sum_{\pi\in S}\mathop{\mathbf{E}}_f[X_{f,\pi}] = 2^{-\frac{n}{100}}.$$

Furthermore, the pairwise disjointness property of S guarantees that the indicator variables $X_{f,\pi}$ are pairwise independent. Therefore, by Chernoff's bound,

$$\Pr_f\left[\left|\Pr_{\pi\in S}[f_\pi(Q_b) = r] - 2^{-\frac{n}{100}}\right| > \tfrac{1}{3}\cdot 2^{-\frac{n}{100}}\right] < e^{-\Omega(|S|2^{-n/100})}. \qquad \square$$

The proof of Lemma 9.4 can now be completed as follows.

Lemma 9.4 (Restated)**.** *When f is drawn uniformly at random from the set of functions $\{0,1\}^n \to \{0,1\}$,*

$$\Pr_f\left[\Delta(f) > \tfrac{1}{3}\cdot 2^{-\frac{n}{100}}\right] \leq 2^{-\Omega(2^{n/25})}.$$

Proof. Fix a permutation π and a set Q_u of $\frac{n}{100}$ unbalanced queries such that $|\Pi_f(f_\pi, Q_u)| \geq \frac{n!}{2^{n/50}}$. Let $S = \Pi_f(f_\pi, Q_u)$, and fix a set Q_b of $\frac{n}{100}$ balanced queries.

By Lemma 9.5, there exists a partition $S_1\dot\cup\cdots\dot\cup S_k$ of S such that for each part S_i, $|S_i| \geq 2^{n/20}$ and $\{\pi(Q_b)\}_{\pi\in S_i}$ are pairwise disjoint. By Lemma 9.6, for every set S_i in the partition,

$$\Pr_f\left[\left|\Pr_{\pi\in S_i}[f_\pi(Q_b) = r] - 2^{-\frac{n}{100}}\right| > \tfrac{1}{3}\cdot 2^{-\frac{n}{100}}\right] \leq 2^{-\Omega(2^{n/25})}.$$

Taking the union bound over all $k < n!$ sets S_i, we get that

$$\Pr_f \left[\left| \Pr_{\pi \in S}[f_\pi(Q_b) = r] - 2^{-\frac{n}{100}} \right| > \tfrac{1}{3} \cdot 2^{-\frac{n}{100}} \right] < n! \cdot 2^{-\Omega(2^{n/25})}.$$

Applying a union bound once again, this time over all $\binom{2^n}{n/100} < 2^{\frac{n^2}{100}}$ choices of Q_b and $2^{\frac{n}{100}}$ choices for r, we obtain

$$\Pr_f \left[\Delta_S(f) > \tfrac{1}{3} \cdot 2^{-\frac{n}{100}} \right] < 2^{\frac{n^2}{100} + \frac{n}{100}} \cdot n! \cdot 2^{-\Omega(2^{n/25})}.$$

Finally, applying the union bound one last time over the $n!$ choices for π and $\binom{2^n}{n/100} \leq 2^{\frac{n^2}{100}}$ choices for Q_u, we get

$$\Pr_f \left[\Delta(f) > \tfrac{1}{3} \cdot 2^{-\frac{n}{100}} \right] < 2^{\frac{2n^2}{100} + \frac{n}{100}} \cdot n!^2 \cdot 2^{-\Omega(2^{n/25})} = 2^{-\Omega(2^{n/25})}. \qquad \square$$

9.3 Notes and Discussion

Testing isomorphism to juntas. By combining the result in this chapter with independent and overlapping results of Chakraborty, García Soriano, and Matsliah [40], we obtain a generalization of Theorem 9.1. Specifically, we obtain a lower bound showing that for *almost all* functions f that are k-juntas, testing f-isomorphism requires at least $\Omega(k)$ queries. The details of this result are found in [4].

One implication of the generalized statement is that the upper bound of $O(k \log k)$ queries in Theorem 8.1 on the query complexity for testing isomorphism to $(n-k)$-symmetric functions cannot be improved by more than a logarithmic factor since there are (many) $(n-k)$-symmetric functions that require $\Omega(k)$ queries to test.

Optimality of the lower bound. There is a logarithmic gap between the lower bound of Theorem 9.1 and the universal upper bound described at the beginning of the last chapter. It is quite possible that the gap is only a byproduct of the limitations of our proof argument and that $\Theta(n \log n)$ queries are indeed required to test f-isomorphism for almost every function $f : \{0,1\}^n \to \{0,1\}$. That problem remains open.

Testing isomorphism to linear functions. The result in this chapter is non-constructive: while it states that for *almost all* functions f, testing f-isomorphism requires at least $\Omega(n)$ queries, it does not identify any *concrete* functions for which this lower bound applies.

Our result in Chapter 7 provides an example of a concrete function for which the same lower bound applies. Consider $f(x) = x_1 \oplus \cdots \oplus x_{\frac{n}{2}}$. The set of functions that are isomorphic to f is exactly the set of $\frac{n}{2}$-linear functions. Therefore, Theorem 7.1 implies that testing f-isomorphism in this case requires $\frac{n}{2} - o(n)$ queries.

Chapter 10

Testing Isomorphism to Juntas

The first two chapters of this part of the thesis gave some partial results related to the characterization of the set of functions f for which it is possible to test f-isomorphism with a constant number of queries. Chapter 8's main result was a sufficient condition for efficient isomorphism-testability: all partially symmetric functions are efficiently isomorphism-testable.

The main result of Chapter 9 gave a strong lower bound in the sense that the set of functions that are efficiently isomorphism-testable (or, indeed, that can be tested with $o(n)$ queries) contains only a $o(1)$ fraction of all boolean functions. Being non-constructive, however, that result did not identify specific characteristics of the set of functions that are not efficiently isomorphism-testable. The goal of this chapter is to establish such a characteristic. In other words, we want to identify a concrete class of functions that are not efficiently isomorphism-testable.

As we mentioned at the beginning of the last chapter, until recently the only class of functions that were known to not be efficiently isomorphism-testable was the set of k-linear functions for $\omega(1) \leq k \leq o(\sqrt{n})$ [52]. Fischer et al. conjectured that these functions were all contained in a much larger class of functions that are not efficiently isomorphism-testable. Specifically, they conjectured that if n is sufficiently large compared to k and $f : \{0,1\}^n \to \{0,1\}$ is a k-junta that is ϵ-far from all $(k-1)$-juntas, then any ϵ-tester for f-isomorphism requires a number of queries that depends on k.

The main result of this chapter confirms Fischer et al.'s conjecture. In fact, we do more: we show that for every function f that is a k-junta and is far from all $(k-e)$-juntas for some $\omega(1) \leq k \leq n - \omega(1)$ and $e = o(\sqrt{k})$, testing f-isomorphism requires a super-constant number of queries.

Theorem 10.1. *Let $g : \{0,1\}^n \to \{0,1\}$ be a k-junta which is ϵ-far from being a $(k-e)$-junta for some $e \geq 1$. Then any non-adaptive ϵ-tester for g-isomorphism must make at least* $\log_2(k'/e^2) - O(1)$ *queries, where $k' = \min(k, n-k)$.*

The rest of this chapter is dedicated to proving this theorem. On first reading, the reader is encouraged to focus on the simplest case, where $e = 1$.

10.1 Two distributions on functions

We prove Theorem 10.1 with Yao's Minimax Principle [100] via Lemma 3.15. To do so, we must introduce distributions $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$ on functions that are isomorphic to f and ϵ-far from isomorphic to f, respectively, for some fixed function f that satisfies the conditions of the theorem.

The main challenge in the construction of our distributions is that it must apply to a large class of functions. That is, we cannot use any other structural property of f *except* that it is a k-junta and is far from $(k-e)$-juntas. In fact, this restriction suggests a natural way to define $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$.

In the following, fix $0 < e < k < n$ and let f be a k-junta that is also ϵ-far from all $(k-e)$-juntas. Without loss of generality, we may assume that the k relevant coordinates of f are $\{1, 2, \ldots, k\}$. Let $f_{\text{core}} : \{0,1\}^k \to \{0,1\}$ be the restriction of f to these coordinates.

The distribution $\mathcal{D}_{\text{yes}}$ is defined in the most natural way, by randomly embedding f_{core} into $[n]$. More precisely, to draw $g \sim \mathcal{D}_{\text{yes}}$, we first draw a random subset $J \subseteq [n]$ uniformly at random from all subsets of $[n]$ of size k. We then draw a random bijection $\sigma : [k] \to J$ uniformly at random. Finally, we define $g(x) := f_{\text{core}}(x_{\sigma(1)}, \ldots, x_{\sigma(k)})$. This construction guarantees that every function g in the support of $\mathcal{D}_{\text{yes}}$ is isomorphic to f.

The distribution $\mathcal{D}_{\text{no}}$ is defined in a similar way. To draw $g \sim \mathcal{D}_{\text{no}}$, we begin by drawing a set $J \subseteq [n]$ uniformly at random from all subsets of $[n]$ of size $k - e$. We then draw a random map $\sigma : [k] \to J$ uniformly at random among all the maps that satisfy the following conditions:

1. There exists a single element $j^* \in J$ such that $|\{i \in [k] : \sigma(i) = j^*\}| = e + 1$.
2. For every $j \in J \setminus \{j^*\}$, $|\{i \in [k] : \sigma(i) = j\}| = 1$.

We then define $g(x) = f_{\text{core}}(x_{\sigma(1)}, \ldots, x_{\sigma(k)})$. This construction guarantees that every function g in the support of $\mathcal{D}_{\text{no}}$ is a $(k-e)$-junta. As a result, every such g is ϵ-far from isomorphic to f.

To complete the proof of Theorem 10.1, we show below that for any set of $q = \log(k'/e^2) - O(1)$ queries, the distributions on the responses obtained from functions drawn from $\mathcal{D}_{\text{yes}}$ or from $\mathcal{D}_{\text{no}}$ are very similar.

10.2 Distance between multivariate hypergeometrics

The typical way to prove a property testing bound such as Theorem 10.1 is as follows. First, we write the q queries of tester $\mathcal{T}$ as $x^1, \ldots, x^q \in \{0,1\}^n$. We then introduce the *response vector* random variables R_{yes} and R_{no}. Here $R_{\text{yes}} \in \{0,1\}^q$ is defined by drawing $f_{\text{yes}} \sim \mathcal{F}_{\text{yes}}$ and letting $R_{\text{yes}} = \langle f_{\text{yes}}(x^1), \ldots, f_{\text{yes}}(x^q)\rangle$, and R_{no} is defined analogously. Finally, we show that

$$\mathrm{d}_{\mathrm{TV}}(R_{\text{yes}}, R_{\text{no}}) \leq 2^q \cdot \frac{O(e^2)}{k'} + .01. \tag{10.1}$$

We will in fact prove a stronger statement. To understand it, let's reconsider the complete random processes P_{yes} and P_{no} by which the response vectors R_{yes} and R_{no} are generated. We begin by focusing on the "yes" process, P_{yes}.

Given the tester $\mathcal{T}$'s queries $x^1, \ldots, x^q \in \{0,1\}^n$, we think of them as row vectors and arrange them into a $q \times n$ *query matrix* Q. We will be especially interested in the *columns* of this matrix Q, the jth column consisting of the jth bits of all the query strings. Abstractly, we define the set of all possible column (types)

$$\mathfrak{C} = \{0,1\}^q.$$

Since $|\mathfrak{C}| = 2^q \ll n$, some columns will occur many times in the matrix Q. In fact, we will think of the query matrix Q as being an ordered *multiset* of columns from $\mathfrak{C}$.

Recalling the definition of $\mathcal{F}_{\text{yes}}$, we think of the first step of P_{yes} as choosing k column indices $j_1, \ldots, j_k$ randomly and without replacement from $[n]$. We next extract columns $j_1, \ldots, j_k$ from Q. We view this as a multiset of columns, and call it the *argument multiset* S_{yes}. Next, we randomly order the columns in S_{yes}, forming a $q \times k$ *argument matrix* A_{yes}. Finally, we produce the response vector R_{yes} by applying g_{core} to the argument matrix, row-wise.

The reader can easily verify this process P_{yes} generates the correct distribution on the response vector random variable R_{yes}.

The "no" process P_{no} is very similar, differing only in the way it generates the argument multiset from the query matrix. Recalling the definition of $\mathcal{F}_{\text{no}}$, we think of P_{no} as forming the argument multiset S_{no} by choosing $\ell = k - e$ random columns from Q

without replacement, and including an *additional* e copies of the first-chosen column. The process P_{no} then forms the argument matrix A_{no} by again randomly ordering the columns in the argument multiset, and finally produces the response rector R_{no} again by applying g_{core} to A_{no}, row-wise. The reader can again easily verify that P_{no} generates the correct distribution on R_{no}.

Because the processes are identical after the argument multiset is formed, a coupling argument immediately implies that

$$\mathrm{d}_{\mathrm{TV}}(R_{\text{yes}}, R_{\text{no}}) \leq \mathrm{d}_{\mathrm{TV}}(S_{\text{yes}}, S_{\text{no}}). \tag{10.2}$$

This inequality can be extremely lossy, depending on the function g_{core}. However, since Theorem 10.1 applies for an extremely broad range of functions, we are almost forced to design a proof of Theorem 10.1 that *uses no properties of the function* g_{core}. That is, in the absence of additional restrictions on the class of functions considered, there is no obvious way to bound $\mathrm{d}_{\mathrm{TV}}(R_{\text{yes}}, R_{\text{no}})$ except by $\mathrm{d}_{\mathrm{TV}}(S_{\text{yes}}, S_{\text{no}})$.

Letting $\mathcal{S}_{\text{yes}}$ denote the subprocess of P_{yes} generating S_{yes}, and similarly for $\mathcal{S}_{\text{no}}$, we have reduced proving (10.1), and hence Theorem 10.1, to the following:

Theorem 10.2. *For* $S_{\text{yes}} \sim \mathcal{S}_{\text{yes}}, S_{\text{no}} \sim \mathcal{S}_{\text{no}}$*, we have*

$$\mathrm{d}_{\mathrm{TV}}(S_{\text{yes}}, S_{\text{no}}) \leq |\mathfrak{C}| \cdot \frac{O(e^2)}{\min(k, n-k)} + .01.$$

The reader can see now why our query complexity lower bound in Theorem 10.1 is only logarithmic; we have $|\mathfrak{C}| = 2^q$ competing against $\frac{1}{k}$ in the above bound. Indeed, we can never prove a better-than-logarithmic lower bound if our proof only involves showing statistical closeness of the argument multisets S_{yes} and S_{no}. To see this, suppose $k = n/2$, so $n - k = n/2$ as well. Then if $2^q \gg n/2$, it is possible that every column in the query matrix is unique. In this case, the total variation distance between argument multisets S_{yes} and S_{no} will be 1 even in the case $e = 1$, because S_{yes} will always consist of unique columns, whereas S_{no} will always have one column duplicated.

Notice that the ordering of the columns in the query matrix Q has proven to be unimportant; we can think of Q simply as an unordered multiset of columns from $\mathfrak{C}$. Thus Theorem 10.2 is really a statement about the total variation distance between certain multivariate hypergeometric random variables. Specifically, for each column $\mathfrak{c} \in \mathfrak{C}$, let $m(\mathfrak{c})$ denote the number of copies of $\mathfrak{c}$ in Q. In process $\mathcal{S}_{\text{yes}}$, we choose k random columns from Q without replacement and count the number of copies of each column (type) in the draw. Process $\mathcal{S}_{\text{no}}$ is similar, except we choose ℓ random columns from Q without replacement, and count an extra e copies of the first-drawn column.

