Advances in Quantum Computational Learning Theory

Alp Atıcı

Submitted in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

Columbia University

2006

ABSTRACT

Advances in Quantum Computational Learning Theory

Alp Atıcı

This dissertation explores results at the intersection of two important branches of theoretical computer science: *quantum computation*, which studies the power of computing devices based on quantum physical phenomena, and *computational learning theory*, which studies the foundations of machine learning algorithms. We refer to the area of research at this intersection as *quantum computational learning theory*. Like its classical counterpart, quantum computational learning theory aims to understand the computational and information theoretic requirements of learning problems and algorithms. However unlike the classical models, the models for quantum learners generally involve quantum sources of information and quantum gates employing quantum physical phenomena such as superposition and entanglement, which do not have classical equivalents.

Our results can be summarized as follows:

- In Chapter 3, we study the information theoretic requirements of quantum exact learning, partition learning and "Probably Approximately Correct" (PAC) learning. First, we develop a new general quantum exact learning algorithm, resolving a conjecture by Hunziker *et al.* [HMP$^+$03]. Next, we present positive and negative results towards the rather general problem of partition learning of which both "exact learning" and "computing a binary property" are special cases. Finally, we derive an improved lower bound on the number of quantum examples required for PAC learning.

- In Chapter 4, we consider the problem of learning unions of high dimensional rectangles over the domain $[b]^n$ using membership queries and uniform quantum examples. These classes are natural generalizations of classes of DNF formulae over $\{0,1\}^n$ that have been extensively studied in the learning theory literature.

- In Chapter 5, we develop quantum algorithms for learning and testing juntas (Boolean functions that depend on a few variables) and learning sparse functions. These algorithms are based on the quantum subroutine due to Bshouty and Jackson [BJ99] that enables sampling from the Fourier spectrum.

Contents

List of Algorithms

ACKNOWLEDGEMENTS

I would like to take this opportunity to acknowledge and thank people who have contributed to the development of this thesis and shaped my life as a graduate student in Columbia.

First and foremost, I am deeply grateful to my advisor, Rocco A. Servedio. It has been a great pleasure to work with him in this project. Rocco has always been very patient and generous with his time. His enthusiasm and optimism have been an inspiration and his insights and advice invaluable. His influence is very much present in every page of this dissertation.

I would like to thank Dave Bayer for his confidence and active support in the department, without which this work never would have come to fruition. I thank Shou-Wu Zhang for his advice during my early graduate years. I also thank the members of my defense committee: Lisa Hellerstein, Ronnitt Rubinfeld, Dylan P. Thurston.

I thank my most wonderful friends who have made my years in Columbia so enjoyable and memorable: Johan and Philip (for the brilliant conversations around nutella and tea), Matt and P. J. (for all the senseless fun), John (for the milk shakes and the help for my future career), Yano and Bart (for the out-of-the-box discussions), Eun-Jung and Jan-Willem (for inviting me as a special guest to their home gatherings), Gülru (for her warm company and the artistic advice), and many others including friends in Columbia: Jeff, Eric, Sonja, Keiko, Eli, Sharon, Özge & Sercan, Özgür & Sinem as well as my high school and college friends studying in the States: Umut, Cantürk, Barış & Zeynep, Anıl & Zeynep, Ayça, Mahir & Emek, Erhan, Devin and Mustafa.

Finally, I thank my family for their constant affection and support during this (in their eyes a remarkable but a rather curious) endeavor including my aunts, my grandmother and my cousins but especially my uncle and my mother to whom this thesis is dedicated.

To my mother, Mine

Chapter 1

Introduction

1.1 Motivation

It is relatively recent in the history of computation when Richard P. Feynman first came to realize [Fey82] that it is possible to build more efficient computing devices than the conventional Turing machine by harnessing a deeper knowledge of physics giving rise to the field of *quantum computation*. The emerging field of quantum computation has elicited immense interest after the discovery of Shor's Algorithm [Sho94] both because its study could give rise to other breakthroughs in theory of computation and also because the construction of a practical quantum computer has remained a big challenge over the years.

On the other hand, since Valiant's seminal paper "A Theory of the Learnable" [Val84], the field of *computational learning theory* has evolved into a rich mathematical theory for the study of machine learning algorithms (see e.g. [AB97, KV94, Vap98]). Although the methods and aims of computational learning theory are abstract, techniques and insights from this field have had a great impact on both positive and negative directions in applied machine learning research and real-world learning systems. Many fertile connections have been established between computational learning theory and other research areas in theoretical computer science such as complexity theory, cryptography and computational geometry, to name a few.

The main motivation behind the research in this dissertation is to investigate and derive results at the intersection of these two fields that have so far mostly remained disjoint. The theoretical

motivations arise most significantly from the fact that many natural questions in quantum computational learning theory are closely related to well-studied problems in the theory of quantum computation. Moreover the results in this field could improve our understanding of abilities and limitations in classical learning algorithms. Such a study is also motivated from a practical viewpoint since the development of efficient quantum learning algorithms may have a broad range of potential applicability when quantum computers are realized.

As in the classical computational learning theory, the main factors considered by quantum computational learning theory are computational and information theoretic requirements of learning problems and algorithms. However unlike the classical learning models, the models for quantum learners generally permit quantum gates as basic steps of computation as well as oracles that provide quantum information involving a superposition of concept values.

In this research, we investigate quantum computational learning theory primarily along the following directions:

Exploring the intrinsic limitations of quantum learning algorithms: Information theoretic lower bounds have been established in widely studied classical models of computational learning theory such as the Probably Approximately Correct model, the model of exact learning from membership queries and the model of learning from membership and equivalence queries. Many such lower bounds are known to be tight due to complementing algorithms whose consumption match these bounds. Since these bounds apply to all possible learning algorithms, one has a good overall understanding of the limitations of algorithms in the aforementioned classical models. However the techniques employed in establishing such lower bounds do not extend to the quantum setting in most of the cases. In general, little was known about possible quantum analogues of these bounds. We make considerable progress towards improving existing lower bounds and establishing new ones particularly in Chapter 3 but also in Chapter 5.

Designing more efficient algorithms exploiting inherent capabilities of quantum models: There are concrete examples of concept classes for which quantum algorithms are known to be more efficient in learning and property testing. However for most well-studied concept classes it is not known whether the standard quantum techniques (or possibly newer ones)

could be employed to give a reasonable improvement. One of the goals of this research is to design explicit quantum algorithms whose time and information-theoretic efficiency come close to or match the established quantum lower bounds. We present results in Chapters 3 and 5 towards this goal.

The design of new and more efficient algorithms is a constructive challenge and immediately contributes to our understanding of what *is really* possible within the quantum models. Surprisingly some of these new quantum algorithms are known to perform better than all classical algorithms since their consumption is below the established classical lower bounds.

Development of efficient algorithms towards learning more expressive concept classes: A natural question is whether it is possible to develop computationally efficient quantum algorithms towards learning more expressive concept classes for which no efficient classical algorithm is known. In particular, Chapter 4 presents new classical and quantum results along this direction for the problem of learning unions of high dimensional rectangles.

1.2 Outline and summary of the contributions

Background: In this chapter we present a description of quantum computation, a brief explanation of computational learning theory and some further mathematical preliminaries.

Improved Bounds on Quantum Learning Algorithms: This chapter is based on the article [AS05] which appeared in the Journal of Quantum Information Processing.

A major focus of study in quantum computation is the power of quantum algorithms to extract information from a "black-box" oracle for an unknown Boolean function. Many of the most powerful ideas for both algorithmic results and lower bounds in quantum computing have emerged from this framework, which has been studied for more than a decade (see e.g. [BBBV97, Gro96, Sim97, BBC$^+$01, SG04, AIK$^+$04a, BJ99]).

In this chapter we give several new results on the complexity of algorithms that learn Boolean functions from quantum queries and quantum examples, which are the most widely studied quantum oracles.

- Hunziker *et al.* [HMP$^+$03] conjectured that for any class C of Boolean functions, the number of quantum black-box queries which are required to exactly identify an unknown function from C is $O(\frac{\log |C|}{\sqrt{\hat{\gamma}^C}})$, where $\hat{\gamma}^C$ is a combinatorial parameter of the class C.

  We essentially resolve this conjecture in the affirmative by giving a quantum algorithm that, for any class C, identifies any unknown function from C using $O(\frac{\log |C| \log\log |C|}{\sqrt{\hat{\gamma}^C}})$ quantum black-box queries.