10.2.1 Reduction of Theorem 10.2 to two lemmas

This preceding discussion motivates the following notation:

Definition 10.3. Given integers $N, e \geq 1$, $M, L \geq 0$, with $M, L + e \leq N$, we define $\lambda_{N,M,L}(e) = \mathrm{d}_{\mathrm{TV}}(X, Y)$, where $X \sim \mathcal{H}_{N,M,L+e}$ and $Y \sim \mathcal{H}_{N,M,L} + e$.

The proof of Theorem 10.2 relies on the following two lemmas. The first lemma is relatively straightforward, and relates the distance between $\mathcal{S}_{\mathrm{yes}}$ and $\mathcal{S}_{\mathrm{no}}$ to the total variation distance between hypergeometric distributions.

Lemma 10.4.

$$\mathrm{d}_{\mathrm{TV}}(S_{\mathrm{yes}}, S_{\mathrm{no}}) \leq \sum_{\mathfrak{c} \in \mathfrak{C} : m(\mathfrak{c}) \neq 0} \frac{m(\mathfrak{c})}{n} \cdot \lambda_{n-1, m(\mathfrak{c})-1, \ell-1}(e).$$

The second lemma is a total variation distance bound between (univariate) hypergeometric random variables which may be of independent interest.

Lemma 10.5. *There is a universal constant $2 \leq \kappa < \infty$ such that for any N, M, L, if $L' = \min(L, N-L)$ satisfies $\frac{ML'}{N} \geq \kappa e^2$, then $\lambda_{N,M,L}(e) \leq .01$.*

We briefly comment on why the hypothesis $\frac{ML'}{N} \gg e^2$ is necessary to show that $\mathcal{H}_{N,M,L+e}$ and $\mathcal{H}_{N,M,L} + e$ are close in total variation distance. For simplicity, first suppose that $e = 1$. It is necessary that that $\frac{ML}{N} \gg 1$; this quantity is the mean of $\mathcal{H}_{N,M,L}$, and if it is $\ll 1$ then $X \sim \mathcal{H}_{N,M,L+1}$ is likely to be 0 whereas $Y \sim \mathcal{H}_{N,M,L} + 1$ is at least 1. Second, it is also necessary that $\frac{M(N-L)}{N} = M(1 - \frac{L}{N}) \gg 1$. To see this, note that if by way of contrast $1 - \frac{L}{N} \ll \frac{1}{M}$, then X is concentrated at M and Y is concentrated at $M + 1$. Finally, to understand the hypothesis's dependence on e, suppose $M = N/2$ and L is quite small. Then $\mathcal{H}_{N,M,L}$ is distributed very much like $\mathrm{Bin}(L, e)$; hence we require $L \gg e^2$ or else the extra $+e$ in Y will dominate the standard deviation of $\mathrm{Bin}(L, e)$.

We prove Lemmas 10.4 and 10.5 in the next sections, but first we show how Theorem 10.2 follows from the lemmas.

Proof of Theorem 10.2. Note that we may freely assume $k \geq 2e + 2$, as otherwise the bound we are trying to prove exceeds 1 (assuming the constant in the $O(\cdot)$ is large enough). Let us introduce the notation $N = n-1$, $M(\mathfrak{c}) = m(\mathfrak{c})-1$, $L = \ell-1$, $L' = \min(L, N-L)$.

Then by Lemma 10.4,

$$\begin{aligned}
\mathrm{d}_{\mathrm{TV}}(\mathcal{S}_{\text{yes}}, \mathcal{S}_{\text{no}}) &\leq \sum_{\mathfrak{c}\in\mathfrak{C}: m(\mathfrak{c})\neq 0} \frac{m(\mathfrak{c})}{n} \cdot \lambda_{n-1, m(\mathfrak{c})-1, \ell-1}(e) \\
&= \sum_{0 \leq \frac{M(\mathfrak{c})}{N} < \frac{\kappa e^2}{L'}} \frac{m(\mathfrak{c})}{n} \cdot \lambda_{N, M(\mathfrak{c}), L}(e) \\
&\quad + \sum_{\frac{M(\mathfrak{c})}{N} \geq \frac{\kappa e^2}{L'}} \frac{m(\mathfrak{c})}{n} \cdot \lambda_{N, M(\mathfrak{c}), L}(e) \\
&\leq \sum_{0 \leq \frac{M(\mathfrak{c})}{N} < \frac{\kappa e^2}{L'}} \tfrac{m(\mathfrak{c})}{n} + \sum_{\frac{M(\mathfrak{c})}{N} \geq \frac{\kappa e^2}{L'}} \tfrac{m(\mathfrak{c})}{n} \cdot .01,
\end{aligned}$$

where the last inequality uses Lemma 10.5. Since $\sum_{\mathfrak{c}\in\mathfrak{C}} m(\mathfrak{c}) = n$, the second sum above is at most .01. Thus it remains to bound the first sum by $|\mathfrak{C}| \frac{O(e^2)}{\min(k, n-k)}$. There are at most $|\mathfrak{C}|$ summands in this first sum, and for each we have

$$\frac{m(\mathfrak{c})}{n} \leq \frac{M(\mathfrak{c})+1}{N} \leq \frac{\kappa e^2}{L'} + \frac{1}{N} \leq \frac{2\kappa e^2}{L'}$$

by the condition of the sum.

To complete the proof, it remains to show that $L' = \min(\ell - 1, n - \ell) \geq \Omega(\min(k, n - k))$. When $\ell - 1 \leq n - \ell$, then $L' = \ell - 1 = k - e - 1 \geq k/2$ by the fact that $k \geq 2e + 2$. And when $n - \ell < \ell - 1$, then $L' = n - \ell = n - k + e \geq n - k$. so $L' \geq \frac{1}{2}\min(k, n - k)$, as we wanted to show. □

10.2.2 Proof of Lemma 10.4

Let us think of the experiment $\mathcal{S}_{\text{yes}}$ in an alternate way. We begin by choosing a first column from Q for S_{yes} — call it C_1. We next decide how many additional copies of C_1 to include into S_{yes}. Call this quantity T. We have

$$T \mid (C_1 = \mathfrak{c}) \sim \mathcal{H}_{n-1, m(\mathfrak{c})-1, k-1}.$$

(Note that $m(\mathfrak{c}) - 1 \geq 0$ always, because $\mathfrak{c}$ won't be chosen if $m(\mathfrak{c}) = 0$.) So far, S_{yes} consists of $T+1$ copies of C_1. Finally, we complete the draw of S_{yes} by choosing $k-(T+1)$ columns without replacement from "$Q \setminus C_1$", meaning the multiset of columns formed from Q by removing all copies of C_1.

We think of the experiment $\mathcal{S}_{\text{no}}$ in a similar way. Again, we begin by choosing a first column C_1 from Q for S_{no}. We next determine how many additional copies of C_1 there will be from among the remaining $\ell - 1$ choices. Calling this quantity U, we have

$$U \mid (C_1 = \mathfrak{c}) \sim \mathcal{H}_{n-1,m(\mathfrak{c})-1,\ell-1}.$$

Recall, however, that in $\mathcal{S}_{\text{no}}$, we include an additional e copies of C_1 into S_{no}. Hence S_{no} ends up with $U + e + 1$ copies of C_1. Finally, we complete S_{no} by adding $\ell - (U + 1)$ columns drawn without replacement from $Q \setminus C_1$.

Let $V = U + e$. We claim that by coupling the random variables $T \mid (C_1 = \mathfrak{c})$ and $V \mid (C_1 = \mathfrak{c})$, we couple S_{yes} and S_{no}. This follows immediately from the two descriptions, as then $T + 1 = V + 1 = U + e + 1$, and $k - (T + 1) = \ell + e - (V + 1) = \ell - (U + 1)$. Hence

$$\mathrm{d}_{\mathrm{TV}}(S_{\text{yes}}, S_{\text{no}}) \leq \sum_{\mathfrak{c} \in \mathfrak{C}} \Pr[C_1 = \mathfrak{c}] \cdot \mathrm{d}_{\mathrm{TV}}(T \mid (C_1 = \mathfrak{c}), V \mid (C_1 = \mathfrak{c})).$$

On one hand, $\Pr[C_1 = \mathfrak{c}]$ is simply $\frac{m(\mathfrak{c})}{n}$. On the other hand, we have

$$\begin{aligned} T \mid (C_1 = \mathfrak{c}) &\sim \mathcal{H}_{n-1,m(\mathfrak{c})-1,\ell+e-1}, \\ V \mid (C_1 = \mathfrak{c}) &\sim \mathcal{H}_{n-1,m(\mathfrak{c})-1,\ell-1} + e. \end{aligned}$$

So by definition, $\mathrm{d}_{\mathrm{TV}}(T \mid (C_1 = \mathfrak{c}), V \mid (C_1 = \mathfrak{c})) = \lambda_{n-1,m(\mathfrak{c})-1,\ell-1}(e)$, and hence

$$\mathrm{d}_{\mathrm{TV}}(S_{\text{yes}}, S_{\text{no}}) \leq \sum_{\mathfrak{c} \in \mathfrak{C} : m(\mathfrak{c}) \neq 0} \frac{m(\mathfrak{c})}{n} \cdot \lambda_{n-1,m(\mathfrak{c})-1,\ell-1}(e),$$

as claimed.

10.2.3 Proof of Lemma 10.5

Recall that $L' = \min(L, N - L)$,

$$\frac{ML'}{N} \geq \kappa e^2, \tag{10.3}$$

and our goal is to bound $\lambda_{N,M,L}(e) = \mathrm{d}_{\mathrm{TV}}(X, Y) \leq .01$, where $X \sim \mathcal{H}_{N,M,L+e}$ and $Y \sim \mathcal{H}_{N,M,L} + e$.

We begin by coupling X and Y, as follows. Imagine drawing balls randomly and without replacement from an urn containing N balls, M of which are white. We draw

$L+e$ balls from the urn. We let X be the number of white balls among all balls drawn; we let Y be the number of white balls among the first L balls drawn, plus e. Note that $X \le Y$ always under this coupling.

Let us now compare the probability mass functions of X and Y. The integers $u < e$ can be in X's range but not Y's; the integers $u > \min(M, L+e)$ can be in Y's range but not X's. The remaining integers are in the range of both X and Y, and we have

$$\begin{aligned}\frac{\Pr[X=u]}{\Pr[Y=u]} &= \frac{\binom{M}{u}\binom{N-M}{L+e-u}}{\binom{N}{L+e}} \Big/ \frac{\binom{M}{u-e}\binom{N-M}{L+e-u}}{\binom{N}{L}} \\ &= \frac{\binom{M}{u}}{\binom{M}{u-e}} \cdot \frac{\binom{N}{L}}{\binom{N}{L+e}} \\ &= \tfrac{(M-u+e)(M-u+e-1)\cdots(M-u+1)}{u(u-1)\cdots(u-e+1)} \cdot \tfrac{\binom{N}{L}}{\binom{N}{L+e}}.\end{aligned}$$

Evidently (and unsurprisingly), this ratio is a decreasing function of u. Letting t be the largest integer for which the ratio is at least 1, we conclude that

$$\Pr[X=u] \ge \Pr[Y=u] \text{ iff } u \le t.$$

It follows immediately that

$$\mathrm{d_{TV}}(X,Y) = \Pr[X \le t] - \Pr[Y \le t].$$

But by our coupling,

$$\begin{aligned}\Pr[X \le t] - \Pr[Y \le t] &= \Pr[X \le t \cap Y > t] \\ &\qquad - \Pr[X > t \cap Y \le t] \\ &= \Pr[X \le t \cap Y > t],\end{aligned}$$

since $X \le Y$ always. Our goal, then, is to bound

$$\mathrm{d_{TV}}(X,Y) = \Pr[X \le t \cap Y > t]. \tag{10.4}$$

We will in fact prove something slightly stronger: we will show that for *any* value of t, the right-hand side of (10.4) is small.

To analyze (10.4) we recall the ball and urn process defining X and Y. Having drawn $L+e$ balls, let W be the number of white balls among the *last* e balls drawn, and let Z be the number of white balls among the first L. Thus $X = W + Z$ and $Y = e + Z$. As a first observation, we may note that if $W = e$ then $X = Y$ and hence the event in (10.4) does not occur. I.e.,

$$\mathrm{d_{TV}}(X, Y) \leq \Pr[W \neq e] \leq e(1 - \tfrac{M}{N}), \tag{10.5}$$

where we used a union bound over each of the last e balls being non-white. Now by (10.3),

$$e \leq \sqrt{\frac{1}{\kappa}\frac{ML'}{N}} \leq \sqrt{\frac{L'}{\kappa}} \leq .001\sqrt{N}, \tag{10.6}$$

if we assume κ large enough. It follows that we may additionally assume

$$M \leq N - .01\sqrt{N} \qquad \Leftrightarrow \qquad 1 - \frac{M}{N} \geq \frac{.01}{\sqrt{N}} \tag{10.7}$$

because otherwise the bound in (10.5) is at most $.001\sqrt{N} \cdot \frac{.01}{\sqrt{N}} = .00001$, which establishes the theorem with room to spare. We also use this opportunity to mention that

$$M \geq 2e, \;\; L' \geq 2e, \;\; \text{(and hence certainly } N \geq 2e) \tag{10.8}$$

follow easily from (10.3).