- We consider a range of natural problems intermediate between the exact learning problem (in which the learner must obtain all bits of information about the black-box function) and the usual problem of computing a predicate (in which the learner must obtain only one bit of information about the black-box function). We give positive and negative results on when the quantum and classical query complexities of these intermediate problems are polynomially related to each other.

- Finally, we improve the known lower bounds on the number of quantum examples (as opposed to quantum black-box queries) required for (ϵ, δ)-PAC learning any concept class of Vapnik-Chervonenkis dimension d over the domain $\{0,1\}^n$ from $\Omega(\frac{d}{n})$ to $\Omega(\frac{1}{\epsilon}\ln\frac{1}{\delta} + d + \frac{\sqrt{d}}{\epsilon})$. This new lower bound comes closer to matching known upper bounds for classical PAC learning.

These results address the information theoretic requirements of the listed problems.

Learning Unions of $\omega(1)$-Dimensional Rectangles: This chapter is based on the article [AS06], chosen as the recipient of the E. M. Gold award by the committee of the 17th International Conference on Algorithmic Learning Theory. It will appear in the journal Theoretical Computer Science special issue ALT 2006.

The learnability of Boolean valued functions defined over the domain $[b]^n = \{0, 1, \ldots, b-1\}^n$ has long elicited interest in computational learning theory literature. In particular, much research has been done on learning various classes of "unions of rectangles" over $[b]^n$ (see e.g. [BK98, CH96, CM94, GGM94, Jac97, MW98]), where a rectangle is a conjunction of properties of the form "the value of attribute x_i lies in the range $[\alpha_i, \beta_i]$". One motivation for studying these classes

is that they are a natural analogue of classes of DNF formulae over $\{0,1\}^n$.

In this chapter we consider the problem of learning unions of rectangles over the domain $[b]^n$, in the uniform distribution membership query learning setting, where both b and n are "large". We obtain poly$(n, \log b)$-time algorithms for the following classes:

- poly$(n \log b)$-MAJORITY of $O(\frac{\log(n \log b)}{\log\log(n \log b)})$-dimensional rectangles.
- Union of poly$(\log(n \log b))$ $O(\frac{\log^2(n \log b)}{(\log\log(n \log b) \log\log\log(n \log b))^2})$-dimensional rectangles.
- poly$(n \log b)$-MAJORITY of poly$(n \log b)$-OR of disjoint $O(\frac{\log(n \log b)}{\log\log(n \log b)})$-dimensional rectangles.

Our main algorithmic tool is an extension of Jackson's boosting- and Fourier-based Harmonic Sieve algorithm [Jac97] to the domain $[b]^n$, building on work of Akavia *et al.* [AGS03]. Other ingredients used to obtain the results stated above are techniques from exact learning [BK98] and ideas from recent work on learning augmented AC^0 circuits [JKS02] and on representing Boolean functions as thresholds of parities [KS04].

Finally, we explore consequences of these results towards learning with quantum computation. We translate the derived positive learnability results to the uniform dimensional quantum PAC learning model, in which no access to classical or quantum membership queries is permitted.

Quantum Algorithms for Testing and Learning Juntas: In this chapter, we develop quantum algorithms for learning and testing juntas and learning sparse functions.

Our objective is to develop efficient algorithms:

- whose query complexity has no dependence on n, the dimension of the domain the Boolean functions are defined over.
- with no access to any classical or quantum membership queries.
- consuming only a few quantum examples but possibly many classical random examples (which are considered relatively "cheap" compared with quantum examples).

We are primarily interested in the information theoretic requirements of the learning and testing problems that we discuss.

Our quantum algorithms are based on the quantum subroutine FS which permits sampling according to the Fourier spectrum due to Bshouty and Jackson [BJ99]. In particular our results can be summarized as follows:

- A k-junta testing algorithm with $O(k/\epsilon)$ quantum examples.
- We establish the lower bound: Any FS based k-junta testing algorithm requires $\Omega(\sqrt{k})$ queries.
- A k-junta learning algorithm with $O(\epsilon^{-1}k \log k)$ quantum examples and $O(\log(1/\epsilon)2^k)$ random examples.
- A learning algorithm for almost t-sparse functions using $O(\frac{t}{\epsilon} \log \frac{t}{\epsilon})$ quantum examples.

Our learning algorithms come close to the best possible due to complementing lower bounds.

Conclusions: In this chapter we summarize our results and discuss promising directions for future research arising from our work.

Chapter 2

Background

2.1 Quantum computation

Brief history and definition: Richard P. Feynman was the first to note that it is possible to build more efficient computing devices than the conventional Turing machine by harnessing a deeper knowledge of physics. He pointed out in 1982 [Fey82] that it appears to be extremely difficult by using an ordinary computer to simulate efficiently how a quantum physical system evolves with time. He also demonstrated that, if we had a computer that runs according to the laws of quantum physics, then this simulation could be made efficiently. Thus he actually suggested that a *quantum computer* could be essentially more efficient than any traditional one.

In 1985 in his notable paper [Deu85], Deutsch was the first to establish a solid ground for the theory of quantum computation by introducing a fully quantum model for computation and giving the description of a *universal quantum computer*. Later, Deutsch also defined quantum networks in [Deu89]. The construction of a universal quantum Turing machine was improved by Bernstein and Vazirani in [BV97], where the authors show how to construct a universal quantum Turing machine capable of simulating any other quantum Turing machine with polynomial efficiency. But it was after Peter W. Shor introduced his celebrated quantum algorithms for factoring integers and extracting discrete logarithms in polynomial time [Sho94] that the field of quantum computation has elicited immense interest.

A quantum computer is any device for computation that makes direct use of distinctively quan-

tum mechanical phenomena, such as *superposition* and *entanglement*, to perform operations on data. In a classical (or conventional) computer, the basic unit of data is measured by bits; in a quantum computer, it is measured by *qubits*. The basic principle of quantum computation is that the quantum properties of particles can be used to represent and structure data, and that devised quantum mechanisms can be used to perform operations with this data.

In this section we give a brief explanation of the fundamentals of quantum computation. For a more detailed description of the quantum computational model, the reader is referred to textbooks and survey articles such as [KSV02, NC00, BV97, Yao93].

Quantum bits and registers: A *quantum bit*, *qubit* for short, is the basic unit of quantum information. It is a two dimensional quantum system equipped with fixed basic states: $\{|0\rangle, |1\rangle\}$, known as the *computational basis*. Just as a classical bit has a state – either 0 or 1 – the *state* of a single qubit is a vector

$$|\psi\rangle = c_0|0\rangle + c_1|1\rangle, \text{ where } c_0, c_1 \in \mathbb{C} \text{ and } |c_0|^2 + |c_1|^2 = 1. \tag{2.1.1}$$

The convention in the quantum computation literature of using "$|\cdot\rangle$" to describe vectors is called the *Dirac notation*. The complex numbers c_0, c_1 are called the *amplitudes* of the basis states $|0\rangle$ and $|1\rangle$ respectively. Therefore, in contrast to a classical bit whose state is a binary value, a qubit can be in a linear combination of these basic states. Being in such a state of linear combination is referred to as *superposition*. We will represent the *state space* of a qubit by $\mathcal{H}$. Note that $\mathcal{H}$ is a subset of the two dimensional complex vector space spanned by all combinations of $|0\rangle$ and $|1\rangle$.

A *measurement* or an *observation* of a qubit in state (2.1.1) will yield 0 or 1 as the outcome with probabilities $|c_0|^2$ and $|c_1|^2$ respectively. And after a measurement the qubit will always produce the same value in subsequent measurements. Therefore after the qubit is measured, it *collapses* into the state $|i\rangle$ with probability $|c_i|^2$.

Example 2.1.1. When measured the first time, the state $\frac{3}{5}|0\rangle + \frac{4}{5\sqrt{2}}(1-i)|1\rangle$ produces the outcome 0 or 1 with probabilities $9/25$ and $16/25$ respectively. After the measurement, the state of the qubit becomes $|i\rangle$, where i is the outcome of the measurement. Consequently all subsequent measurements yield the same outcome as the first.