We next give a more refined upper bound on (10.4). By conditioning on W we have

$$\begin{aligned} \mathrm{d_{TV}}(X, Y) &= \Pr[X \leq t \cap Y > t] \\ &= \sum_{i=0}^{e-1} \Pr[W = i] \Pr[t - e < Z \leq t - i \mid W = i]. \end{aligned}$$

Now $Z \mid (W = i)$ has distribution $\mathcal{H}_{N-e, M-i, L}$ (and note that $M - i \geq M - e \geq 0$ by (10.8)). Let us write $\sigma^2 = L(1 - \frac{L}{N-e})\frac{M-i}{N-e}(1 - \frac{M-i}{N-e})$. Applying Corollary **??** and a union bound we get

$$\begin{aligned} \mathrm{d_{TV}}(X, Y) &\leq \sum_{i=0}^{e-1} \Pr[W = i] \cdot (e - i)\frac{C}{\sigma} \\ &\leq \max_{0 \leq i < e}\left\{\frac{C}{\sigma}\right\} \cdot \sum_{i=0}^{e-1} \Pr[W = i](e - i) \\ &= \max_{0 \leq i < e}\left\{\frac{C}{\sigma}\right\} \cdot \mathbf{E}[e - W]. \end{aligned}$$

We have $W \sim \mathcal{H}_{N,M,e}$, and thus $\mathbf{E}[e - W] = e(1 - \frac{M}{N})$. And by definition,

$$\max_{0 \le i < e} \left\{ \frac{C}{\sigma} \right\} = \max_{0 \le i < e} \left\{ \frac{C}{\sqrt{L(1 - \frac{L}{N-e})(\frac{M-i}{N-e})(1 - \frac{M-i}{N-e})}} \right\}$$
$$\le \frac{C}{\sqrt{L(1 - \frac{L}{N-e})(\frac{M-e}{N-e})(1 - \frac{M}{N-e})}}.$$

Thus we have established

$$\mathrm{d}_{\mathrm{TV}}(X, Y) \le \frac{Ce}{\sqrt{L(1 - \frac{L}{N-e})}} \cdot \frac{1 - \frac{M}{N}}{\sqrt{1 - \frac{M}{N-e}}} \cdot \frac{1}{\sqrt{\frac{M-e}{N-e}}}. \tag{10.9}$$

We will bound the three fractions in (10.9) one at a time. We begin with the middle one. Note first that

$$\frac{d}{dM} \left(\frac{1 - \frac{M}{N}}{\sqrt{1 - \frac{M}{N-e}}} \right) = -\frac{N - 2e - M}{2N\sqrt{1 - \frac{M}{N-e}}(N - e - M)}.$$

By combining (10.6) and (10.7) we get $M \le N - 10e < N - 2e$. Hence the derivative above is always negative, implying that $(1 - \frac{M}{N})/\sqrt{1 - \frac{M}{N-e}}$ is a decreasing function of M on M's range. Hence we may upper-bound this fraction by taking $M = 0$, giving an upper bound of 1. Substituting this into (10.9) gives

$$\mathrm{d}_{\mathrm{TV}}(X, Y) \le \frac{Ce}{\sqrt{L(1 - \frac{L}{N-e})}} \cdot \frac{1}{\sqrt{\frac{M-e}{N-e}}}. \tag{10.10}$$

We next examine the fraction on the right. It is at most

$$\frac{1}{\sqrt{\frac{M-e}{N}}} \le \frac{1}{\sqrt{\frac{M/2}{N}}} = \sqrt{\frac{2N}{M}},$$

where we used (10.8). By virtue of (10.3), we can upper-bound this by $\sqrt{\frac{2}{\kappa}} \cdot \frac{\sqrt{L'}}{e}$. Substituting this upper bound into (10.10) yields

$$\begin{aligned}\mathrm{d}_{\mathrm{TV}}(X,Y) &\le C\sqrt{\frac{2}{\kappa}}\cdot\sqrt{\frac{L'}{L(1-\frac{L}{N-e})}}\\ &\le .001\sqrt{\frac{L'}{L(1-\frac{L}{N-e})}}, \qquad (10.11)\end{aligned}$$

assuming κ is sufficiently large compared with C.

Finally, we split into two cases, depending on whether $L \le N/2$. If indeed $L \le N/2$, then $L' = L$ and we have

$$.001\sqrt{\frac{L'}{L(1-\frac{L}{N-e})}} = \frac{.001}{\sqrt{1-\frac{L}{N-e}}} \le \frac{.001}{\sqrt{1-\frac{N/2}{N-e}}}.$$

But $N - e \ge N - .001\sqrt{N} \ge (2/3)N$ (using (10.6) and $N \ge 2$ from (10.8)), so we upper-bound

$$\mathrm{d}_{\mathrm{TV}}(X,Y) \le \frac{.001}{\sqrt{1-\frac{N/2}{(2/3)N}}} = .002 \le .01,$$

as needed. The second case is that $L \ge N/2$, in which case $L = N - L'$ and the bound in (10.11) is

$$\begin{aligned}.001\sqrt{\frac{L'}{(N-L')(1-\frac{N-L'}{N-e})}} &= .001\sqrt{\frac{L'}{(N-L')\frac{L'-e}{N-e}}}\\ &= .001\sqrt{\frac{L'}{L'-e}}\sqrt{\frac{N-e}{N-L'}}. \qquad (10.12)\end{aligned}$$

But using (10.8),

$$\sqrt{\frac{L'}{L'-e}} \le \sqrt{\frac{L'}{L'/2}} = \sqrt{2},$$

and using $L' \le N/2$,

$$\sqrt{\frac{N-e}{N-L'}} \le \sqrt{\frac{N}{N-L'}} \le \sqrt{\frac{N}{N/2}} = \sqrt{2}.$$

Hence the upper bound (10.12) on $\mathrm{d}_{\mathrm{TV}}(X,Y)$ is at most $.001\sqrt{2}\sqrt{2} < .01$, as needed.

This completes the proof of Lemma 10.5.

10.3 Notes and Discussion

Majority functions. The lower bound in this chapter applies to most k-juntas that we consider to "strongly depend" on all k of their relevant variables. One notable exception to this observation, however, is the class of majority functions on exactly k variables. This function is symmetric, so intuitively it "strongly depends" on all k of its relevant variables. Majority functions on k variables, however, are $o(1)$-close to Majority functions on $k - e$ variables for any $e = o(k)$. As a result, the lower bound of this chapter does not apply to the problem of testing isomorphism to the k-majority functions.

As a result of this perceived gap, in [25] we also proved a lower bound for testing isomorphism to k-majority functions. In fact, the bound obtained in this problem is stronger than the one in this chapter and shows that $\mathrm{poly}(k)$ queries are required to non-adaptively test isomorphism to k-majority functions.

Generalizing the lower bound. At first glance, the reader may wonder why we require the two conditions that f be a k-junta *and* that f be far from $(k-e)$-juntas for some small e in the statement of Theorem 10.1. A natural question to ask is whether the theorem can be generalized by removing one of these two conditions. It cannot. To see this, consider the constant function and the parity function. The constant function is a k-junta for any $0 \leq k \leq n$ and the parity function is far from all $(k - e)$-juntas for any $k - e < n$. But both functions are symmetric so testing isomorphism to either of them can be done with a constant number of queries.

It is possible, however, that Theorem 10.1 may be generalized by establishing a lower bound on the query complexity for testing isomorphism to all functions that are far from all k*-symmetric* functions. In [26], we conjecture that this is indeed the case, but the problem remains open. Note that if this result were established, that lower bound combined with the result presented in Chapter 8 would essentially characterize the set of functions that are efficiently isomorphism-testable.

Part III

Connections

Chapter 11

Communication Complexity

Communication complexity is an area of theoretical computer science that has developed techniques which have been spectacularly effective in proving lower bounds in a large variety of other areas of computer science. In this chapter, we will explore how communication complexity can also be used to prove strong lower bounds on the query complexity of different property testing problems. As a bonus, we will also see how the resulting proofs are often surprisingly simple.

The basic setup in communication complexity is straight-forward. We have two players, Alice and Bob, who each receive some input. They cannot see each other's inputs, but would like to compute some function on their joint input. For example, Alice may receive a set $A \subseteq [n]$, Bob may receive a set $B \subseteq [n]$, and they might want to determine if their sets intersect or if the two sets are disjoint. They do so by communicating some information to each other (following a pre-determined protocol), until they have the desired answer. The main goal in communication is to determine the *minimum* number of bits that Alice and Bob must communicate to each other to compute the answer.

The versatility of communication complexity in establishing lower bounds in various areas of computer science stems mainly from two useful properties. The first is that there are very strong lower bounds on the communication complexity required to compute some basic functions. For example, it is known that even when Alice and Bob run a randomized protocol and have access to a common source of randomness, solving the set disjointness problem mentioned above requires $\Omega(n)$ bits of communication [70, 86]. (In other words, Alice and Bob cannot do asymptotically better than just exchanging A and B with a trivial communication protocol.)

The second reason that communication complexity is so versatile is that there are often natural reductions from communication complexity to problems in other areas of computer science. The main contribution of this chapter is to establish such a reduction for property testing. We will then apply this reduction to establish lower bounds on the query complexity for three different property testing problems. The notes section at the end of the chapter discusses more lower bounds we can obtain with this method.

For the first result in this chapter, we revisit the problem of testing k-linearity. We saw in Chapter 7 that $\min\{k, n-k\} \cdot (1 - o(1))$ queries are required to test k-linearity. In this chapter, we establish a slightly less precise lower bound of $\Omega(\min\{k, n-k\})$ queries. The proof, however, ends up being significantly simpler.

Theorem 11.1. *For any $0 < \epsilon < \frac{1}{2}$, ϵ-testing k-linearity requires $\Omega(\min\{k, n-k\})$ queries.*

The second problem we examine is testing monotonicity. For this problem, we consider a slightly more general class of functions. Fix $R \subseteq \mathbb{R}$. Recall that the function $f : \{0,1\}^n \to R$ is *monotone* if for any $x \preceq y$, $f(x) \leq f(y)$. We show that when R is large enough, testing monotonicity requires a large number of queries.

Theorem 11.2. *Fix $R \subseteq \mathbb{R}$. For any $0 < \epsilon < \frac{1}{8}$, ϵ-testing the function $f : \{0,1\}^n \to R$ for monotonicity requires $\Omega(\min\{n, |R|^2\})$ queries.*

Finally, the third result we establish in this chapter is a lower bound for testing whether a function can be computed by a decision tree of size s. We saw in Theorem 7.10 that $\Omega(\log s)$ queries are required to test this property. The next result shows that if we require the tester to have one-sided error (i.e., to always accept functions computable by size-s decision trees), then $\Omega(s)$ queries are required for the task.

Theorem 11.3. *For any $0 < \epsilon < \frac{3}{8}$, at least $\Omega(s)$ queries are required to ϵ-test size-s decision trees with one-sided error.*

11.1 Communication Complexity Definitions

This section contains a brief introduction to the communication complexity definitions and results that will be used in the proofs in the remainder of the chapter. For a more detailed introduction to communication complexity, we highly recommend the book of Kushilevitz and Nisan [76] to the reader.

11.1.1 The Model

In a typical communication game, there are two parties—Alice, who receives an input x, and Bob, who receives some input y. Alice and Bob wish to jointly compute some function $f(x, y)$ of their inputs. Neither player sees all the information needed to compute f, so they must communicate together to solve the problem. Communication complexity is the study of how much communication is necessary to compute f, for various functions f.

A *protocol* is a distributed algorithm that Alice and Bob use to compute $f(x, y)$; in particular, it specifies what messages Alice and Bob send to each other. In a *deterministic* protocol, Alice's messages are a function only of her input x and the previous communication in the protocol. Similarly, Bob's messages are a function of y and the previous communication. The *cost* of a protocol is the maximum (over all inputs) number of bits sent by Alice and Bob. The deterministic communication complexity of f, denoted $D(f)$, is the minimum cost of a deterministic protocol computing f.

In a *randomized* protocol, Alice and Bob have shared access to a (public coin) random string $r \in \{0,1\}^*$. We say that P is an δ-error protocol for f if for any input pair x, y, P computes $f(x, y)$ with probability at least $1 - \delta$, where the probability is taken over the random string r. We use $R_\delta(f)$ to denote the minimum cost of an ϵ-error protocol for f and define $R(f) := R_{1/3}(f)$. When f is a binary function, we say that a protocol computes f with *one-sided error* if there exists $z \in \{0,1\}$ such that P computes f with certainty whenever $f(x, y) \neq z$, and with probability at least $1 - \delta$ when $f(x, y) = z$. When considering randomized protocols with one-sided error, it is important to note which "side" the error guarantee is on. We use $R^z_\delta(f)$ to denote the minimum cost of a randomized protocol for f that correctly computes f whenever $f(x, y) \neq z$ and computes f with probability at least $1 - \delta$ whenever $f(x, y) = z$. We define $R^z(f) := R^z_{1/3}(f)$.

A protocol is *one-way* if the communication consists of a single message from Alice to Bob, who then outputs an answer. We use $R^{\rightarrow}_\delta(f)$ to denote the minimum communication cost of a randomized, δ-error, one-way protocol for f. Finally, we use $R^{\rightarrow,z}_\delta(f)$ to denote the minimum communication cost of randomized one-way protocols for f with with one-sided error δ, and we define $R^{\rightarrow,z}(f) := R^{\rightarrow,z}_{1/3}(f)$.

11.1.2 Set Disjointness

In the Set Disjointness communication problem, Alice and Bob are given n-bit strings x and y respectively and wish to compute

$$\textsc{disj}_n(x, y) := \bigvee_{i=1}^{n} x_i \wedge y_i \,.$$

Equivalently, Alice and Bob's inputs can be described as sets $A, B \subseteq [n]$. In this case, $\textsc{disj}(A, B) = 1$ if and only if their sets intersect.

When n is clear from context, we drop the subscript. A celebrated result of Kalyanasundaram and Schnitger, later simplified by Razborov, showed that $R(\textsc{disj}_n) = \Omega(n)$, even under the promise that A and B intersect in at most one element.

Theorem 11.4 ([70, 86])**.** $R(\textsc{disj}_n) = \Omega(n)$.

We will also use a balanced version of disjointness called k-$\textsc{bal}$-$\textsc{disj}$. In this version, Alice receives a set $A \subseteq [n]$ of size $|A| = \lfloor k/2 \rfloor + 1$, Bob receives a set $B \subseteq [n]$ of size $\lceil k/2 \rceil + 1$, and there is a promise that $|A \cap B| \leq 1$.

Lemma 11.5. *For all* $0 \leq k \leq n - 2$*, we have* $R(k\text{-}\textsc{bal}\text{-}\textsc{disj}) = \Omega\left(\min\{k, n - k\}\right)$.

Proof. If $n - k = O(1)$, there is nothing to prove. Otherwise, let $m := \min\{\lfloor k/2 \rfloor + 1, n - k - 2\}$. We reduce from $\textsc{disj}_m$. Partition the elements of $[n] \setminus [m]$ into sets $I := \{m+1, \ldots, m+1+\lfloor k/2 \rfloor\}$ and $J := \{m+2+\lfloor k/2 \rfloor, \ldots, n\}$. Note that $|I| = \lfloor k/2 \rfloor + 1$. Furthermore, we have $|J| \geq \lceil k/2 \rceil + 1$, since

$$\begin{aligned}
|J| &= n - (m + 2 + \lfloor k/2 \rfloor) + 1 \\
&= n - 1 - m - \lfloor k/2 \rfloor \\
&= n - 1 - m + \lceil k/2 \rceil - k \\
&= \lceil k/2 \rceil + 1 + n - 2 - m - k \\
&\geq \lceil k/2 \rceil + 1 \,,
\end{aligned}$$

where the penultimate equality holds because $k = \lfloor k/2 \rfloor + \lceil k/2 \rceil$, and the inequality comes from the fact that $m \leq n - k - 2$.

Let A' and B' be the sets received by Alice and Bob respectively as inputs to $\textsc{disj}_m$. Alice pads her input with elements from I until she gets a set of size $\lfloor k/2 \rfloor + 1$. Bob similarly pads his input with elements from J. Let $a := \lfloor k/2 \rfloor + 1 - |A'|$ and $b :=$

$\lceil k/2 \rceil + 1 - |B'|$. Specifically, Alice sets $A = A' \cup \{m+1, \ldots, m+a\}$ and Bob sets $B = B' \cup \{n, n-1, \ldots, n-b+1\}$.

Note that $|A| = \lfloor k/2 \rfloor + 1$, $|B| = \lceil k/2 \rceil + 1$, and $A \cap B = A' \cap B'$. Therefore, a solution to $k\text{-}\textsc{bal-disj}(A, B)$ gives a solution to $\textsc{disj}_m(A', B')$, hence

$$R(k\text{-}\textsc{bal-disj}) \geq R(\textsc{disj}_m) = \Omega(m) = \Omega(\min\{k, n-k\}) \ . \qquad \square$$

11.1.3 Gap Equality

In the Gap Equality problem, Alice and Bob are given n-bit strings x and y respectively and wish to compute

$$\textsc{geq}_{n,t}(x, y) := \begin{cases} 1 & \text{if } x = y \,, \\ 0 & \text{if } \operatorname{dist}(x, y) = t \,, \\ * & \text{otherwise.} \end{cases}$$

We drop the subscripts when n is clear from context and $t = n/8$. When $\textsc{geq}(x, y) = *$, we allow the protocol to output 0 or 1. We are interested in $R^z(\textsc{geq})$; recall that $R^z(\textsc{geq})$ is the minimum communication cost of a protocol for $\textsc{geq}$ that only makes mistakes when $\textsc{geq}(x, y) = z$. The standard public-coin EQUALITY protocol gives $R^0(\textsc{geq}) = O(1)$. For protocols that only err when $\textsc{geq}(x, y) = 1$, the complexity is drastically different.

Buhrman, Cleve, and Wigderson [35] proved an $\Omega(n)$ lower bound on the deterministic communication complexity of $\textsc{geq}_{n,n/2}$; their result extends to other gap sizes and to randomized protocols with one-sided error.

Lemma 11.6 ([35]). $R^1(\textsc{geq}_{n,t}) = \Omega(n)$ *for all even* $t = \Theta(n)$.[1]

11.2 Main Reduction Lemma

Below we define a class of property testing communication games and show how communication lower bounds for these games yield query complexity lower bounds for property testers. Our communication games are based on what we call *combining operators*.

[1] Curiously, the parity of t turns out to be necessary. Since $\operatorname{dist}(x, y) = |x| + |y| - 2|x \wedge y|$, Alice and Bob can deterministically distinguish $x = y$ from $\operatorname{dist}(x, y)$ being odd with a single bit of communication—Alice sends Bob the parity of $|x|$, and Bob computes the parity of $|x| + |y|$. This does not affect our property testing lower bounds.

Definition 11.7 (Combining operator)**.** A *combining operator* is an operator ψ that takes as input two functions $f, g : \{0,1\}^n \to Z$ and returns a function $h : \{0,1\}^n \to R$.

We refer to the inputs f and g as the *base functions* of ψ. By convention, we use h to refer to the output of ψ. Given a combining operator ψ and a property $\mathcal{P}$, we define $\mathrm{C}_\psi^{\mathcal{P}}$ to be the following property testing communication game. Alice receives f. Bob receives a function g. They need to compute

$$\mathrm{C}_\psi^{\mathcal{P}}(f,g) := \begin{cases} 1 & \text{if } \psi(f,g) \in \mathcal{P} \\ 0 & \text{if } \psi(f,g) \text{ is } \epsilon\text{-far from } \mathcal{P}. \end{cases}$$

We prove all of our testing lower bounds by reducing from an associated communication game $\mathrm{C}_\psi^{\mathcal{P}}$. This reduction is simple—Alice and Bob solve $\mathrm{C}_\psi^{\mathcal{P}}$ by emulating a $\mathcal{P}$-testing algorithm on $h := \psi(f,g)$. Note that neither Alice nor Bob have enough information to evaluate a query $h(x)$, because h depends on both f and g. Instead, they must communicate to jointly compute $h(x)$. For this reduction to give a strong query complexity lower bound for the property testing problem, it is essential that the joint computation of $h(x)$ occurs in a communication-efficient manner.

The following definition gives a sufficient condition on combining operators that yield strong reductions to testing problems.

Definition 11.8 (Simple combining operator)**.** A combining operator ψ is *simple* if for all f, g, and for all x, the query $h(x)$ can be computed given only x and the queries $f(x)$ and $g(x)$.

For example, when the base functions are boolean, the combining operator defined by $\psi(f,g) := f \oplus g$ is clearly simple—each $h(x) = f(x) \oplus g(x)$ can trivially be computed from $f(x)$ and $g(x)$. On the other hand, the combining operator ψ that returns the function defined by $h(x) := \bigoplus_{y \in T} [f(y) \cdot g(y)]$ is not simple when T is a large set of strings (say a Hamming ball centered at x), since computing $h(x)$ requires knowledge of $f(y)$ and $g(y)$ for several y.

All of the property testing communication games we use in this paper are based on simple combining operators and give us a tight connection between property testing and communication complexity via the following lemma.

Lemma 11.9 (Main Reduction Lemma)**.** *Fix Z to be a finite set. For any simple combining operator ψ with base functions $f, g : \{0,1\}^n \to Z$ and any property $\mathcal{P}$, we have*

(i) $R(\mathrm{C}_\psi^{\mathcal{P}}) \le 2\, Q(\mathcal{P}) \cdot \lceil \log |Z| \rceil$,

(ii) $R^0(\mathrm{C}_\psi^{\mathcal{P}}) \le 2\,Q^1(\mathcal{P}) \cdot \lceil \log |Z| \rceil$,
(iii) $R^{\to}(\mathrm{C}_\psi^{\mathcal{P}}) \le Q^{\mathrm{na}}(\mathcal{P}) \cdot \lceil \log |Z| \rceil$, *and*
(iv) $R^{\to,0}(\mathrm{C}_\psi^{\mathcal{P}}) \le Q^{\mathrm{na},1}(\mathcal{P}) \cdot \lceil \log |Z| \rceil$.

Proof. We begin by proving (iii). Let $\mathcal{A}$ be a q-query non-adaptive tester for $\mathcal{P}$. We create a one-way protocol P for $\mathrm{C}_\psi^{\mathcal{P}}$ in the following manner. Alice and Bob use public randomness to generate queries $x^{(1)}, \dots, x^{(q)}$. Then, Alice computes $f(x^{(1)}), \dots, f(x^{(q)})$ and sends them to Bob in a single $(q \cdot \lceil \log |Z| \rceil)$-bit message. For each i, Bob computes $g(x^{(i)})$ and combines it with $f(x^{(i)})$ to compute $h(x^{(i)})$. Finally, Bob emulates $\mathcal{A}$ using the responses $h(x^{(1)}), \dots, h(x^{(q)})$ and outputs 1 if and only if $\mathcal{A}$ accepts h.

If $\mathcal{A}$ has two-sided error, then by the correctness of $\mathcal{A}$, P computes $\mathrm{C}_\psi^{\mathcal{P}}$ with probability at least $2/3$. Hence, $R^{\to}(\mathrm{C}_\psi^{\mathcal{P}}) \le q \cdot \lceil \log |Z| \rceil$. In particular, if $\mathcal{A}$ is an optimal non-adaptive tester with two-sided error, then $q = Q^{\mathrm{na}}(\mathcal{P})$, and part (iii) of the lemma is proved.

If $\mathcal{A}$ has one-sided error, then whenever $h \in \mathcal{P}$, the protocol P correctly outputs 1, and when h is ϵ-far from $\mathcal{P}$, the protocol correctly outputs 0 with probability at least $2/3$. Therefore, $R^{\to,0}(\mathrm{C}_\psi^{\mathcal{P}}) \le q \cdot \lceil \log |Z| \rceil$. In particular, when $\mathcal{A}$ is an optimal non-adaptive tester with one-sided error, $R^{\to,0}(\mathrm{C}_\psi^{\mathcal{P}}) \le Q^1(\mathcal{P}) \cdot \lceil \log |Z| \rceil$.

Now, suppose $\mathcal{A}$ is a q-query adaptive tester for $\mathcal{P}$. Again, Alice and Bob use public randomness to generate queries $x^{(1)}, \dots, x^{(q)}$. However, since $\mathcal{A}$ is adaptive, the distribution of the ith query $x^{(i)}$ depends on $h(x^{(j)})$ for all $j < i$. Instead of generating all queries in advance, Alice and Bob generate queries one at a time. Each time a query $x^{(i)}$ is generated, Alice and Bob exchange $f(x^{(i)})$ and $g(x^{(i)})$. Since ψ is a simple combining operator, this is enough information for Alice and Bob to individually compute $h(x^{(i)})$, which in turn gives them enough information to generate the next query with the appropriate distribution. When $h(x^{(1)}), \dots, h(x^{(q)})$ have all been computed, Bob outputs 1 if and only if $\mathcal{A}$ accepts h. This protocol costs $2q \cdot \lceil \log |Z| \rceil$ bits of communication, and if $\mathcal{A}$ is an optimal adaptive tester, then $R(\mathrm{C}_\psi^{\mathcal{P}}) \le 2\,Q(\mathcal{P}) \cdot \lceil \log |Z| \rceil$. Similarly, if $\mathcal{A}$ is an optimal adaptive tester with one-sided error, then $R^0(\mathrm{C}_\psi^{\mathcal{P}}) \le 2\,Q^1(\mathcal{P}) \cdot \lceil \log |Z| \rceil$. □

11.3 Testing k-Linearity

In this section we prove Theorem 11.1.

Theorem 11.1 (Restated). *For any* $0 < \epsilon < \frac{1}{2}$, *$\epsilon$-testing k-linearity requires* $\Omega(\min\{k, n-k\})$ *queries.*

Proof. We prove the lower bound with a reduction from the k-BAL-DISJ problem. Recall that $\mathrm{C}_{\oplus}^{k\text{-lin}}$ is the communication game where the inputs are the functions $f, g : \{0,1\}^n \to \{0,1\}$ and the players must test whether the function $h = f \oplus g$ is k-linear. Lemmas 11.9 and 11.5 imply that

$$2\,Q(k\text{-linearity}) \geq R(\mathrm{C}_{\oplus}^{k\text{-lin}}) \quad \text{and} \quad R(k\text{-BAL-DISJ}) = \Omega(\min\{k, n-k\})\,.$$

To complete the proof, we show that $R(\mathrm{C}_{\oplus}^{k\text{-lin}}) \geq R(k\text{-BAL-DISJ})$ with a reduction from k-BAL-DISJ to $\mathrm{C}_{\oplus}^{k\text{-lin}}$.

Let $A, B \subseteq [n]$ be the two sets of size $|A| = \lfloor \frac{k}{2} \rfloor + 1$ and $|B| = \lceil \frac{k}{2} \rceil + 1$ received by Alice and by Bob, respectively, as the input to an instance of k-BAL-DISJ. Alice and Bob can construct the functions $\mathrm{Parity}_A, \mathrm{Parity}_B : \{0,1\}^n \to \{0,1\}$. When $|A \cap B| = 1$, the symmetric difference of the two sets has size $|A \triangle B| = |A| + |B| - 2|A \cap B| = k$, and the function $\mathrm{Parity}_A \oplus \mathrm{Parity}_B = \mathrm{Parity}_{A \triangle B}$ is k-linear. Conversely, when A and B are disjoint, the function $\mathrm{Parity}_A \oplus \mathrm{Parity}_B$ is a (k+2)-parity function and, by Proposition 2.24, it is $\frac{1}{2}$-far from k-linear functions. So Alice and Bob can solve their instance of k-BAL-DISJ with a communication protocol for $\mathrm{C}_{\oplus}^{k\text{-lin}}$. This implies that $R(\mathrm{C}_{\oplus}^{k\text{-lin}}) \geq R(k\text{-BAL-DISJ})$, as we wanted to show. □

11.4 Testing Monotonicity

Theorem 11.2 (Restated). *Fix $R \subseteq \mathbb{R}$. For any $0 < \epsilon < \frac{1}{8}$, ϵ-testing the function $f : \{0,1\}^n \to R$ for monotonicity requires $\Omega(\min\{n, |R|^2\})$ queries.*

Proof. We prove the theorem in three steps. First, we give an $\Omega(n)$ lower bound for the case when $R = \mathbb{Z}$. Secondly, we handle the case where $|R| = \sqrt{n}$ by a standard range reduction argument. Finally, we give an $\Omega(|R|^2)$ bound for small $|R|$ by reducing from the $|R| = \sqrt{n}$ case.

Suppose $R = \mathbb{Z}$. We prove the lower bound for testing monotonicity in this case with a reduction from SET-DISJOINTNESS. Let ψ be the combining function that, given two functions $f, g : \{0,1\}^n \to \{0,1\}$ and an element $x \in \{0,1\}^n$, returns $\psi(f, g, x) = 2\,|x| + f(x) + g(x)$. Define $\mathrm{C}_{\psi}^{\mathrm{MONO}}$ be the communication game where Alice and Bob are given two functions $f, g : \{0,1\}^n \to \{0,1\}$ and they must test whether the function $h : \{0,1\}^n \to \mathbb{R}$ defined by $h(x) = \psi(f, g, x)$ is monotone. By Lemma 11.9 and Theorem 11.4,

$$2\,Q(\mathrm{MONO}) \geq R(\mathrm{C}_{\psi}^{\mathrm{MONO}}) \quad \text{and} \quad R(\mathrm{DISJ}) \geq \Omega(n)\,.$$

We complete the proof by showing that $R(\mathrm{C}_{\psi}^{\mathrm{MONO}}) \geq R(\mathrm{DISJ})$.

Let $A, B \subseteq [n]$ be the subsets received by Alice and Bob as the input to an instance of the SET-DISJOINTNESS problem. Alice and Bob can build the functions $\chi_A, \chi_B : \{0,1\}^n \to \{-1,1\}$, respectively, by setting $\chi_A(x) = (-1)^{\sum_{i \in A} x_i}$ and $\chi_B(x) = (-1)^{\sum_{i \in B} x_i}$. Let $h : \{0,1\}^n \to \mathbb{R}$ be the function defined by $h(x) = \psi(\chi_A, \chi_B, x) = 2 \cdot |x| + \chi_A(x) + \chi_B(x)$. We claim that (a) when A and B are disjoint, h is monotone, and (b) when A and B are not disjoint, h is $\frac{1}{8}$-far from monotone. If this claim is true, then we have completed our lower bound since it implies that Alice and Bob can run a protocol for $\mathrm{C}_{\psi}^{\textsc{mono}}$ to solve their instance of SET-DISJOINTNESS and, therefore, $R(\mathrm{C}_{\psi}^{\textsc{mono}}) \geq R(\textsc{disj})$.

We now prove Claim (a). Fix $i \in [n]$. For $x \in \{0,1\}^n$, let $x_0, x_1 \in \{0,1\}^n$ be the vectors obtained by fixing the ith coordinate of x to 0 and to 1, respectively. For any set $S \subseteq [n]$, $\chi_S(x_1) = (-1)^{\mathbf{1}[i \in S]} \cdot \chi_S(x_0)$. So when $i \notin A$ and $i \notin B$,

$$h(x_1) - h(x_0) = 2\,|x_1| - 2\,|x_0| = 2 > 0\ ,$$

when $i \in A$ and $i \notin B$,

$$h(x_1) - h(x_0) = 2\,|x_1| - 2\,|x_0| - 2\,\chi_A(x_0) \geq 0\ ,$$

and when $i \notin A$ and $i \in B$,

$$h(x_1) - h(x_0) = 2\,|x_1| - 2\,|x_0| - 2\,\chi_B(x_0) \geq 0\ .$$

Those three inequalities imply that when $i \notin A \cap B$, the function h is monotone on each edge (x_0, x_1) in the ith direction. As a result, when A and B are disjoint the function h is monotone.

Let us now prove Claim (b). Let $A \cap B \neq \emptyset$. When $i \in A \cap B$,

$$h(x_1) - h(x_0) = 2\,|x_1| - 2\,|x_0| - 2\,\chi_A(x_0) - 2\,\chi_B(x_1).$$

This implies that for each x where $\chi_A(x_0) = \chi_B(x_0) = 1$, $h(x_1) < h(x_0)$. Partition $\{0,1\}^n$ into 2^{n-1} pairs that form the endpoints to all the edges in the ith direction. Exactly $\frac{1}{4}$ of these pairs will satisfy the condition $\chi_A(x_0) = \chi_B(x_0) = 1$, and for each of these pairs, either $h(x_0)$ or $h(x_1)$ must be modified to make h monotone. So when A and B are not disjoint, h is $\frac{1}{8}$-far from monotone.