A general finite *quantum system* can consist of an arbitrarily large (but finite) number of qubits. Sometimes these qubits are called the *quantum registers* of the system. The state of an n-qubit quantum system is a vector

$$|\psi\rangle = \sum_{(x_1,\ldots,x_n)\in\mathbb{F}_2^n} c_{x_1,\ldots,x_n}|x_1,\ldots,x_n\rangle, \text{ where } c_{x_1,\ldots,x_n} \in \mathbb{C} \text{ and } \sum_{(x_1,\ldots,x_n)\in\mathbb{F}_2^n} |c_{x_1,\ldots,x_n}|^2 = 1. \tag{2.1.2}$$

Therefore the state space for such a system is a subset of the complex vector space of dimension 2^n with the basis

$$\{|x_1,\ldots,x_n\rangle \colon (x_1,x_2,\ldots,x_n) \in \mathbb{F}_2^n\}.$$

This basis is called the computational basis of the quantum system and each $c_{x_1,\ldots,x_n}$ the amplitude of the corresponding basis state $|x_1,\ldots,x_n\rangle$. Consequently, one of the fundamental differences between a classical and a quantum computer is that the state of a quantum computer at a given moment can be in a superposition of the basic states, i.e. the elements of the computational basis.

Similarly, a measurement of the quantum system in state $|\psi\rangle$ will yield $(x_1,\ldots,x_n)$ with probability $|c_{x_1,\ldots,x_n}|^2$. Observe that the measurement probabilities depend on the choice of the computational basis. If a different orthonormal basis for $\mathbb{C}^{2^n}$ were set as the new computational basis, a new set of probabilities would be obtained for the same state $|\psi\rangle$.

The state space of an n qubit quantum system can be expressed as the following tensor product over $\mathbb{C}$:

$$\underbrace{\mathcal{H}\otimes\mathcal{H}\otimes\ldots\otimes\mathcal{H}}_{n \text{ times}} = \bigotimes_{i=1}^{n}\mathcal{H}.$$

Recall that a k-dimensional complex vector space has the inner product and induced norm defined by:

$$\text{for } v=(v_1,\ldots,v_k), w=(w_1,\ldots,w_k)\in\mathbb{C}^k, \langle v,w\rangle = \sum_i \overline{v_i}w_i, \|v\| = \sqrt{\langle v,v\rangle}.$$

This induces an inner product on the state space of n qubit quantum system $\bigotimes_{i=1}^{n}\mathcal{H}$ as follows:

$$\text{if } |\psi\rangle,|\phi\rangle \in \bigotimes_{i=1}^{n}\mathcal{H}, \text{where } |\psi\rangle = \sum_{x\in\mathbb{F}_2^n} c_x|x\rangle, |\phi\rangle = \sum_{x\in\mathbb{F}_2^n} d_x|x\rangle \text{ then } \langle\psi|\phi\rangle = \sum_{x\in\mathbb{F}_2^n}\overline{c_x}d_x.$$

Note that by convention this inner product is expressed as $\langle\psi|\phi\rangle$ instead of $\langle\psi,\phi\rangle$.

Compound quantum systems and entanglement: Indeed, if two quantum systems of n and m qubits with states $|\psi\rangle$ and $|\xi\rangle$ are consolidated into a single system of $n+m$ qubits the final state of the new system will be $|\psi\rangle\otimes|\xi\rangle$ (for notational simplicity the tensor product is sometimes dropped and the state of this system is expressed as $|\psi\rangle|\xi\rangle$). However not every state of an $n+m$ qubit quantum system can be expressed as such a tensor product. In other words, an $n+m$ qubit quantum system can be in a state $|\phi\rangle$ such that

$$\text{there exists no } |\psi\rangle\in\mathcal{H}^{\otimes n}, |\xi\rangle\in\mathcal{H}^{\otimes m}, \text{ for which } |\phi\rangle=|\psi\rangle\otimes|\xi\rangle.$$

In this case the first n and the last m qubits, considered as separate quantum systems, are said to be *entangled* rather than independent. This phenomenon called *entanglement* is quite important in quantum information processing.

Example 2.1.2. The simplest yet arguably the most consequential example is the two qubit state called the *Bell state* or *EPR pair*,

$$\frac{|00\rangle+|11\rangle}{\sqrt{2}}.$$

It is easily proved by contradiction that there are no one qubit states $|\psi\rangle, |\xi\rangle$ whose tensor product is the Bell state. Thus the first and the second qubits are entangled.

Generalized measurements: According to the *projection postulate*, given an n-qubit quantum system $|\psi\rangle=\sum_{x\in\mathbb{F}_2^n} c_x|x\rangle$, the state of the system after observation is $|i\rangle$ with probability $|c_i|^2$. In other words the system is said to collapse into the state $|i\rangle$ with probability $|c_i|^2$. However more generally a partial measurement / observation of qubits is possible. Suppose we have a compound system of $n+m$ qubits in state $|\psi\rangle=\sum_{x\in\mathbb{F}_2^n}\sum_{y\in\mathbb{F}_2^m}\alpha_{x,y}|x\rangle|y\rangle$ and the first n-qubit system is observed in the state $|i\rangle$. Then the projection postulate implies that the post-observation state of the whole system is:

$$\frac{1}{\sqrt{\mathbf{Pr}(i)}}\sum_{y\in\mathbb{F}_2^m}\alpha_{i,y}|i\rangle|y\rangle, \text{where } \mathbf{Pr}(i)=\sum_{y\in\mathbb{F}_2^m}|\alpha_{i,y}|^2.$$

The fact that gives the projection postulate its name is that associated to any measurement in the computational basis, there exists a collection of linear projections $\{P_i \colon P_i^2 = P_i\}$ (where the index i refers to the measurement outcomes) satisfying the following properties:

- The probability that result i is observed is $\mathbf{Pr}(i) = \langle\psi|P_i|\psi\rangle$,
- If result i is observed, the state of the system after the measurement is $\frac{P_i|\psi\rangle}{\sqrt{\langle\psi|P_i|\psi\rangle}}$.
- The linear projections satisfy the *completeness equation*: $\sum_i P_i = I$.

Above, $\langle\psi|P_i|\psi\rangle$ denotes the inner product of the states $|\psi\rangle$ and $P_i|\psi\rangle$.

Example 2.1.3. For the earlier example of measuring the first n qubits of the compound system of $n + m$ qubits we have the following set of projections satisfying the above properties:

$$P_i : \sum_{x\in\mathbb{F}_2^n}\sum_{y\in\mathbb{F}_2^m} \alpha_{x,y}|x\rangle|y\rangle \mapsto \sum_{y\in\mathbb{F}_2^m} \alpha_{i,y}|i\rangle|y\rangle.$$

In other words, during a measurement in the computational basis, the state of the system is projected to the subspace that corresponds to the observed state, and renormalized to the unit length.

In light of the projection postulate we can give an example to how entanglement can give rise to surprising facts. Given two entangled quantum systems no matter how distant from each other, an operation on one system would affect the other.

Example 2.1.4. Consider measuring the first qubit of the Bell state $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$. By the projection postulate, if this measurement is 0 then the post-observation state of the system is $|00\rangle$ and if it is 1 then the post-observation state of the system is $|11\rangle$. Therefore if the first qubit of the Bell state is measured then the second qubit will collapse to the same value as the first no matter how physically distant the particles associated to these qubits could be.

Quantum gates: We have yet to describe the transformations of a quantum computer which correspond to the basic computational steps. Just as a classical computer is composed of Boolean gates, a quantum computer is composed of *quantum gates* acting on bounded number of qubits. As unitary operators describe the evolution of quantum systems, quantum gates are themselves unitary transformations, i.e. linear maps U over a finite dimensional complex vector space satisfying

$U^\dagger U = I$ where $U^\dagger$ denotes the *conjugate transpose* of U. Recall that unitary transformations preserve the inner product over $\mathbb{C}^k$. Therefore a quantum gate applied to an n-qubit system preserves the length $\sum_{x \in \mathbb{F}_2^n} |c_x|^2$ of any vector $\sum_{x \in \mathbb{F}_2^n} c_x |x\rangle$. One can easily verify that composition of unitary transformations is also unitary.

Since these transformations are linear, it's sufficient to describe their action on some basis of the state space. We will usually provide the action over the computational basis states to uniquely identify the quantum gate.

Example 2.1.5. The prototypical multi-qubit quantum gate is the controlled-NOT or CNOT gate. This gate acts on two qubits: the *control* qubit and the *target* qubit. The action of this gate is defined as follows: If the control qubit is set to 0 then the target qubit's value is unaltered; but if the control qubit is set to 1 then the target qubit is flipped. In equations this corresponds to the following action on the computational basis states:

$$\text{CNOT} : |00\rangle \mapsto |00\rangle, |01\rangle \mapsto |01\rangle, |10\rangle \mapsto |11\rangle, |11\rangle \mapsto |10\rangle.$$

In a classical computer	**In a quantum computer**
The gates are functions $\mathbb{F}_2^n \to \mathbb{F}_2^m$ where $n, m = O(1)$.	The gates are unitary operators U over $\mathbb{C}^{2^n}, n = O(1)$ preserving the length $\sum_{x \in \mathbb{F}_2^n} \lvert c_x \rvert^2$ of any vector $\sum_{x \in \mathbb{F}_2^n} c_x \lvert x \rangle$.