To handle the case where $|R| = \sqrt{n}$, we sketch the proof of a standard range reduction argument (see, e.g., [33].) Specifically, we can assume without loss of generality that $R = \{-\frac{\sqrt{n}}{2}, \ldots, \frac{\sqrt{n}}{2}\}$ and we modify the construction of the function h to create h'

$$h'(x) = \begin{cases} -\frac{\sqrt{n}}{2} & \text{when } |x| - \frac{n}{2} < -\frac{\sqrt{n}}{2} + 1, \\ \frac{\sqrt{n}}{2} & \text{when } |x| - \frac{n}{2} > \ \frac{\sqrt{n}}{2} - 1, \\ |x| - \frac{n}{2} + \frac{\chi_A(x) + \chi_B(x)}{2} & \text{when } \big|\,|x| - \frac{n}{2}\big| \leq \frac{\sqrt{n}}{2} - 1. \end{cases}$$

It is easy to see that h' is identical to $h/2$, except when $\big| |x| - \frac{n}{2} \big| \geq \frac{\sqrt{n}}{2}$, which only occurs for a constant fraction of x's. Using the same reasoning as before, h' is monotone when A and B are disjoint, and a constant distance from monotone when A and B intersect. We leave the details to the reader.

Finally, suppose that $|R| = o(\sqrt{n})$, and let $m := |R|^2$. We will use a q-query testing algorithm for f to create a q-query testing algorithm for functions $g : \{0,1\}^m \to \{0,1\}$.

Specifically, given g, create $h : \{0,1\}^n \to R$ by defining $h(x,y) := g(x)$ for $x \in \{0,1\}^m$ and $y \in \{0,1\}^{n-m}$. Clearly, if g is monotone then so is h. We now want to argue that if g is ϵ-far from monotone, then so is h. We do so by proving the contrapositive. Suppose that h is *not* ϵ-far from monotone. Let $\tilde{h}$ be the monotone function closest to h; thus, $\Pr_{x,y}[h(x,y) \neq \tilde{h}(x,y)] \leq \epsilon$. By an averaging argument, there exists y such that $\Pr_x[h(x,y) \neq \tilde{h}(x,y)] \leq \epsilon$. Define $\tilde{g} : \{0,1\}^m \to R$ as $\tilde{g}(x) := \tilde{h}(x,y)$. It is easy to see that $\Pr_x[g(x) \neq \tilde{g}(x)] = \Pr_{x,y}[h(x,y) \neq \tilde{h}(x,y)] \leq \epsilon$. Therefore, g is not ϵ-far from monotone.

Our testing algorithm for g is simple: test h and return the result. By the above claim, a correct answer for testing h gives a correct answer for testing g. Since testing g for monotonicity requires $\Omega(m) = \Omega(|R|^2)$ queries, the same bound holds for testing h. □

11.5 Decision trees

Theorem 11.3 (Restated)**.** *For any $0 < \epsilon < \frac{3}{8}$, at least $\Omega(s)$ queries are required to ϵ-test size-s decision trees with one-sided error.*

Proof. We first consider the case where $s = 2^{n-1}$ for some $n \geq 5$. We prove this case with a reduction from the GAP-EQUALITY problem on s-bit strings. Let $\mathrm{C}_{\oplus}^{s\text{-DT}}$ be the communication game where Alice and Bob receive the functions $f, g : \{0,1\}^n \to \{0,1\}$ and they must test whether the function $h = f \oplus g$ is computable by a decision tree of size s. By Lemmas 11.9 and 11.6,

$$2\,Q^1\big(s\text{-DT}\big) \geq R^1(\mathrm{C}_{\oplus}^{s\text{-DT}}) \quad \text{and} \quad R^1(\mathrm{GEQ}_{s,\frac{s}{8}}) = \Omega\big(s\big)\ .$$

We complete the proof by showing that $R^1(\mathrm{C}_{\oplus}^{s\text{-DT}}) \geq R^1(\mathrm{GEQ}_{s,\frac{s}{8}})$.

Let $a, b \in \{0,1\}^s$ be received by Alice and Bob as input to an instance of the GAP-EQUALITY problem. They must determine if $a = b$ or whether $\mathrm{dist}(a,b) = \frac{s}{8}$. Alice and Bob can solve their instance of the GEQ problem with the following protocol. Let the set of vectors $x \in \{0,1\}^n$ with even parity $\mathrm{Parity}(x) = x_1 \oplus \cdots \oplus x_n = 0$ define an indexing of the bits of a. (I.e., fix a bijection between those strings and $[s]$.) Alice and Bob build the functions $f, g : \{0,1\}^n \to \{0,1\}$ by setting

$$f(x) = \begin{cases} a_x & \text{when } \mathrm{Parity}(x) = 0, \\ 0 & \text{when } \mathrm{Parity}(x) = 1, \end{cases}$$

and

$$g(x) = \begin{cases} b_x & \text{when } \mathrm{Parity}(x) = 0, \\ 1 & \text{when } \mathrm{Parity}(x) = 1. \end{cases}$$

Alice and Bob then test whether $f \oplus g$ can be represented with a decision tree of size at most $\frac{15}{16}2^n$; when it can, they answer $\mathrm{dist}(a,b) = \frac{s}{8}$.

Let us verify the correctness of this protocol. For any $x \in \{0,1\}^n$ where $\mathrm{Parity}(x) = 0$, we have that $(f \oplus g)(x) = a_x \oplus b_x$. Furthermore, for each x where $\mathrm{Parity}(x) = 1$, we get $(f \oplus g)(x) = 1$. So when $a = b$, then $f \oplus g$ is the Parity function. By Lemma **??**, this function is $\frac{1}{32}$-far from every decision tree of size at most $\frac{15}{16}2^n$. When $\mathrm{dist}(a,b) = \frac{s}{8}$, consider the (complete) tree that computes $f \oplus g$ by querying x_i in every node at level i. This tree has 2^n leaves, but for every input x where $a_x \neq b_x$, we have that the corresponding leaf has the same value as its sibling. So for each such input, we can eliminate one leaf. Therefore, we can compute $f \oplus g$ with a decision tree of size at most $2^n - 2^{n-1}/8 < \frac{15}{16}2^n$. □

11.6 Notes and Discussion

Other results. As we saw in Chapter 7, lower bounds on the query complexity for testing k-linearity—or, more precisely, on the number of queries required to distinguish k-linear and $(k+2)$-linear functions—yield lower bounds on the query complexity for testing a number of other properties of boolean functions.

Our proof of Theorem 11.1 does give a lower bound on the number of queries required to distinguish k-linear and $(k+2)$-linear functions. As a result, all the (asymptotic) bounds in Table 7.1 can be obtained directly from Theorem 11.1.

The lower bound on the query complexity required to test monotonicity in Theorem 11.2 also implies a matching $\Omega(n)$ lower bound on the query complexity for testing submodularity. This follows from a result of Seshadhri and Vondrák [91], who showed that testing submodularity is at least as hard (in terms of query complexity) as testing monotonicity. For the details, see [23].

Following the initial publication of the research presented in this chapter, the communication complexity method has been used by Brody, Matulef, and Wu [34] to establish new lower bounds related to testing properties computable by small-width ordered binary decision diagrams. It has also been used by Jha and Raskhodnikova [68] to obtain lower bounds for testing Lipschitz functions, as part of their investigation on property testing and data privacy.

Chapter 12

Active Testing

In this chapter, we introduce a new model of property testing, which we call the *active testing* model. Before describing the model itself, we begin by describing the *model selection* problem in learning theory which motivated the research presented in this chapter.

Motivation. The central problem in learning theory can be formulated as follows. An algorithm $\mathcal{A}$ is given query access to some function $f : \{0,1\}^n \to \{0,1\}$ that is promised to have some given property $\mathcal{P}$.[1] The learner $\mathcal{A}$'s task is to identify a hypothesis function $h : \{0,1\}^n \to \{0,1\}$ that, with probability at least $1 - \delta$, is ϵ-close to f. The goal is to minimize the number of queries to f and the running time required by $\mathcal{A}$ to complete this task. When $\mathcal{A}$ is free to query f on any inputs of its choosing, the resulting learning framework is called the *membership query* model. If the hypothesis h is guaranteed to also have property $\mathcal{P}$, the algorithm $\mathcal{A}$ is called a *proper* learner.

Consider now a slight variation on the basic learning problem: a learning algorithm $\mathcal{A}$ is again given query access to some function $f : \{0,1\}^n \to \{0,1\}$, but the promise is weakened to only guarantee that f has at least one of the properties $\mathcal{P}_1, \ldots, \mathcal{P}_m$. As before, $\mathcal{A}$ is required to find a hypothesis $h : \{0,1\}^n \to \{0,1\}$ that is ϵ-close to f with probability at least $1 - \delta$. How many queries does $\mathcal{A}$ require for this task?

One way $\mathcal{A}$ can learn a good hypothesis h for f is to learn m hypotheses $h_1, \ldots, h_m$, one for each of the possible assumptions $f \in \mathcal{P}_1, \ldots, f \in \mathcal{P}_m$. It can then test these hypotheses to identify the good hypothesis. This approach is sound, but it is not query-efficient. If $q_1, \ldots, q_m$ are the minimum number of queries required to learn a good hypothesis for a function f in $\mathcal{P}_1, \ldots, \mathcal{P}_m$, respectively, then this approach requires at least

[1] In learning theory, a property of boolean functions is typically called a *class* of functions.

$q_1 + \cdots + q_m$ queries.

A more query-efficient solution to the problem is to perform *model selection*: instead of learning one hypothesis per property, we first identify a property $\mathcal{P}_i$ of the function f, then we learn a good hypothesis for f under the promise that $f \in \mathcal{P}_i$.

For the model selection approach to be query-efficient, we need a good way to tell if f has properties $\mathcal{P}_1, \ldots, \mathcal{P}_m$. This is where property testing comes in handy: by testing whether f is in $\mathcal{P}_1, \ldots, \mathcal{P}_m$, we can quickly reject all the properties that do not contain any hypothesis ϵ-close to f while accepting the property (or properties) that contains f. Furthermore, very efficient testers have been designed for many properties commonly studied in machine learning: linear threshold functions [77], unions of intervals [74], juntas [52, 20], DNFs [46], and decision trees [46].

Active learning. While the application of property testing to model selection as described above is interesting, its potential utility is limited by the fact that the assumptions in the membership query model are unrealistic in most machine learning applications.

For example, consider the problem of classifying documents by topic. A document can be represented as a boolean vector $x \in \{0,1\}^n$ by letting each x_i denote the presence (or absence) of a given word in the document. A boolean classifier—for illustration, let's consider "sports articles" vs. "non-sports articles"—can be described as a boolean function $f : \{0,1\}^n \to \{0,1\}$. Let $\mathcal{A}$ be a learning algorithm that is tasked with constructing a classifier $h : \{0,1\}^n \to \{0,1\}$ that is close to f. To execute $\mathcal{A}$, we must give it query access to f. When we have an existing article (say, taken from the web) whose corresponding word vector is $x \in \{0,1\}^n$, we can let $\mathcal{A}$ query the value of $f(x)$ by having a user determine if the article is about sports or not. This approach, however, does not work if we generate an arbitrary input $x \in \{0,1\}^n$ and then try to get users to determine the value of $f(x)$.

As a result, the dominant query paradigm in machine learning is not the membership query model, but the *active learning* model. In this model, the algorithm can still choose the inputs on which it wants to query the function f, but it can only choose inputs *among those that exist in nature*. We describe this model more precisely in Section 12.1.

Results. In this chapter, we bring the active model in learning to the domain of testing. The main contribution of the research presented in this chapter is the definition of the model itself, which we present in Section 12.1.

The active testing model is a restricted form of property testing, so it is natural to expect that some properties which are efficiently testable in the standard property testing model are not as efficiently testable in the active testing model. Our first (negative) result confirms this suspicion. Recall that dictator functions can be tested with a constant number of queries in the standard property testing model. In the active testing model, however, testing dictator functions requires as many queries as are needed to *learn* dictator functions.

Theorem 12.1. *Active testing of dictatorships under the uniform distribution requires* $\Omega(\log n)$ *queries.*

The proof of the theorem says something even stronger: $\Omega(\log n)$ queries are required in the active testing model to distinguish dictator functions from *random* functions. As a result, the same lower bound applies to the query complexity of many other property testing tasks: testing computability by small decision trees, testing juntas, testing linear threshold functions, etc. Despite this note of caution, it is still possible that many of those properties can still be tested much more efficiently than they can be learned.

The second, and principal, result of this section confirms that there are indeed fundamental properties of boolean functions that can be tested more efficiently than they can be learned in the active testing model. Specifically, we show that testing linear threshold functions is much easier than learning the same class of functions.

Theorem 12.2. *There is an active tester for linear threshold functions over the standard* n*-dimensional Gaussian distribution that makes* $O(\sqrt{n \log n})$ *queries. Furthermore, any such active tester for linear threshold functions makes at least* $\Omega(n^{1/3})$ *queries.*

There are other properties of functions that can be tested efficiently in the active model. We discuss these results and the relation of active testing to other models of property testing in Section 12.5.

12.1 The active testing model

Active learning, in general, describes models of machine learning in which the learning algorithm has some freedom in choosing what inputs on which to query the target function. The name *active* is chosen to contrast with the *passive learning* model, in which the learning algorithm observes the value of the target function on some inputs that are drawn at random from some distribution. The “most active” model of learning, in which the learning algorithm has complete freedom over the set of queries it makes to the target function, is called the *membership query* learning model.

The model of active learning that we consider in this chapter lies somewhere in between the passive and membership query learning models. Specifically, we consider the following class of learning algorithms.

Definition 12.3 (Sample-restricted learner)**.** The randomized algorithm $\mathcal{A}$ with oracle access to a function $f : \{0,1\}^n \to \{0,1\}$ in $\mathcal{P}$ is an *s-sample, q-query (ϵ, δ)-learner* for the property $\mathcal{P}$ over the distribution $\mathcal{D}$ if it draws a set S of $|S| = s$ samples from $\mathcal{D}$, queries the value of f on q elements from S, and with probability at least $1 - \delta$ outputs a hypothesis $h : \{0,1\}^n \to \{0,1\}$ that satisfies $\operatorname{dist}_{\mathcal{D}}(f, h) \leq \epsilon$.

Our concept of *active learner* corresponds to a sample-restricted learner that has access to a polynomial number of (unlabeled) samples.

Definition 12.4 (Active learner)**.** The algorithm $\mathcal{A}$ is a *q-query active (ϵ, δ)-learner* for $\mathcal{P}$ over the distribution $\mathcal{D}$ if it is a $\operatorname{poly}(n)$-sample, q-query (ϵ, δ)-learner for $\mathcal{P}$ over $\mathcal{D}$.

We are now ready to introduce the main definitions for the active testing model. The goal of our definitions is to mirror the active learning definitions. We do so as follows.

Definition 12.5 (Sample-restricted tester)**.** The randomized algorithm $\mathcal{A}$ with oracle access to a function $f : \{0,1\}^n \to \{0,1\}$ is an *s-sample, q-query ϵ-tester* for $\mathcal{P}$ over the distribution $\mathcal{D}$ if it draws s samples from $\mathcal{D}$, queries the value of f on q of those samples, and

1. Accepts with probability at least $\frac{2}{3}$ when $f \in \mathcal{P}$; and
2. Rejects with probability at least $\frac{2}{3}$ when $\operatorname{dist}_{\mathcal{D}}(f, \mathcal{P}) \geq \epsilon$.

Note that our definition of sample-restricted tester coincides with the standard definition of a tester when the number of samples is unlimited and the support of $\mathcal{D}$ is $\{0,1\}^n$.

When we fix $q = s$, then the definition of sample-restricted tester corresponds to the *passive testing* model first studied in [59] in which the queries made by the algorithm are completely determined by the random draws to the distribution $\mathcal{D}$.

Our concept of *active tester* corresponds to a sample-restricted tester that has access to a polynomial number of (unlabeled) samples.

Definition 12.6 (Active tester)**.** A randomized algorithm is a *q-query active ϵ-tester* for $\mathcal{P} \subseteq \{0,1\}^n \to \{0,1\}$ over D if it is a $\operatorname{poly}(n)$-sample, q-query ϵ-tester for $\mathcal{P}$ over $\mathcal{D}$.

12.2 Testing dictator functions

We prove the lower bound for testing dictator functions with a variant of Lemma 3.15. We start by stating and proving this basic lemma.

Lemma 12.7. *Fix a property $\mathcal{P} \subseteq \{0,1\}^n \to \{0,1\}$. Let $\mathcal{D}_{\text{yes}}$ be a distribution over $\mathcal{P}$ and let $\mathcal{D}_{\text{no}}$ be a distribution such that $f \sim \mathcal{D}_{\text{no}}$ is ϵ-far from $\mathcal{P}$ with probability $1 - o(1)$. Let $S \subseteq \{0,1\}^n$ be obtained by drawing s samples independently at random from the distribution $\mathcal{D}$ over $\{0,1\}^n$. Suppose that with probability at least $\frac{5}{6}$ (over the choice of S), every subset $X \subseteq S$ of size $|X| = q$ and every $r \in \{0,1\}^q$ satisfy*

$$\Pr_{f \sim \mathcal{D}_{\text{yes}}}[f(X) = r] < \tfrac{6}{5} \Pr_{f \sim \mathcal{D}_{\text{no}}}[f(X) = r]. \tag{12.1}$$

Then there is no s-sample q-query ϵ-tester for $\mathcal{P}$.

Proof. Let $\mathcal{A}$ be a deterministic algorithm that, for each possible subset $S \subseteq \{0,1\}^n$, selects a set $X \subseteq S$ of size $|X| = q$ and queries $f : \{0,1\}^n \to \{0,1\}$ on each input in X. Suppose that $\mathcal{A}$ is guaranteed to accept $f \sim \mathcal{D}_{\text{yes}}$ with probability at least $\frac{2}{3}$ when S contains s elements drawn independently from $\mathcal{D}$.[2]

When every subset $X \subseteq S$ of size $|X| = q$ and every $r \in \{0,1\}^q$ satisfy (12.1), let us call S *bad*. By our assumption on the acceptance probability of $\mathcal{A}$,

$$\mathop{\mathbf{E}}_{S}\left[\Pr_{f \sim \mathcal{D}_{\text{yes}}}[\mathcal{A} \text{ accepts } f] \mid S \text{ is bad}\right] \geq \tfrac{6}{5} \cdot (\tfrac{2}{3} - \tfrac{1}{6}) = \tfrac{3}{5}.$$

But conditioned on S being bad, the probability that $\mathcal{A}$ accepts a function drawn from $\mathcal{D}_{\text{no}}$ cannot be much smaller than that of accepting a function drawn from $\mathcal{D}_{\text{yes}}$ so we have

$$\mathop{\mathbf{E}}_{S}\left[\Pr_{f \sim \mathcal{D}_{\text{no}}}[\mathcal{A} \text{ accepts } f] \mid S \text{ is bad}\right] \geq \tfrac{5}{6} \cdot \tfrac{3}{5} = \tfrac{1}{2}.$$

This means that $\mathcal{A}$ accepts a function drawn from $\mathcal{D}_{\text{no}}$ with probability at least

$$\Pr_{f,S}[\mathcal{A} \text{ accepts } f] \geq \Pr[S \text{ is bad}] \cdot \mathop{\mathbf{E}}_{S}\big[\Pr_{f \sim \mathcal{D}_{\text{no}}}[\mathcal{A} \text{ accepts } f] \mid S \text{ is bad}\big] = \tfrac{5}{6} \cdot \tfrac{1}{2} = \tfrac{5}{12} > \tfrac{1}{3}.$$

Therefore, no deterministic s-sample q-query algorithm can distinguish functions drawn from $\mathcal{D}_{\text{yes}}$ or from $\mathcal{D}_{\text{no}}$ with probability at least $\frac{2}{3}$, and the lemma follows directly from Yao's Minimax Principle [100]. □

[2] Here, the probability that $\mathcal{A}$ accepts is over the random choice of both f and S.

We are now ready to complete the proof of Theorem 12.8.

Theorem 12.8. *Active testing of dictatorships under the uniform distribution requires* $\Omega(\log n)$ *queries.*

Proof. Let $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$ denote the uniform distributions over dictator functions and all boolean functions $\{0,1\}^n \to \{0,1\}$, respectively. Fix X to be a set of q queries from $\{0,1\}^n$. We can write X as a $q \times n$ matrix. For each $r \in \{0,1\}^q$, write $c_r(X)$ to denote the number of columns of X that are identical to r. For every r, we have

$$\left| \Pr_{f\sim\mathcal{D}_{\text{yes}}}[f(X) = r] - \Pr_{f\sim\mathcal{D}_{\text{no}}}[f(X) = r] \right| = \left| \frac{c_r(X)}{n} - 2^{-q} \right|.$$

When X is formed by drawing q queries independently and uniformly at random from $\{0,1\}^n$, we can equivalently say that X is formed by drawing n columns independently and uniformly at random from $\{0,1\}^q$. Thus, for any $r \in \{0,1\}^q$ we have

$$\mathbf{E}[c_r(X)] = 2^{-q}n$$

and by Chernoff's bound

$$\Pr_X \left[\left| \frac{c_r(X)}{n} - 2^{-q} \right| > \frac{1}{36} 2^{-q} \right] < e^{-cn2^{-q}}$$

for some appropriate constant c. By applying the union bound over all 2^q column types in $\{0,1\}^n$ and over the $\binom{\text{poly}(n)}{q} < n^{c'q\log n}$ ways to choose a set X of q queries from the set S of $\text{poly}(n)$ queries sampled independently from the uniform distribution, we get that

$$\Pr_S \left[\exists X \subseteq S, r \in \{0,1\}^q : \left| \frac{c_r(X)}{n} - 2^q \right| > \frac{1}{36} 2^{-q} \right] < e^{c'q\log n + q - cn2^{-q}}.$$

When $q \geq c'' \log n$ for some appropriately large constant c'', this probability is less than $\frac{1}{6}$ and the lower bound follows from Lemma 12.7. □

12.3 Upper bound for Testing LTFs

We now turn to the problem of testing linear threshold functions (LTFs) in the active property testing model. For this section and the next one, we leave the setting of boolean functions to explore a slightly more general class of functions: functions of the form $f : \mathbb{R}^n \to \{0,1\}$. The definition of linear threshold functions, or *halfspaces*, in this setting is as follows.

Definition 12.9. The function $f : \mathbb{R}^n \to \{0,1\}$ is a *linear threshold function* if there exist $n+1$ parameters $w_1, \ldots, w_n, t \in \mathbb{R}$ such that for every $x \in \mathbb{R}^n$,

$$f(x) = \operatorname{sgn}(w_1 x_1 + \cdots + w_n x_n - t),$$

where the function $\operatorname{sgn} : \mathbb{R} \to \{0,1\}$ returns $\operatorname{sgn}(z) = 1$ iff $z \geq 0$.

We will study the problem of testing linear threshold functions over the standard n-dimensional Gaussian distribution $\mathcal{N}_n(0, I)$ over $\mathbb{R}^n$. There is a close connection to this problem and the Hermite decomposition of functions that we introduced in Section 4.3.2. This connection is described in the following key lemma of Matulef, O'Donnell, Rubinfeld, and Servedio [77].

Lemma 12.10 (Matulef et al. [77])**.** *There is an explicit continuous function $W : \mathbb{R} \to \mathbb{R}$ with bounded derivative $\|W'\|_\infty \leq 1$ and peak value $W(0) = \frac{2}{\pi}$ such that every linear threshold function $f : \mathbb{R}^n \to \{-1,1\}$ satisfies*

$$\sum_{i=1}^{n} \hat{f}(e_i)^2 = W(\mathop{\mathbf{E}}_{x} f).$$

Moreover, every function $g : \mathbb{R}^n \to \{-1,1\}$ that satisfies

$$\left| \sum_{i=1}^{n} \hat{g}(e_i)^2 - W(\mathop{\mathbf{E}}_{x} g) \right| \leq 4\epsilon^3$$

is ϵ-close to being a linear threshold function.

Lemma 12.10 shows that $\sum_i \hat{f}(e_i)^2$, the sum of the level-one Hermite coefficients of a function, provides a characterization of linear threshold functions. We can thus test linear threshold functions by estimating the value of this expression for a given function. We will do so by using the characterization of the level-1 Hermite weight of a function that we established in Section 4.3.2.

12.3.1 Algorithm

Lemmas 12.10 and 4.18 suggest that linear threshold functions may be tested with the following simple LTFTESTER algorithm.

This algorithm queries the value of f on all the samples in S, so it makes a total of m queries. The value $\tilde{\mu}$ is an unbiased estimate of $\mathbf{E} f$ and, since the items of S are

```
LTFTESTER($f$, $\epsilon$)
    1. Draw a set $S \subset \mathbb{R}^n$ of $m$ samples indep. from $\mathcal{N}_n(0, I)$.
    2. Set $\tilde{\mu} = \mathbf{E}_{x \in S}[f(x)]$.
    3. Set $\tilde{\nu} = \mathbf{E}_{x \neq y \in S}[f(x)f(y)\langle x, y\rangle]$.
    4. Accept iff $|\tilde{\nu} - W(\tilde{\mu})| \leq 2\epsilon^3$.
```

drawn independently, we can use standard Chernoff bound arguments to show that $m = O(\sqrt{n})$ samples are sufficient to guarantee that this estimate has very small error with high probability.

The value $\tilde{\nu}$, by Lemma 4.18, is an unbiased estimate of $\sum_{i=1}^{n} \hat{f}(e^i)^2$. The values of $f(x)f(y)\langle x, y\rangle$ are no longer all independent, so we cannot use standard Chernoff bounds directly to bound the error of $\tilde{\nu}$. But the value $\tilde{\nu}$ is a U-statistic of order two, so we can use the concentration of measure tools from Section 4.1.3 to bound this error. In order to do so, however, it is convenient to modify the definition of $\tilde{\nu}$ slightly so that it is bounded. The resulting algorithm is described in Figure 12.1.

```
LTFTESTER*($f$, $\epsilon$)
Parameters: $\tau = \sqrt{4n \log(4n/\epsilon^3)}$, $m = 800\tau/\epsilon^3 + 32/\epsilon^6$.
    1. Draw a set $S \subset \mathbb{R}^n$ of $m$ samples indep. from $\mathcal{N}_n(0, I)$.
    2. Set $\tilde{\mu} = \mathbf{E}_{x \in S}[f(x)]$.
    3. Set $\tilde{\nu} = \mathbf{E}_{x \neq y \in S}[f(x)f(y)\langle x, y\rangle \cdot \mathbf{1}[|\langle x, y\rangle| \leq \tau]]$.
    4. Accept iff $|\tilde{\nu} - W(\tilde{\mu})| \leq 2\epsilon^3$.
```

Figure 12.1: Algorithm for testing linear threshold functions in the active testing model.

12.3.2 Analysis of the algorithm

To verify the correctness of the LTFTESTER* algorithm, we must establish two facts: that $\tilde{\mu}$ and $\tilde{\nu}$ are close enough to the value they estimate, and that $\mathbf{E}\,\tilde{\nu}$ is close enough to the true value $\mathbf{E}\, f(x)f(y)\langle x, y\rangle$ that we want to compute. We begin with this latter task.

For a function $f : \mathbb{R}^n \to \mathbb{R}$, define $\psi_f : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ to be $\psi_f(x, y) = f(x)f(y)\langle x, y\rangle$.

Let $\psi_f^* : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be the truncation of ψ_f defined by setting

$$\psi_f^*(x, y) = \begin{cases} f(x)f(y)\,\langle x, y\rangle & \text{if } |\,\langle x, y\rangle\,| \le \sqrt{4n\log(4n/\epsilon^3)} \\ 0 & \text{otherwise.} \end{cases}$$

The next lemma shows that $\mathbf{E}\,\psi_f^*$ and $\mathbf{E}\,\psi_f$ are close.

Lemma 12.11. *For any function $f : \mathbb{R}^n \to \mathbb{R}$, $|\,\mathbf{E}\,\psi_f - \mathbf{E}\,\psi_f^*| \le \frac{1}{2}\epsilon^3$.*

Proof. For notational clarity, fix $\tau = \sqrt{4n\log(4n/\epsilon^3)}$. By the definition of ψ_f and ψ_f^* and with the trivial bound $|f(x)f(y)\,\langle x, y\rangle\,| \le n$ we have

$$\begin{aligned} |\,\mathbf{E}\,\psi_f - \mathbf{E}\,\psi_f^*| &= \left| \Pr_{x,y}\big[\,|\langle x, y\rangle| > \tau\big] \cdot \mathop{\mathbf{E}}_{x,y}\Big[f(x)f(y)\,\langle x, y\rangle \;\big|\; |\langle x, y\rangle| > \tau\Big]\right| \\ &\le n \cdot \Pr_{x,y}\big[\,|\langle x, y\rangle| > \tau\big]. \end{aligned}$$

The right-most term can be bounded with a standard Chernoff argument. By Markov's inequality and the independence of the variables $x_1, \ldots, x_n, y_1, \ldots, y_n$,

$$\Pr_{x,y}\big[\,\langle x, y\rangle > \tau\big] = \Pr\big[e^{t\langle x,y\rangle} > e^{t\tau}\big] \le \frac{\mathbf{E}\,e^{t\langle x,y\rangle}}{e^{t\tau}} = \frac{\prod_{i=1}^n \mathbf{E}\,e^{tx_iy_i}}{e^{t\tau}}.$$

The moment generating function of a standard normal random variable is $\mathbf{E}\,e^{ty} = e^{t^2/2}$, so

$$\mathop{\mathbf{E}}_{x_i,y_i}\big[e^{tx_iy_i}\big] = \mathop{\mathbf{E}}_{x_i}\big[\mathop{\mathbf{E}}_{y_i} e^{tx_iy_i}\big] = \mathop{\mathbf{E}}_{x_i} e^{(t^2/2)x_i^2}.$$

When $x \sim \mathcal{N}(0, 1)$, the random variable x^2 has a χ^2 distribution with 1 degree of freedom. The moment generating function of this variable is $\mathbf{E}\,e^{tx^2} = \sqrt{\frac{1}{1-2t}} = \sqrt{1 + \frac{2t}{1-2t}}$ for any $t < \frac{1}{2}$. Hence,

$$\mathop{\mathbf{E}}_{x_i} e^{(t^2/2)x_i^2} \le \sqrt{1 + \frac{t^2}{1 - t^2}} \le e^{\frac{t^2}{2(1-t^2)}}$$

for any $t < 1$. Combining the above results and setting $t = \frac{\tau}{2n}$ yields

$$\Pr_{x,y}\big[\,\langle x, y\rangle > \tau\big] \le e^{\frac{nt^2}{2(1-t^2)} - t\tau} \le e^{-\frac{\tau^2}{4n}} = \frac{\epsilon^3}{4n}.$$

The same argument shows that $\Pr[\langle x, y\rangle < -\tau] \le \frac{\epsilon^3}{4n}$ as well. □

We can now complete the analysis of the LTFTESTER* algorithm and the proof of the upper bound in Theorem 12.2.