Table 2.1: A comparison of the gates between a classical computer and a quantum computer.

Sometimes we will talk about a unitary transformation U over $\mathbb{C}^{2^m}$ to be applied to a subset of qubits $S, |S| = m$ of a larger quantum system. This means we are applying to the quantum system the unitary transformation $U_S \otimes I_{S^c}$, which is the tensor product of linear transformations: U applied to the qubits in S and identity I to the rest of the qubits. It is straightforward to check that tensor products of unitary maps is unitary.

Example 2.1.6. Suppose we have a 3 qubit quantum register and we apply the CNOT gate of Example 2.1.5 to the first and third qubits. Consequently the unitary transformation acting on the quantum register will be: $\text{CNOT}_{[1,3]} \otimes I_{[2]}$. This has the following action on the computational

basis states:

$$\begin{aligned}\text{CNOT}_{[1,3]} \otimes I_{[2]} :&|000\rangle \mapsto |000\rangle, |001\rangle \mapsto |001\rangle, |100\rangle \mapsto |101\rangle, |101\rangle \mapsto |100\rangle,\\ &|010\rangle \mapsto |010\rangle, |011\rangle \mapsto |011\rangle, |110\rangle \mapsto |111\rangle, |111\rangle \mapsto |110\rangle.\end{aligned}$$

We emphasize that the state evolution during a measurement, as described by the projection postulate, *is not consistent with the unitary time evolution*. This is because a unitary evolution described by a unitary transformation U is always reversible via U^{-1}, while there is no way to recover the original state after measurement (see for instance Example 2.1.4).

Summary of the quantum circuit model: The circuit model of quantum computation assumes that one has the ability to prepare states in the computational basis, perform quantum gates and perform measurements in the computational basis.

Moreover, the *principle of deferred measurement* [NC00, pp. 186] implies that given any quantum algorithm composed of a sequence of quantum gates and measurements, it is possible to re-express this algorithm in such a way that there is only one measurement after some sequence of quantum gates. Equivalently, this fact is sometimes stated as "measurements can be delayed until the very end". Therefore w.l.o.g. we will always assume in our algorithms that the measurement is performed at the end.

Hence during any quantum algorithm, the information processing is carried out according to the following scheme:

1. The system is first prepared in an initial basis state.

2. Next, the quantum gates are applied.

3. Finally, the system is observed in the computational basis to determine the outcome.

Also, whenever the ultimate result of a quantum algorithm is known to be binary valued, i.e. either TRUE or FALSE, we will assume without loss of generality that measuring the last qubit of the system will produce the result of the algorithm. This is because all post-measurement computation steps could be moved before the measurement as we discussed. In other words if the state of the

quantum system after all quantum gates are applied is $|\psi\rangle$, the probabilities that TRUE or FALSE will be returned by the algorithm after measurement is $\langle\psi|P_1|\psi\rangle$ and $\langle\psi|P_0|\psi\rangle$ respectively where P_0, P_1 are the projection operators associated to measuring the final qubit in 0 or 1.

On the quantum computational complexity of functions: In order to explain the quantum computational complexity of functions we adopt the following definition for a quantum algorithm.

Definition 2.1.7. Let $F\colon \mathbb{F}_2^* \to \mathbb{F}_2^*$ be the function we are interested in computing. Consider its decomposition into a collection of functions based on the length of the input: $F_n\colon \mathbb{F}_2^n \to \mathbb{F}_2^{m(n)}$.

- A *quantum algorithm* computing a function F is a classical algorithm described by a Turing machine $\mathcal{T}$ that computes a function of the form $n \mapsto Z(n)$, where $Z(n)$ is a description of a quantum circuit which computes F_n at each input $x \in \mathbb{F}_2^n$ (i.e. when the quantum circuit $Z(n)$ is invoked over the initial state $|x\rangle|0\ldots0\rangle$) with high probability.

- We say that quantum algorithm *computes F in time $T(n)$* if building the circuit takes at most $T(n)$ steps. The size of the circuit is obviously not greater than $T(n)$.

- Provided there exists such a Turing machine T, such a collection of circuits $Z(n)$ is said to be *uniformly generated* in time $T(n)$.

- The *computational complexity* of F is defined as $T'(n)$, if there is a quantum algorithm computing F in time $T'(n)$ and there is no quantum algorithm computing F in $T''(n)$ where $T''(n) < T'(n)$ for some $n \geq 1$. The function F is said to belong to the class **BQP** if its computational complexity is $\mathrm{poly}(n)$.

Note that this definition is essentially the same as the classical complexity – the only difference is that the description the Turing machine produces involves a quantum circuit instead of a classical circuit. A subtle technical point we have not explained is how the quantum circuits will be described by the Turing machine. It turns out that just as there exists a finite set of Boolean gates constituting a complete basis in terms of which the classical circuits are described, the quantum circuits can be described in terms of a *finite* set of quantum gates called *the standard basis* up to arbitrarily small accuracy. The reader is referred to [Kit97, DN05], [KSV02, Theorem 8.3], [NC00, Appendix 3] for a detailed treatise on the *Solovay-Kitaev Theorem* elaborating this result.

Theorem 2.1.8 (See [DN05])**.** *Any quantum gate U acting on a bounded number of qubits can be realized with accuracy δ (under the operator norm) by a sequence of $\mathrm{poly}(\log \delta^{-1})$ quantum gates from the standard basis. There is an $\mathrm{poly}(\log \delta^{-1})$ time classical algorithm compiling this circuit on the description of U.*

Therefore given a quantum circuit of size T where each quantum gate acts on $O(1)$ qubits, one can invoke the above theorem with accuracy δ/T on each quantum gate to obtain a description of the quantum circuit with accuracy δ purely in terms of the gates in the standard basis.

Naturally, we expect a quantum computer to be at least as efficient as a classical computer for our efforts to be meaningful. Now we explain how any classical algorithm can be efficiently simulated by a quantum algorithm.

The following fact will be the key idea towards simulating classical gates by quantum gates.

Remark 2.1.9. Any classical gate computing a permutation τ can be realized by a quantum gate mapping $|x\rangle \mapsto |\tau(x)\rangle, x \in \mathbb{F}_2^k$ (note this map is unitary as τ is a permutation) followed by a measurement in the computational basis. In other words, every *reversible* classical gate has an equivalent quantum gate.

The following lemma shows that given any classical circuit over a finite basis of gates with bounded fan-in and fan-out, one can efficiently compute an equivalent circuit consisting only of permutations.

Lemma 2.1.10 (See [KSV02, Section 7])**.** *Let a function $F\colon \mathbb{F}_2^n \to \mathbb{F}_2^m$ be realized by a classical circuit of size L and depth d over some basis of gates $\mathcal{G}$ (the fan-in and fan-out being bounded by a constant). Then there is a classical algorithm running in time $O(L+n+m)$ and producing a reversible classical circuit consisting only of permutations and computing the permutation:*

$$F_\oplus\colon \mathbb{F}_2^{n+m} \to \mathbb{F}_2^{n+m},\ \textit{mapping}\ (x,y) \mapsto (x, y \oplus F(x))$$

(where $\oplus$ denotes bitwise addition modulo 2) with size $O(L+n+m)$ and depth $O(d)$.

The new circuit could be fed with $(x,0)$ to obtain $(x,F(x))$ and consequently $F(x)$.

Suppose we are given a classical algorithm, or equivalently a uniformly generated collection of classical circuits $C(n)$ (expressed in terms of a finite basis of gates with bounded fan-in and fan-out)

by a Turing machine $\mathcal{T}_1$. Then using the above lemma, given the circuit description for each $C(n)$ produced by $\mathcal{T}_1$, there is another Turing machine $\mathcal{T}_2$ efficiently computing an equivalent circuit $D(n)$ consisting only of permutations. Hence, replacing each of these permutations with equivalent quantum gates (as described by Remark 2.1.9) gives rise to a uniformly generated collection of quantum circuits and consequently a quantum algorithm.

Quantum algorithms with queries: In our discussion we will restrict our attention to query based quantum algorithms, in which information about the input is extracted through *queries* to a *quantum oracle*. Sometimes an oracle is referred to as a *black-box* throughout our discussion.

A *quantum circuit (or network) with queries* is a sequence of unitary transformations each of which is either a quantum gate or an oracle query. Each oracle query is a unitary transformation whose action is a function of the input to the circuit, whereas the quantum gates in the circuit are *constant* in the sense that they do not depend on the input function. Also each quantum gate in the circuit acts on $O(1)$ qubits by the definition of a quantum gate, which is not necessarily true for the oracle queries. Consequently, Theorem 2.1.8 implies all quantum gates in such a circuit can be approximated to arbitrary accuracy and replaced by quantum gates from the standard basis, which is a finite set.

Because the information about the input (which will be a Boolean valued function f in our problems) to a quantum circuit with queries is extracted through the queries, the initial state of the system is assumed to be constant. Therefore as presented in [BBC$^+$01], we will assume our quantum system is initially prepared in the state $|0^N\rangle$. This is in contrast to ordinary quantum circuits without oracles where the initial state contains the input to the circuit and the circuit consists of constant unitary transformations.

Example 2.1.11. The *quantum membership oracle* QMQ associated to a function $f\colon \{0,1\}^n \to \{0,1\}$ is defined as follows. Each query to this oracle is a unitary transformation $\mathsf{QMQ}(f)$ acting on a quantum system of $n+1$ qubits:

$$\mathsf{QMQ}(f) : |x\rangle|y\rangle \mapsto |x\rangle|y \oplus f(x)\rangle, \text{ where } x \in \mathbb{F}_2^n, y \in \mathbb{F}_2.$$

Clearly the action of this unitary transformation is a function of f.

Definition 2.1.12. A *quantum algorithm with queries* is a classical algorithm uniformly generating a collection of quantum circuits $Z(n)$ with queries each of which

- consists of quantum gates from the standard basis and queries which are unitary transformations that are functions of $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$;
- is assumed to start in the state $|0^{N(n)}\rangle$.

We say that the quantum algorithm with queries *runs in time* $T(n)$ *using* $Q(n)$ *queries* if building the circuit $Z(n)$ takes at most $T(n)$ steps and $Q(n)$ is the number of queries in the circuit.

Example 2.1.13. *Grover's Algorithm* [Gro96] is a quantum algorithm with membership queries (as defined in Example 2.1.11) where each $Z(n)$ is a quantum circuit

- using $O(2^{n/2})$ membership queries $\mathsf{QMQ}(f)$ where $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$;
- and for any f that is not the identically 0 function, the outcome of the circuit $Z(n)$ is w.h.p. some $x \in \mathbb{F}_2^n$ for which $f(x) = 1$.

2.2 Computational learning theory

Computational learning theory is a mathematical field devoted to studying the design and analysis of algorithms for making predictions about the future based on past experiences. It can also be understood as an attempt to to adapt ideas from complexity theory and analysis of algorithms to develop more concrete and detailed models of learning phenomena.

Since Valiant's seminal paper "A Theory of the Learnable" [Val84] two decades ago, the field of computational learning theory has has matured dramatically, growing in mathematical depth and finding its natural connections to many other disciplines including statistics and cryptography. The models and algorithms of the field have had widespread impact on the practice of machine learning. The potential areas of application of insights and methods from learning theory are very diverse, including such fields as natural language processing, DNA analysis, expert systems, robotics, data mining, user interfaces, software agents, and cognitive science.

Only a brief overview is provided in this section. For further information the reader is encouraged to refer to the textbooks and the survey articles in the field such as [KV94, AB97, Gol99, Vap98] and consult the references therein.

The primary question studied in computational learning theory is identifying or approximating an unknown *target function* based on its input-output behavior. Most of the literature involves Boolean valued functions $c : X \mapsto \{\text{TRUE}, \text{FALSE}\}$ where X is the input domain of possible *instances* and $c(x)$ is the label at instance x. Such Boolean valued functions are referred to as *concepts*.

Associated to every learning problem is a *concept class*, the set of all possible target functions the learning algorithm may be expected to learn: $\mathfrak{C} = \cup_{n \geq 1} C_n$ where C_n consists of those concepts in $\mathfrak{C}$ whose domain is $\{0,1\}^n$. Therefore it is implicit that a learning algorithm for $\mathfrak{C}$ "knows" the class $\mathfrak{C}$ but does not know the identity of the target concept $c \in \mathfrak{C}$.

The learning model defines how the learning algorithm is allowed to obtain information about the target concept. In the classical case the learning algorithm generally has access to an oracle which provides examples $\langle x, c(x) \rangle$ that are labeled according to a fixed but unknown target concept $c \in \mathfrak{C}$. For quantum learners the oracle usually provides quantum information involving a superposition of such examples. As we discuss in later chapters, several different specific instantiations of these oracles have been considered in both the classical and quantum settings.

The most important parameters of a learning algorithm are its *sample* or *query* complexity (how many oracle calls the algorithm requires for successful learning) and its *computational* complexity (the number of basic computation steps the algorithm consumes). Therefore most of the research in computational learning theory literature address the asymptotic requirements for learning the concept class C_n as n is allowed to grow. For ease of notation throughout this dissertation we will omit the subscript in C_n and simply write C to denote a collection of concepts over $\{0,1\}^n$.

In Chapters 3 and 5, the query complexity of learning algorithms will be our main concern, whereas in Chapter 4 we are interested in the computational complexity. In the preliminaries section of each of those chapters, detailed information will be provided about the precise learning models and problems involved in the corresponding chapter.

2.3 Mathematical background

In this section we provide basic mathematical tools and conventions that are common to all chapters of the dissertation. Other relevant tools like Fourier analysis and Boosting will be introduced in the subsequent chapters as needed.

The following well-known bounds show an exponential fall-off of probability with distance from the mean for a sum of many *independent* random variables.

Fact 2.3.1 (Chernoff, 1952)**.** *Let $X_1, \ldots, X_m$ be independent $\{0,1\}$-valued random variables which satisfy $\mathbf{E}[X_i = 1] = p_i$ and $0 < p_i < 1$ for all i. Let $\mu = \sum_{i=1}^{m} p_i$. Then for all $0 < \lambda < 1$ we have*

$$\mathbf{Pr}[\sum_{i=1}^{m} X_i \leq (1-\lambda)\mu] \leq \exp(\frac{-\lambda^2\mu}{2}) \quad \textit{and} \quad \mathbf{Pr}[\sum_{i=1}^{m} X_i \geq (1+\lambda)\mu] \leq \exp(\frac{-\lambda^2\mu}{3}).$$

Fact 2.3.2 (Hoeffding, 1963)**.** *Let $X_1, \ldots, X_m$ be independent random variables taking values in the range $[a,b]$. Let $X = \frac{1}{m}\sum_{i=1}^{m} X_i$. Then for all $\lambda > 0$ we have*

$$\mathbf{Pr}[X \geq \mathbf{E}[X] + \lambda] \leq \exp(\frac{-2\lambda^2 m}{(b-a)^2}) \quad \textit{and} \quad \mathbf{Pr}[X \leq \mathbf{E}[X] - \lambda] \leq \exp(\frac{-2\lambda^2 m}{(b-a)^2}).$$

Observe that in the Chernoff Bound the deviation is measured multiplicatively, whereas in the Hoeffding Bound it is measured additively.

We also use the following elementary fact.

Fact 2.3.3 (Markov's Inequality)**.** *For any random variable $X : \Omega \to \mathbb{R}^+ \cup \{0\}$,*

$$\mathbf{Pr}[X \geq \lambda] \leq \mathbf{E}[X]/\lambda.$$

As a convention, $\log$ denotes logarithm base 2, $\ln$ denotes the natural logarithm. For real-valued functions f and g, we write $f(n) = \tilde{O}(g(n))$ if there is a fixed constant c such that $f(n) = O(g(n)\log^c n)$. Given two binary strings, $\oplus$ denotes bitwise addition modulo 2. For $z \in \mathbb{R}$, sgn is the function such that $\mathrm{sgn}(z) = 1$ if $z \geq 0$, $\mathrm{sgn}(z) = -1$ if $z < 0$. For a complex number $z = a + bi \in \mathbb{C}$, $\overline{z}$ denotes its conjugate $a - bi$ and $|z|$ denotes its absolute value $\sqrt{a^2 + b^2}$.

Chapter 3

Improved Bounds on Quantum Learning Algorithms

This chapter is based on the article [AS05] which appeared in the Journal of Quantum Information Processing.

3.1 Introduction

3.1.1 Motivation and background

A major focus of study in quantum computation is the power of quantum algorithms to extract information from a "black-box" oracle for an unknown Boolean function. Many of the most powerful ideas for both algorithmic results and lower bounds in quantum computing have emerged from this framework, which has been studied for more than a decade.