Theorem 12.12 (Upper bound in Theorem 12.2, restated)**.** *There is an active tester for linear threshold functions over the standard n-dimensional Gaussian distribution that makes $O(\sqrt{n \log n})$ queries.*

Proof. Consider the LTFTESTER* algorithm. When the estimates $\tilde{\mu}$ and $\tilde{\nu}$ satisfy

$$|\tilde{\mu} - \mathbf{E} f| \le \epsilon^3 \qquad \text{and} \qquad |\tilde{\nu} - \mathbf{E}[f(x)f(y)\langle x, y\rangle]| \le \epsilon^3,$$

Lemmas 12.10 and 4.18 guarantee that the algorithm correctly distinguishes LTFs from functions that are far from LTFs. To complete the proof, we must therefore show that the estimates are within the specified error bounds with probability at least $2/3$.

The values $f(x^1), \ldots, f(x^m)$ are independent $\{-1, 1\}$-valued random variables. By Hoeffding's inequality,

$$\Pr[|\tilde{\mu} - \mathbf{E} f| \le \epsilon^3] \ge 1 - 2e^{-\epsilon^6 m/2} = 1 - 2e^{-O(\sqrt{n})}.$$

The estimate $\tilde{\nu}$ is a U-statistic with kernel ψ_f^*. This kernel satisfies

$$\|\psi_f^* - \mathbf{E}\,\psi_f^*\|_\infty \le 2\|\psi_f^*\|_\infty = 2\sqrt{4n \log(4n/\epsilon^3)}.$$

Let $\Sigma^2 = \mathbf{E}_x[\mathbf{E}_y[\psi_f^*(x,y)]^2] - \mathbf{E}_{x,y}[\psi_f^*(x,y)]^2$. Then

$$\Sigma^2 \le \mathop{\mathbf{E}}_y\big[\mathop{\mathbf{E}}_x[\psi_f^*(x,y)]^2\big] = \mathop{\mathbf{E}}_y\big[\mathop{\mathbf{E}}_x[f(x)f(y)\langle x, y\rangle\,\mathbf{1}[|\langle x, y\rangle| \le \tau]]^2\big].$$

Applying the Cauchy-Schwartz inequality to the expression for Σ^2 gives

$$\begin{aligned}\Sigma^2 &\le \mathop{\mathbf{E}}_y\big[\mathop{\mathbf{E}}_x[f(x)f(y)\langle x, y\rangle]^2\big] \\ &= \mathop{\mathbf{E}}_y\Big[\Big(\sum_{i=1}^n f(y)y_i \mathop{\mathbf{E}}_x[f(x)x_i]\Big)^2\Big] \\ &= \sum_{i,j} \hat{f}(e_i)\hat{f}(e_j)\mathop{\mathbf{E}}_y[y_i y_j] = \sum_{i=1}^n \hat{f}(e_i)^2.\end{aligned}$$

By Parseval's identity, we have $\sum_i \hat{f}(e_i)^2 \le \|\hat{f}\|_2^2 = \|f\|_2^2 = 1$. We can now apply Arcones' Theorem (Theorem 4.7) and Lemma 12.11 to obtain

$$\Pr[|\tilde{\nu} - \mathbf{E}\,\psi_f| \le \epsilon^3] = \Pr[|\tilde{\nu} - \mathbf{E}\,\psi_f^*| \le \tfrac{1}{2}\epsilon^3] \ge 1 - 4e^{-\frac{mt^2}{8 + 200\sqrt{n\log(4n/\epsilon^3)}t}} \ge \tfrac{11}{12}.$$

The union bound completes the proof of correctness. □

12.4 Lower Bound for Testing LTFs

The last section showed that it is possible to test linear threshold functions with $O(\sqrt{n})$ queries in the active testing model. The lower bound in Section 12.2 on the number of queries needed to distinguish dictator functions from random functions implies that $\Omega(\log n)$ queries are required to test linear threshold functions in the active model.

In this section, we prove a better lower bound showing that the number of queries required to test linear threshold functions in the active model must be polynomial in n. To establish this lower bound, we begin by extending Lemma 12.7 to real-valued functions.

Lemma 12.13. *Fix a property $\mathcal{P} \subseteq \mathbb{R}^n \to \mathbb{R}$. Let $\mathcal{D}_{\text{yes}}$ be a distribution over $\mathcal{P}$ and let $\mathcal{D}_{\text{no}}$ be a distribution such that $f \sim \mathcal{D}_{\text{no}}$ is ϵ-far from $\mathcal{P}$ with probability $1 - o(1)$. For any set $X \subset \mathbb{R}^n$ of size $|X| = q$, let $p_{X,\text{yes}}, p_{X,\text{no}} : \mathbb{R}^q \to \mathbb{R}$ be the probability density functions for the random variable $f(X)$ when $f \sim \mathcal{D}_{\text{yes}}$ and $f \sim \mathcal{D}_{\text{no}}$, respectively. Let $S \subset \mathbb{R}^n$ be a set of s samples drawn independently from the n-dimensional Gaussian distribution $\mathcal{N}_n(0, I)$. Suppose that with probability at least $\frac{5}{6}$ (over the choice of S), every subset $X \subseteq S$ of size $|X| = q$ satisfy*

$$\Pr_{r \sim \mathcal{N}_q(0,I)} \left[p_{X,\text{yes}}(r) < \tfrac{6}{5} p_{X,\text{no}}(r) \right] > \tfrac{3}{4}.$$

Then there is no s-sample q-query ϵ-tester for $\mathcal{P}$.

The proof of Lemma 12.13 is essentially identical to that of Lemma 12.7. We refer the reader to [13] for the details.

We now complete the proof of the lower bound in Theorem 12.2.

Theorem 12.14 (Upper bound of Theorem 12.2 restated)**.** *At least $\Omega(n^{1/3})$ queries are required to test linear threshold functions over the standard n-dimensional Gaussian distribution in the active testing model.*

Proof. We begin by introducing two distributions $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$ over linear threshold functions and functions that (with high probability) are far from linear threshold functions, respectively. We draw a function f from $\mathcal{D}_{\text{yes}}$ by first drawing a vector $\mathbf{w} \sim \mathcal{N}(0, I_{n \times n})$ from the n-dimensional standard normal distribution. We then define $f : x \mapsto \text{sgn}(\frac{1}{\sqrt{n}} x \cdot \mathbf{w})$. To draw a function g from $\mathcal{D}_{\text{no}}$, we define $g(x) = \text{sgn}(y_x)$ where each y_x variable is drawn independently from the standard normal distribution $\mathcal{N}(0, 1)$.

Let $X \in \mathbb{R}^{n \times q}$ be a random matrix obtained by drawing q vectors from the n-dimensional normal distribution $\mathcal{N}(0, I_{n \times n})$ and setting these vectors to be the columns of X. Equivalently, X is the random matrix whose entries are independent standard normal variables.

When we view X as a set of q queries to a function $f \sim \mathcal{D}_{\text{yes}}$ or a function $g \sim \mathcal{D}_{\text{no}}$, we get $f(X) = \text{sgn}(\frac{1}{\sqrt{n}} X\mathbf{w})$ and $g(X) = \text{sgn}(y_X)$. Note that $\frac{1}{\sqrt{n}} X\mathbf{w} \sim \mathcal{N}(0, \frac{1}{n}X^*X)$ and $y_X \sim \mathcal{N}(0, I_{q\times q})$. To apply Lemma 12.13 it suffices to show that the ratio of the pdfs for both these random variables is bounded by $\frac{6}{5}$ for all but $\frac{1}{5}$ of the probability mass.

The pdf $p : \mathbb{R}^q \to \mathbb{R}$ of a q-dimensional random vector from the distribution $\mathcal{N}_{q\times q}(0, \Sigma)$ is

$$p(x) = (2\pi)^{-\frac{q}{2}} \det(\Sigma)^{-\frac{1}{2}} e^{-\frac{1}{2}x^T\Sigma^{-1}x}.$$

Therefore, the ratio function $r : \mathbb{R}^q \to \mathbb{R}$ between the pdfs of $\frac{1}{\sqrt{n}} X\mathbf{w}$ and of y_X is

$$r(x) = \det(\tfrac{1}{n}X^*X)^{-\frac{1}{2}} e^{\frac{1}{2}x^T((\frac{1}{n}X^*X)^{-1}-I)x}.$$

Note that

$$x^T((\tfrac{1}{n}X^*X)^{-1} - I)x \leq \|(\tfrac{1}{n}X^*X)^{-1} - I\|\|x\|_2^2 = \|\tfrac{1}{n}X^*X - I\|\|x\|_2^2,$$

so by Lemma 4.8 with probability at least $1 - 2e^{-t^2/2}$ we have

$$r(x) \leq e^{\frac{q}{2}\left(\frac{(\sqrt{q}+t)^2}{n}+2\frac{\sqrt{q}+t}{\sqrt{n}}\right)+3\frac{\sqrt{q}+t}{\sqrt{n}}\|x\|_2^2}.$$

By a union bound, for $U \sim \mathcal{N}(0, I_{n\times n})^u$, $u \in \mathbb{N}$ with $u \geq q$, the above inequality for $r(x)$ is true for all subsets of U of size q, with probability at least $1 - u^q 2e^{-t^2/2}$. Fix $q = n^{\frac{1}{3}}/(50(\ln(u))^{\frac{1}{3}})$ and $t = 2\sqrt{q\ln(u)}$. Then $u^q 2e^{-t^2/2} \leq 2u^{-q}$, which is $< 1/4$ for any sufficiently large n. When $\|x\|_2^2 \leq 3q$ then for large n, $r(x) \leq e^{74/625} < \frac{6}{5}$. To complete the proof, it suffices to show that when $x \sim \mathcal{N}(0, I_{q\times q})$, the probability that $\|x\|_2^2 > 3q$ is at most $\frac{1}{5}2^{-q}$. The random variable $\|x\|_2^2$ has a χ^2 distribution with q degrees of freedom and expected value $\mathbf{E}\,\|x\|_2^2 = \sum_{i=1}^q \mathbf{E}\,x_i^2 = q$. Standard concentration bounds for χ^2 variables imply that

$$\Pr_{x\sim\mathcal{N}(0,I_{q\times q})}[\|x\|_2^2 > 3q] \leq e^{-\frac{4}{3}q} < \tfrac{1}{5}2^{-q},$$

as we wanted to show, and the theorem follows from Lemma 12.13. □

12.5 Notes and Discussion

Other results in active testing. The manuscript [13] contains other results on active testing that we did not discuss in this chapter. Notably, it is shown there that the property of being a union of d intervals can be tested with a constant number of queries in the active testing model. This is in contrast to the problem of learning unions of d intervals, which requires $\Omega(d)$ queries in the active learning model.

The active testing results also show that it is possible to test common assumptions from the semi-supervised learning model with a constant number of queries. In particular, testing cluster assumptions and margin assumptions can both be done with a constant number of queries. For the details and more background, see [13].

Active testing vs. distribution-free testing. The active testing model that we introduce in this chapter is not the first alternative model of property testing that was proposed to better mirror realistic learning models. Notably, Halevy and Kushilevitz [63] introduced a *distribution-free* model of property testing that has since been studied extensively [61, 62, 56, 48]. In the distribution-free model, the tester can sample inputs from some *unknown* distribution and can query the target function on *any* input of its choosing. It must then distinguish between the case where $f \in \mathcal{P}$ from the case where f is far from the property over the (unknown) distribution.

If we consider distributions with support size that is polynomial in the dimension of the function being tested, the distribution-free testing model appears to be similar to our active testing model. Indeed, in this case the distance of the input function to a fixed property $\mathcal{P}$ depends only on the values of the function on inputs in the support of the distribution. So intuitively it seems that the tester might as well only query the value of the function on inputs in the support of the distribution. This intuition is wrong: in fact, most testers in the distribution-free model strongly rely on the ability to query any input of their choosing. As a result, the task of testing a property in the active model is very different from the task of testing the same property in the distribution-free model and, indeed, many properties of boolean functions have very different query complexities in the two models.

Bibliography

[1] José A. Adell and Pedro Jodrá. Exact Kolmogorov and total variation distances between some familiar discrete distributions. *Journal of Inequalities and Applications*, 2006. 4.4

[2] Rudolf Ahlswede and Levon H. Khachatrian. The complete intersection theorem for systems of finite sets. *European Journal of Combinatorics*, 18:125–136, 1997. 4.2.1

[3] Noga Alon and Eric Blais. Testing boolean function isomorphism. *Proc. 14th International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 394–405, 2010. (document)

[4] Noga Alon, Eric Blais, Sourav Chakraborty, David García Soriano, and Arie Matsliah. Nearly tight bounds for testing function isomorphism, 2011. Manuscript. (document), 9.3

[5] Noga Alon, Eldar Fischer, Ilan Newman, and Asaf Shapira. A combinatorial characterization of the testable graph properties: It's all about regularity. *SIAM Journal on Computing*, 39:143–167, 2009. 8

[6] Noga Alon and Asaf Shapira. A characterization of the (natural) graph properties testable with one-sided error. In *Proc. of the 46th IEEE Symposium on Foundations of Computer Science*, pages 429–438, 2005. 8

[7] Noga Alon and Asaf Shapira. Every monotone graph property is testable. In *Proc. of the 46th IEEE Symposium on Foundations of Computer Science*, pages 128–137, 2005. 8

[8] Noga Alon and Joel H. Spencer. *The Probabilistic Method.* Wiley, third edition, 2008. 9.1

[9] Miguel A. Arcones. A Bernstein-type inequality for U-statistics and U-processes. *Statistics & Probability Letters*, 22(3):239 – 247, 1995. 4.1.3, 4.7

[10] R. F. Arnold and M. A. Harrison. Algebraic properties of symmetric and partially symmetric functions. *IEEE Transactions on Electronic Computers*, EC-12:244–251, 1963. 6.4

[11] Tim Austin and Terence Tao. Testability and repair of hereditary hypergraph properties. *Random Structures and Algorithms*, 36:373–463, 2010. 8

[12] László Babai, Robert Beals, and Pál Takácsi-Nagy. Symmetry and complexity. In *Proc. of the 24th annual ACM symposium on Theory of computing*, pages 438–449, 1992. 6.4

[13] Maria-Florina Balcan, Eric Blais, Avrim Blum, and Liu Yang. Active testing, 2011. Manuscript. (document), 12.4, 12.5