The most frequently considered problem in this setting is to determine whether or not the black-box oracle (which is typically assumed to belong to some particular *a priori* fixed class C of possible functions) has some specific property, such as being identically 0 [BBBV97, Gro96], being exactly balanced between outputs 0 and 1 [DJ92], or being invariant under an XOR mask [Sim97]. However, as described below researchers have also studied several other problems in which the goal is to obtain more than just one bit of information about the target black-box function:

Quantum exact learning from membership queries: In [SG04] Servedio and Gortler initiated a

systematic study of the quantum black-box query complexity required to *exactly learn* any unknown function c from a class C of Boolean functions. This is a natural quantum analogue of the standard classical model of *exact learning from membership queries* which was introduced in computational learning theory by Angluin [Ang88]. This quantum exact learning model was also studied by Hunziker *et al.* [HMP$^+$03] and by Ambainis *et al.* [AIK$^+$04a], who gave a general upper bound on the quantum query complexity of learning any class C.

PAC learning from quantum examples: In another line of related research, Bshouty and Jackson [BJ99] introduced a natural quantum analogue of Valiant's well-known Probably Approximately Correct (PAC) model of Boolean function learning [Val84] which is widely studied in computational learning theory. [SG04] subsequently gave a $\Omega(d/n)$ lower bound on the number of quantum examples required for any PAC learning algorithm for any class C of Boolean functions over $\{0,1\}^n$ which has Vapnik-Chervonenkis dimension d.

3.1.2 The results of this chapter

In this chapter we study three natural problems of quantum learning:

- exact learning from quantum membership queries as described above;
- learning a *partition* of a class of functions from quantum membership queries (this is an intermediate problem between the quantum exact learning problem and the well-studied problem of obtaining a single bit of information about the target function), and
- quantum PAC learning as described above.

For each of these problems we give new bounds on the number of quantum queries or examples that are required for learning.

For the quantum exact learning model, Hunziker *et al.* [HMP$^+$03] conjectured that for any class C of Boolean functions, the number of quantum black-box queries that are required to exactly learn an unknown function from C is $O(\frac{\log |C|}{\sqrt{\hat{\gamma}^C}})$, where $\hat{\gamma}^C$ (defined in Section 3.3.1) is a combinatorial parameter of the class C. We give a new quantum exact learning algorithm based on a multi-target Grover search on a prescribed subset of the inputs, and show that the query complexity

for this algorithm is $O(\frac{\log |C| \log\log |C|}{\sqrt{\hat{\gamma}^C}})$; this resolves the conjecture of Hunziker *et al.* [HMP$^+$03] up to a $\log\log |C|$ factor. Our new bound is incomparable with the upper bound of Ambainis *et al.* [AIK$^+$04a], but as we show it improves on this bound for a wide range of parameter settings. We also show that for every class C of Boolean functions, the query complexity of our generic algorithm is guaranteed to be at most a (roughly) quadratic factor worse than the query complexity of the *best* quantum algorithm for learning C (which may be tailored for the specific class C).

For our second problem, we study a more general problem which is intermediate between learning the black-box function exactly and computing a single Boolean predicate of the unknown black-box function. This problem is the following: given a partition of a class C into disjoint subsets $P_1, \ldots, P_k$, determine which piece the unknown black-box function $c \in C$ belongs to. Ambainis *et al.* proposed the study of this problem as an interesting direction for future work in [AIK$^+$04a]. Note that the problem of computing a single Boolean predicate of an unknown function $c \in C$ corresponds to having a two-way partition, whereas the problem of exact learning corresponds to a partition of C into $|C|$ disjoint pieces.

We show that for any concept class C and any partition size $2 \leq k \leq |C|$, there is a partition of C into k pieces such that the classical and quantum query complexities are polynomially related. On the other hand, we also show that for a wide range of partition sizes k it is possible for the quantum and classical query complexities of learning a k-way partition to have a superpolynomial separation. These results show that the structure of the partition plays a more important role than the size in determining the relationship between quantum and classical complexity of learning.

Finally, for the quantum PAC learning model, we improve the $\Omega(\frac{d}{n})$ lower bound of [SG04] on the number of quantum examples which are required to PAC learn any concept class of Vapnik-Chervonenkis dimension d over $\{0,1\}^n$. Our new bound of $\Omega(\frac{1}{\epsilon}\log\frac{1}{\delta} + d + \frac{\sqrt{d}}{\epsilon})$ is not far from the known lower bound of Ehrenfeucht *et al.* [EHKV89] of $\Omega(\frac{1}{\epsilon}\log\frac{1}{\delta} + \frac{d}{\epsilon})$ for classical PAC learning. Since the lower bound of [EHKV89] is known to be nearly optimal for classical PAC learning algorithms (an upper bound of $O(\frac{1}{\epsilon}\log\frac{1}{\delta} + \frac{d}{\epsilon}\log\frac{1}{\epsilon})$ was given by [BEHW89]), our new quantum lower bound is not far from being the best possible.

3.1.3 Organization of this chapter

Section 3.3 gives our new quantum algorithm for exactly learning a black-box function. Section 3.4 gives some simple examples and poses a question about the relation between query complexity of quantum and classical exact learning. Section 3.5 gives our results on the partition learning problem, and Section 3.6 gives our new lower bound on the sample complexity of quantum PAC learning.

3.2 Preliminaries

3.2.1 Learning preliminaries

In this chapter, a *concept* c over $\{0,1\}^n$ is a Boolean function $c : \{0,1\}^n \to \{0,1\}$. Equivalently we may view a concept as a subset of $\{0,1\}^n$ defined by $\{x \in \{0,1\}^n : c(x) = 1\}$. A *concept class* $\mathfrak{C} = \cup_{n \geq 1} C_n$ is a set of concepts where C_n consists of those concepts in $\mathfrak{C}$ whose domain is $\{0,1\}^n$. For ease of notation throughout the chapter we will omit the subscript in C_n and simply write C to denote a collection of concepts over $\{0,1\}^n$. It will often be useful to think of C as a $|C| \times 2^n$-binary matrix where rows correspond to concepts $c \in C$, columns correspond to inputs $x \in \{0,1\}^n$, and the (i,j) entry of the matrix is the value of the i-th concept on the j-th input.

We say that a concept class C is *1-sensitive* if it has the property that for each input x, at least half of all concepts $c \in C$ have $c(x) = 0$ (i.e. each column of the matrix C is at most half ones). Given any C it is possible to convert it to an equivalent 1-sensitive concept class by flipping the value obtained from any input x which has $|\{c : c(x) = 1\}| > |\{c : c(x) = 0\}|$. This condition on x can simply be checked by enumerating all concepts c in C – without making any queries. In general, we refer to the process of flipping the matrix entries which reside in a particular subset of columns as performing a *column flip*. This notion of 1-sensitivity and a column flip was first introduced by [AIK$^+$04a].

It is important to note that achieving the effect of a column flip in our algorithms involves creating and using simulated oracles. In other words, a column flip affects not only the matrix corresponding to the set of candidate concepts C but also the result of classical and quantum membership queries. Therefore, after a column flip on the subset of inputs K, a membership query

access to the target oracle at one of the inputs in K should be considered to be inverted before returned to the algorithm. As remarked in [AIK$^+$04a], in both the classical and quantum learning models this can be achieved via some additional circuitry which is not significant for our purposes, since we are only interested in the query complexity.

3.2.2 Classical learning models

The model of *exact learning from membership queries* was introduced by Angluin [Ang88] classically and has since been studied by several authors [BCG$^+$96, Gav94, Heg95, HPRW96]. In this framework, a learning algorithm for C is given query access to a black-box oracle $\mathsf{MQ}(c)$ for the unknown target concept $c \in C$, i.e. when the learner provides $x \in \{0,1\}^n$ to $\mathsf{MQ}(c)$ she receives back the value $c(x)$. A learning algorithm is said to be an *exact learning algorithm for concept class* C if the following holds: for any $c \in C$, with probability at least $2/3$ the learning algorithm outputs a Boolean circuit h which is logically equivalent to c. (We remind the reader that a learning algorithm for C "knows" the class C but does not know the identity of the target concept $c \in C$.) The *query complexity* of a learning algorithm is the number of queries that it makes to $\mathsf{MQ}(c)$ before outputting h. We will be chiefly concerned in this chapter with a quantum version of the exact learning model, which we describe in Section 3.3.

In the classical PAC (Probably Approximately Correct) learning model, which was introduced by Valiant [Val84] and subsequently studied by many authors, the learning algorithm has access to a *random example oracle* $\mathsf{EX}(c, \mathcal{D})$ where $c \in C$ is the unknown target concept and $\mathcal{D}$ is an unknown probability distribution over $\{0,1\}^n$. At each invocation the oracle $\mathsf{EX}(c, \mathcal{D})$ (which takes no inputs) outputs a labeled example $\langle x, c(x) \rangle$ where $x \in \{0,1\}^n$ is drawn from the distribution $\mathcal{D}$. An algorithm $\mathcal{A}$ is a *PAC learning algorithm for concept class* C if the following condition holds: given any $\epsilon, \delta > 0$, for all $c \in C$ and all distributions $\mathcal{D}$ over $\{0,1\}^n$, if $\mathcal{A}$ is given ϵ, δ and is given access to $\mathsf{EX}(c, \mathcal{D})$ then with probability at least $1 - \delta$ the output of $\mathcal{A}$ is a Boolean circuit $h : \{0,1\}^n \to \{0,1\}$ (called a hypothesis) which satisfies $\mathbf{Pr}_{x \in \mathcal{D}}[h(x) \neq c(x)] \leq \epsilon$. The *(classical) sample complexity* of $\mathcal{A}$ is the maximum number of calls to $\mathsf{EX}(c, \mathcal{D})$ which it makes for any $c \in C$ and any distribution $\mathcal{D}$. In Section 3.6 we will study a quantum version of the PAC learning model.

3.3 Exact learning with quantum membership queries

Given any concept $c : \{0,1\}^n \to \{0,1\}$, the *quantum membership oracle associated to* c $\mathsf{QMQ}(c)$ is the quantum oracle whose query acts on the computational basis states as $|x, b\rangle \mapsto |x, b \oplus c(x)\rangle$ where $x \in \{0,1\}^n$ and $b \in \{0,1\}$. Recall that we mentioned this oracle earlier in Example 2.1.11. A *quantum exact learning algorithm* for a concept class C is a sequence of unitary transformations $U_0, \mathsf{QMQ}(c), U_1, \mathsf{QMQ}(c), \ldots, \mathsf{QMQ}(c), U_T$ where each U_i is a fixed unitary transformation without any dependence on c. The algorithm must satisfy the following property: for any target concept $c \in C$ which is used to instantiate the QMQ queries, a measurement performed on the final state will with probability at least $2/3$ yield a representation of a (classical) Boolean circuit $h : \{0,1\}^n \to \{0,1\}$ such that $h(x) = c(x)$ for all $x \in \{0,1\}^n$. The *quantum query complexity* of the algorithm is T, the number of queries to $\mathsf{QMQ}(c)$.

Note that a quantum membership oracle $\mathsf{QMQ}(c)$ is identical to the notion of "a quantum black-box oracle for c" which has been widely studied in e.g. [BBC$^+$01, FGGS98, Gro96] and many other works. Most of this work, however, focuses on the quantum query complexity of computing a single bit of information about the unknown oracle, e.g. the OR of all its output values [Gro96] or the parity of all its output values [FGGS98]. The quantum exact learning problem which we consider in this section was proposed in [SG04] and later studied in [AIK$^+$04a] (where it is called the "oracle identification problem") and in [HMP$^+$03].

Throughout the chapter we write $R(C)$ to denote the minimum query complexity of any classical (randomized) exact learning algorithm for concept class C. We write $Q(C)$ to denote the minimum query complexity of any quantum exact learning algorithm for C. We write N to denote 2^n, the number of elements in the domain of each $c \in C$.

In Section 3.3.1 we briefly recap known bounds on the query complexity of quantum and classical exact learning algorithms. In Section 3.3.2 we give our new quantum learning algorithm, prove correctness, and analyze its query complexity.

3.3.1 Known bounds on query complexity for exact learning

We begin by defining a combinatorial parameter $\hat{\gamma}^C$ of a concept class C which plays an important role in bounds on query complexity of exact learning algorithms.

Definition 3.3.1. Let C be a concept class over $\{0,1\}^n$. We define

$$\gamma_a^{C'} = \min_{b\in\{0,1\}} |\{c \in C' : c(a) = b\}|/|C'|, \qquad \text{where } a \in \{0,1\}^n, C' \subseteq C$$

$$\gamma^{C'} = \max_{a\in\{0,1\}^n} \gamma_a^{C'}, \qquad \text{where } C' \subseteq C$$

$$\hat{\gamma}^C = \min_{C'\subseteq C, |C'|\geq 2} \gamma^{C'}.$$

If $C' \subseteq C$ is the set of possible remaining target concepts, then $\gamma^{C'}$ is the maximum fraction of C' which a (classical) learning algorithm can be sure of eliminating with a single query. Thus, intuitively, the smaller $\hat{\gamma}^C$ is the more membership queries should be required to learn C.

The following lower and upper bounds on the query complexity of classical exact learning were established in [BCG$^+$96]:

Theorem 3.3.2. *For any concept class C we have $R(C) = \Omega(\frac{1}{\hat{\gamma}^C})$ and $R(C) = \Omega(\log |C|)$.*

Theorem 3.3.3. *There is a classical exact learning algorithm which learns any concept class C using $O(\frac{\log |C|}{\hat{\gamma}^C})$ many queries, so consequently $R(C) = O(\frac{\log |C|}{\hat{\gamma}^C})$.*

A quantum analogue of this classical lower bound was obtained in [SG04]:

Theorem 3.3.4. *For any concept class C over $\{0,1\}^n$ we have $Q(C) = \Omega(\frac{1}{\sqrt{\hat{\gamma}^C}})$ and $Q(C) = \Omega(\frac{\log |C|}{n})$.*

Given these results it is natural to seek a quantum analogue of the classical $O(\frac{\log |C|}{\hat{\gamma}^C})$ upper bound. Hunziker *et al.* [HMP$^+$03] made the following conjecture:

Conjecture 3.3.5. *There is a quantum exact learning algorithm which learns any concept class C using $O(\frac{\log |C|}{\sqrt{\hat{\gamma}^C}})$ quantum membership queries.*

In Section 3.3.2 we prove this conjecture up to a $\log \log |C|$ factor.

Hunziker *et al.* [HMP$^+$03] also conjectured that there is a quantum exact learning algorithm which learns any concept class C using $O(\sqrt{|C|})$ queries. This was established by Ambainis *et al.* [AIK$^+$04a], who also proved the following result:

Theorem 3.3.6. *There is a quantum exact learning algorithm which learns any concept class C with $|C| > N$ using $O(\sqrt{N \log |C| \log N} \log \log |C|)$ many queries.*

3.3.2 A new quantum exact learning algorithm

We start with a simple yet useful observation:

Lemma 3.3.7. *For any concept class C, there exists an $x \in \{0,1\}^n$ for which at least a $\hat{\gamma}^C$ fraction of concepts $c \in C$ satisfy $c(x) = 1$. More generally for every subset $C' \subseteq C$ with $|C'| \geq 2$, there exists an input x at which the fraction of concepts in C' yielding 1 is at least $\gamma^{C'}$ (which is at least as large as $\hat{\gamma}^C$).*

Proof. It is sufficient to prove the result in the latter general form. Consider any subset $C' \subseteq C$ with $|C'| \geq 2$. By Definition 3.3.1 we know:

- $\gamma^{C'} \geq \hat{\gamma}^C$.
- At any input $z \in \{0,1\}^n$, the fraction of concepts in C' yielding 1 has to be at least $\gamma_z^{C'}$.

Now consider the input a which satisfies $\gamma_a^{C'} = \gamma^{C'}$: the fraction of concepts in C' yielding 1 at input a should therefore be at least $\gamma^{C'}$. Thus taking $x = a$ gives the intended result. □

The quantity $\hat{\gamma}^C$ can be bounded as follows:

Lemma 3.3.8. *For any concept class C with $|C| \geq 2$, $\frac{1}{N+1} \leq \gamma^C \leq \frac{1}{2}$. This also implies $\frac{1}{N+1} \leq \hat{\gamma}^C \leq \frac{1}{2}$ by Definition 3.3.1.*

Proof. $\gamma^C \leq \frac{1}{2}$ is clear from the Definition 3.3.1. To prove the other direction we may assume that $\gamma^C < \frac{1}{N}$, since otherwise the result is obviously true. Therefore $|C| > N$ must hold. Observe that at each input x one of the following must hold:

- The fraction of concepts in C yielding 0 at x is at least $1 - \gamma^C$ and thus the fraction of concepts in C yielding 1 at x is at most γ^C.
- The fraction of concepts in C yielding 1 at x is at least $1 - \gamma^C$ and thus the fraction of concepts in C yielding 0 at x is at most γ^C.