[14] Mihir Bellare, Don Coppersmith, Johan Håstad, Marcos Kiwi, and Madhu Sudan. Linearity testing in characteristic two. *IEEE Trans. on Information Theory*, 42(6):1781 –1795, 1996. 7

[15] Mihir Bellare, Oded Goldreich, and Madhu Sudan. Free bits, PCPs and non-approximability – towards tight results. *SIAM J. Comput.*, 27(3):804–915, 1998. 5, 5, 7

[16] Mihir Bellare, Shafi Goldwasser, Carsten Lund, and Alexander Russell. Efficient probabilistically checkable proofs and applications to approximations. In *Proc. of the 25th Symposium on Theory of Computing*, pages 294–304, 1993. 5, 7

[17] Mihir Bellare and Madhu Sudan. Improved non-approximability results. In *Proc. of the 26th Symposium on Theory of Computing*, pages 184–193, 1994. 5, 7

[18] Arnab Bhattacharyya, Elena Grigorescu, and Asaf Shapira. A unified framework for testing linear-invariant properties. In *Proc. 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 478–487, 2010. 1.3, 8

[19] Eric Blais. Improved bounds for testing juntas. In *Proc. 12th Workshop RANDOM*, pages 317–330, 2008. (document), 5.3

[20] Eric Blais. Testing juntas nearly optimally. In *Proc. 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 151–158, 2009. (document), 5.3, 7.3, 12

[21] Eric Blais. Testing juntas: a brief survey. In O. Goldreich, editor, *Property Testing: Current Research and Surveys*, pages 32–40. Springer, 2010. 5

[22] Eric Blais, Joshua Brody, and Kevin Matulef. Property testing lower bounds via communication complexity. In *Proc. 26th Annual IEEE Conference on Computational Complexity (CCC)*, pages 210–220, 2011. (document), 7.4

[23] Eric Blais, Joshua Brody, and Kevin Matulef. Property testing lower bounds via communication complexity. *Computational Complexity*, 2012. To appear. (document), 7.4, 11.6

[24] Eric Blais and Daniel Kane. Testing properties of linear functions, 2011. Manuscript. (document), 7.4, 7.4

[25] Eric Blais and Ryan O'Donnell. Lower bounds for testing function isomorphism. In *Proc. 25th Conference on Computational Complexity (CCC)*, pages 235–246, 2010. (document), 10.3

[26] Eric Blais, Amit Weinstein, and Yuichi Yoshida. Partially symmetric functions are isomorphism-testable, 2011. Manuscript. (document), 8, 10.3

[27] Avrim Blum. Relevant examples and relevant features: thoughts from computational learning theory. In *AAAI Fall Symposium on 'Relevance'*, 1994. 5

[28] Avrim Blum. Learning a function of r relevant variables. In *Proc. 16th Conference on Computational Learning Theory*, pages 731–733, 2003. 5

[29] Avrim Blum, Lisa Hellerstein, and Nick Littlestone. Learning in the presence of finitely or infinitely many irrelevant attributes. *J. of Comp. Syst. Sci.*, 50(1):32–40, 1995. 1.3.1, 5, 5.1

[30] Avrim Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(2):245–271, 1997. 5

[31] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *J. Comput. Syst. Sci.*, 47:549–595, 1993. Earlier version in STOC'90. 7

[32] Jean Bourgain. On the distribution of the Fourier spectrum of boolean functions. *Israel Journal of Mathematics*, 131:269–276, 2002. 5

[33] Jop Briët, Sourav Chakraborty, David García Soriano, and Arie Matsliah. Monotonicity testing and shortest-path routing on the cube. In *Proc. 14th International Workshop on Randomization and Approximation Techniques in Computer Science*, 2010. 11.4

[34] Joshua Brody, Kevin Matulef, and Chenggang Wu. Lower bounds for testing computability by small-width branching programs. In *Proc. 8th Annual Theory and Applications of Models of Computation*, 2011. 11.6

[35] Harry Buhrman, Richard Cleve, and Avi Wigderson. Quantum vs. classical communication and computation. In *Proc. 30th Annual ACM Symposium on the Theory of Computing*, pages 63–68, 1998. 11.1.3, 11.6

[36] Samuel H. Caldwell. *Switching Circuits and Logical Design*. Wiley, 1958. 6.4

[37] Sourav Chakraborty, Eldar Fischer, David García Soriano, and Arie Matsliah. Junto-symmetric functions, hypergraph isomorphism, and crunching, 2012. Manuscript. 6.4, 6.4

[38] Sourav Chakraborty, David García Soriano, and Arie Matsliah, 2010. Personal Communication. 7.3

[39] Sourav Chakraborty, David García Soriano, and Arie Matsliah. Efficient sample extractors for juntas with applications. *Automata, Languages and Programming*, pages 545–556, 2011. 5.3, 5.1, 7.3, 8.2

[40] Sourav Chakraborty, David García Soriano, and Arie Matsliah. Nearly tight bounds for testing function isomorphism. In *Proc. 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1683–1702, 2011. (document), 5.3, 7.3, 8, 8.1, 9.3

[41] Hana Chockler and Dan Gutfreund. A lower bound for testing juntas. *Information Processing Letters*, 90(6):301–305, 2004. 5, 7.3

[42] P. Clote and E. Kranakis. Boolean functions, invariance groups, and parallel complexity. *SIAM J. of Computing*, 20:553–590, 1991. 6.4

[43] S.R. Das and C.L. Sheng. On detecting total or partial symmetry of switching functions. *IEEE Trans. on Computers*, C-20(3):352–355, 1971. 6.4

[44] Víctor H. de la Peña and Evarist Giné. *Decoupling: From Dependence to Independence*. Springer, 1999. 4.1.3

[45] Ronald de Wolf. *A Brief Introduction to Fourier Analysis on the Boolean Cube*. Number 1 in Graduate Surveys. Theory of Computing Library, 2008. 2

[46] Ilias Diakonikolas, Homin K. Lee, Kevin Matulef, Krzysztof Onak, Ronitt Rubinfeld, Rocco A. Servedio, and Andrew Wan. Testing for concise representations. In *Proc. 48th Symposium on Foundations of Computer Science*, pages 549–558, 2007. 5, 5.3, 7.3, 7.3.4, 7.7, 12

[47] Irit Dinur and Shmuel Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, 162(1):439–485, 2005. 4.2.1, 4.10, 4.2.1, 5

[48] Elya Dolev and Dana Ron. Distribution-free testing algorithms for monomials with a sublinear number of queries. In *Proceedings of the 13th international conference on Approximation, and 14 the International conference on Randomization, and combinatorial optimization: algorithms and techniques*, APPROX/RANDOM'10, pages 531–544. Springer-Verlag, 2010. 12.5

[49] Paul Erdős, Chao Ko, and Richard Rado. Intersection theorems for systems of finite sets. *The Quarterly Journal of Mathematics*, 12(1):313–320, 1961. 4.2.1

[50] W. Feller. *An introduction to probability theory and its applications*, volume 1. John Wiley & Sons, 1968. 4.1

[51] W. Feller. *An introduction to probability theory and its applications*, volume 2. John Wiley & Sons, 1971. 4.1

[52] Eldar Fischer, Guy Kindler, Dana Ron, Shmuel Safra, and Alex Samorodnitsky. Testing juntas. *J. Comput. Syst. Sci.*, 68(4):753–787, 2004. 1.3.1, 1.3.2, 5, 5.1, 1, 7, 8, 8, 8.1, 9, 10, 12

[53] Peter Frankl. The Erdős-Ko-Rado theorem is true for $n = ckt$. In *Combinatorics (Proc. Fifth Hungarian Colloquium, Keszthely)*, volume 1, pages 365–375, 1976. 4.2.1

[54] Ehud Friedgut. Boolean functions with low average sensitivity depend on few coordinates. *Combinatorica*, 18:474–483, 1998. 5

[55] Ehud Friedgut. On the measure of intersecting families, uniqueness and stability. *Combinatorica*, 28(5):503–528, 2008. 4.2.1, 4.10

[56] Dana Glasner and Rocco A. Servedio. Distribution-free testing lower bound for basic boolean functions. *Theory of Computing*, 5(1):191–216, 2009. 12.5

[57] Oded Goldreich. On testing computability by small width OBDDs. *Proc. 14th International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 574–587, 2010. 1.3.1, 7, 7.3, 7.3.7

[58] Oded Goldreich, editor. *Property Testing: Current Research and Surveys*, volume 6390 of *LNCS*. Springer, 2010. 3

[59] Oded Goldreich, Shari Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. of the ACM*, 45(4):653–750, 1998. 3.2, 12.1

[60] András Hajnal and Endre Szemerédi. Proof of a conjecture of Paul Erdős. In Paul Erdős, Alfréd Rényi, and Vera T. Sós, editors, *Combinatorial Theory and its Applications II*, volume 4, pages 601–623. Colloq. Math. Soc. János Bolyai, 1969. 4.2.2, 4.11, 9.2

[61] Shirley Halevy and Eyal Kushilevitz. Distribution-free connectivity testing. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, volume 3122 of *Lecture Notes in Computer Science*, pages 393–404. Springer Berlin / Heidelberg, 2004. 12.5

[62] Shirley Halevy and Eyal Kushilevitz. A lower bound for distribution-free monotonicity testing. In *Approximation, Randomization and Combinatorial Optimization*, volume 3624 of *Lecture Notes in Computer Science*, pages 612–612. Springer Berlin / Heidelberg, 2005. 12.5

[63] Shirley Halevy and Eyal Kushilevitz. Distribution-free property testing. *SIAM J. Comput.*, 37(4):1107–1138, 2007. 12.5

[64] Paul R. Halmos. The theory of unbiased estimation. *Ann.. Math. Statis.*, 17(1):34–43, 1946. 4.1.3

[65] Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48:798–859, 2001. 5

[66] Wassily Hoeffding. A class of statistics with asymptotically normal distribution. *Ann. of Math. Stat.*, 19(3):293–325, 1948. 4.1.3

[67] Tommy R. Jensen and Bjarne Toft. *Graph Coloring Problems*. Wiley, 1994. 4.2.2

[68] Madhav Jha and Sofya Raskhodnikova. Testing and reconstruction of lipschitz functions with applications to data privacy. In *Proc. 52nd Annual IEEE Symposium on Foundations of Computer Science*, pages 433–442, 2011. 11.6

[69] Gil Kalai. A Fourier-theoretic perspective on the Concordet paradox and Arrows theorem. *Adv. in Applied Math.*, 29:412–426, 2002. 5

[70] Bala Kalyanasundaram and Georg Schnitger. The probabilistic communication complexity of set intersection. *SIAM J. Disc. Math.*, 5(4):547–557, 1992. 11, 11.4

[71] Tali Kaufman, Simon Litsyn, and Ning Xie. Breaking the ϵ-soundness bound of the linearity test over GF(2). *SIAM J. on Computing*, 39:1988–2003, 2010. 7

[72] Tali Kaufman and Madhu Sudan. Algebraic property testing: the role of invariance. In *Proc. 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 403–412, 2008. 1.3, 6.4, 8

[73] M. Kearns and U. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994. 3.2

[74] Michael Kearns and Dana Ron. Testing problems with sublearning sample complexity. *Journal of Computer and System Sciences*, 61(3):428 – 456, 2000. 12

[75] Subhash Khot. On the power of unique 2-prover 1-round games. In *Proc. 34th ACM Symposium on the Theory of Computing*, pages 767–775, 2002. 5

[76] Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997. 11.1

[77] Kevin Matulef, Ryan O'Donnell, Ronitt Rubinfeld, and Rocco Servedio. Testing halfspaces. In *Proc. 20th Symposium on Discrete Algorithms*, 2009. 12, 12.3, 12.10

[78] Michael Molloy and Bruce Reed. *Graph Colouring and the Probabilistic Method.* Springer, 2001. 4.2.2

[79] Elchanan Mossel, Ryan O'Donnell, and Rocco A. Servedio. Learning functions of k relevant variables. *J. Comput. Syst. Sci.*, 69(3):421–434, 2004. 5

[80] Ryan O'Donnell. Some topics in analysis of boolean function. In *Proc. 40th Annual ACM Symposium on the Theory of Computing*, pages 569–578, 2008. 2

[81] Ryan O'Donnell. Analysis of boolean functions, 2012. Available at `analysisofbooleanfunctions.org`. 2

[82] Ryan O'Donnell and Karl Wimmer. KKL, Kruskal-Katona, and monotone nets. In *Proc. 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 725–734, 2009. 6.4

[83] Ryan O'Donnell and Karl Wimmer. Sharpness of KKL on Schreier graphs, 2009. Manuscript. 6.4

[84] Michal Parnas, Dana Ron, and Alex Samorodnitsky. Testing basic boolean formulae. *SIAM J. Discret. Math.*, 16(1):20–46, 2002. 5, 5.3, 7

[85] Toniann Pitassi and Rahul Santhanam. Effectively polynomial simulations. In *Proc. First Symposium on Innovations in Computer Science*, pages 370–382, 2010. 6.4

[86] Alexander Razborov. On the distributional complexity of disjointness. In *Proc. 17thInternational Colloquium on Automata, Languages and Programming*, pages 249–253, 1990. 11, 11.4

[87] Dana Ron. Algorithmic and analysis techniques in property testing. *Foundations and Trends in Theoretical Computer Science*, 5:73–205, 2009. 3

[88] Dana Ron and Gilad Tsur. Testing computability by width-two obdds. *Theoretical Computer Science*, 420:64 – 79, 2012. 7.3.7

[89] Ronitt Rubinfeld and Asaf Shapira. Sublinear time algorithms, 2011. ECCC TR11-013. 3

[90] Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996. 1.2, 3.1

[91] C. Seshadhri and Jan Vondrák. Is submodularity testable? In *Proc. 2nd Innovations in Computer Science*, 2011. 11.6

[92] Claude E. Shannon. The synthesis of two-terminal switching circuits. *Bell System Technical Journal*, 28(1):59–98, 1949. 6.4

[93] Georgi E. Shilov. *Linear Algebra*. Dover, 1977. 4.1.4

[94] Spario Y. T. Soon. Binomial approximation for dependent indicators. *Statistica Sinica*, 6:703–714, 1996. 4.3

[95] Madhu Sudan. Invariance in property testing. In O. Goldreich, editor, *Property Testing: Current Research and Surveys*, pages 211–227. Springer, 2010. 1.3, 6.4, 8

[96] Gábor Szegő. *Orthogonal Polynomials*, volume 23 of *Colloquium Publications*. AMS, fourth edition, 1975. 4.3.1

[97] Jacobus H. van Lint. *Introduction to Coding Theory*, volume 86 of *Graduate Texts in Mathematics*. Springer, third edition, 1999. 4.3, 4.3.1

[98] Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. In Y. Eldar and G. Kutyniok, editors, *Compressed Sensing: Theory and Applications*, chapter 5, pages 210–268. Cambridge University Press, 2012. Available at `http://arxiv.org/abs/1011.3027`. 4.1.4, 4.8, 4.1.4

[99] Richard M. Wilson. The exact bound in the Erdős-Ko-Rado theorem. *Combinatorica*, 4(2–3):247–257, 1984. 4.2.1

[100] Andrew C. Yao. Probabilistic computations: towards a unified measure of complexity. In *Proc. 18th Sym. on Foundations of Comput. Sci.*, pages 222–227, 1977. 3.3, 9.1, 10.1, 12.2