Hence C can contain at most $\gamma^C|C|N$ concepts which are not identically equal to the concept c_{maj}

Algorithm 1 Constructing a set of inputs which satisfies the semi-rich row condition.

1: $S \leftarrow \emptyset, \mathcal{I} \leftarrow \emptyset$.
2: **repeat**
3: Perform a column flip on $C \setminus S$ to make $C \setminus S$ be 1-sensitive.
4: $a_{max} \leftarrow$ the input in $\{0,1\}^n \setminus \mathcal{I}$ at which the highest fraction of concepts in $C \setminus S$ yield 1.
5: $\mathcal{I} \leftarrow \mathcal{I} \cup \{a_{\max}\}$.
6: $S \leftarrow S \cup$ the set of concepts in the original matrix C that yield 1 at input a_{max}.
7: **until** $|S| \geq |C|/2$.
8: Output $\leftarrow \mathcal{I}$.

defined as follows:

$$c_{\mathsf{maj}}(x) = \begin{cases} 0, & \text{if at least half of the concepts in } C \text{ yield } 0 \text{ at } x; \\ 1, & \text{otherwise.} \end{cases}$$

Therefore C must be comprised of at most these $\gamma^C|C|N$ concepts and possibly c_{maj}. Thus we obtain:

$$\gamma^C|C|N + 1 \geq |C| \Longrightarrow \gamma^C \geq \frac{|C|-1}{|C|N} \geq \frac{N}{(N+1)N} = \frac{1}{N+1}$$

□

Definition 3.3.9. A subset of inputs $\mathcal{I} \subseteq \{0,1\}^n$ is said to satisfy *the semi-rich row condition for* C if at least half the concepts in C have the property that they yield 1 for at least a $\hat{\gamma}^C$ fraction of the inputs x in $\mathcal{I}$.

The phrase "semi-rich row condition" is used because viewing C as a matrix, at least half the rows of C are "rich" in 1s (have at least a $\hat{\gamma}^C$ fraction of 1s) within the columns indexed by inputs in $\mathcal{I}$. A simple greedy approach can be used to construct a set of inputs which satisfies the semi-rich row condition for C:

Lemma 3.3.10. *Let C be any concept class with $|C| \geq 2$. Then Algorithm 1 outputs a set of inputs $\mathcal{I}$ with $|\mathcal{I}| \leq \frac{1}{\hat{\gamma}^C}$ which satisfies the semi-rich row condition for C.*

Proof. Let $\tau_j|C|$ be the number of concepts in $C \setminus S$ after the j-th execution of the repeat loop in Algorithm 1, so $\tau_0 = 1$. Using the first result of Lemma 3.3.7, we obtain $\tau_1 \leq 1 - \hat{\gamma}^C$. Now invoking Lemma 3.3.7 once again but this time in its general form, we obtain $\tau_2 \leq (1 - \hat{\gamma}^C)^2$;

note that after the second iteration of the loop, each concept in S will yield 1 on at least half of the elements in $\mathcal{I}$. Let j' equal $\lfloor \frac{1}{\hat{\gamma}^C} \rfloor$. If Algorithm 1 proceeds for j' iterations through the loop, then it must be the case that $\tau_{j'} \leq (1 - \hat{\gamma}^C)^{j'}$ and that each concept in S yields 1 on at least a $\hat{\gamma}^C$ fraction of the elements in $\mathcal{I}$. It is easy to verify that $(1-x)^{\lfloor 1/x \rfloor} < 1/2$ for $0 < x < \frac{1}{2}$. We thus have that $\tau_{j'}|C| < |C|/2$, and consequently $|S| > |C|/2$, so $\mathcal{I}$ satisfies the semi-rich row condition for C and the algorithm will terminate before starting the $(\lfloor \frac{1}{\hat{\gamma}^C} \rfloor + 1)$-th iteration.

In the case $\hat{\gamma}^C \leq \frac{1}{N}$, then the set of all N inputs will satisfy the semi-rich row condition for C: since any concept which does not yield 0 for all inputs actually yields 1 for at least a $\hat{\gamma}^C$ fraction of all inputs. Therefore in this case the algorithm will terminate successfully with an output $|\mathcal{I}| \leq N \leq \frac{1}{\hat{\gamma}^C}$. Otherwise, since $\hat{\gamma}^C > \frac{1}{N}$, we have that $j' < N$. This means the algorithm never runs out of inputs to add (i.e. $\{0,1\}^N \setminus \mathcal{I}$ is nonempty at every iteration). □

Our quantum learning algorithm is given in Algorithm 2. Throughout the algorithm the set $S \subseteq C$ should be viewed as the set of possible target concepts that have not yet been eliminated; the algorithm halts when $|S| = 1$. The high-level idea of the algorithm is that in every repetition of the outer loop the size of S is multiplied by a factor which is at most $\frac{1}{2}$, so at most $\log |C|$ repetitions of the outer loop are performed.

Theorem 3.3.11. *Let C be any concept class with $|C| \geq 2$. Algorithm 2 is a quantum exact learning algorithm for C which performs $O(\frac{\log |C| \log\log |C|}{\sqrt{\hat{\gamma}^C}})$ quantum membership queries.*

Proof. Consider a particular iteration of the outer Repeat-Until loop. The set S is 1-sensitive by virtue of the first step (the column flip). By Lemma 3.3.10, in the second step of this iteration, $\mathcal{I}$ becomes a set of at most $\frac{1}{\hat{\gamma}^S}$ many inputs which satisfies the semi-rich row condition for S. Consequently, each execution of the Grover search within the inner Repeat-Until loop uses $O(\sqrt{\frac{1}{\hat{\gamma}^S}})$ (which is also $O(\sqrt{\frac{1}{\hat{\gamma}^C}})$) many queries. Since the inner loop repeats at most $\log(3 \log |C|)$ many times, if we can show that each iteration of the outer loop does indeed with high probability

- cause the size of S to be multiplied by a factor which is at most $\frac{1}{2}$, and
- maintain the property that the target concept is contained in S,

then the theorem will be proved.

Algorithm 2 A quantum exact learning algorithm.

$S \leftarrow C$.
repeat
  Perform a column flip on S to make S 1-sensitive. Let $\mathcal{K}$ be the set of inputs at which the output is flipped during this procedure.
  $\mathcal{I} \leftarrow$ The output of Algorithm 1 invoked on the set of concepts S.
  Counter $\leftarrow 0$, Success$\leftarrow$FALSE.
  repeat
    Perform the multi-target subset Grover search on $\mathcal{I}$ using $\frac{9}{2}\lceil\sqrt{|\mathcal{I}|}\rceil$ queries [BBHT98].
    $a \leftarrow$ Result of the Grover search.
    if a classical query of the oracle at a yields 1 **then**
      $S \leftarrow \{$the concepts in S that yield 1 at $a\}$, Success$\leftarrow$TRUE.
    end if
    Counter$\leftarrow$Counter$+1$.
  until Success OR (Counter$>= \log(3\log|C|)$)
  if NOT Success **then**
    $S \leftarrow$ the set of concepts that yield 0 for all of the inputs in $\mathcal{I}$.
  end if
  Flip the outputs of concepts in S for all elements in $\mathcal{K}$ to reverse the earlier column flip (thus restoring all concepts in S to their original behavior on all inputs).
until $|S| = 1$.
Output $\leftarrow$ A representation of a circuit which computes the sole concept c in S.

As shown in [BBHT98], the multi-target Grover search algorithm over a space of $|\mathcal{I}|$ many inputs using $\frac{9}{2}\lceil\sqrt{|\mathcal{I}|}\rceil$ many queries has the property that if there is any input $a \in \mathcal{I}$ on which the target function yields 1, then the search will output such an a with probability at least $\frac{1}{2}$. Since the inner loop repeats $\log(3\log|C|)$ many times, we thus have that if there is any input $a \in \mathcal{I}$ on which the target concept yields 1, then with probability at least $1 - \frac{1}{3\log|C|}$ one of the $\log(3\log|C|)$ many iterations of the inner loop will yield such an a and the "Success" variable will be set to TRUE. Since the set S is 1-sensitive, when we eliminate from S all the concepts which yield 0 at a we will multiply the size of S by at most $\frac{1}{2}$ as desired in this case (and clearly we will not eliminate the target concept from S). On the other hand, if the set $\mathcal{I}$ contains no input a on which the target concept yields 1, then after $\log(3\log|C|)$ iterations of the inner loop we will exit with Success set to FALSE, and the concepts that yield 1 on any input in $\mathcal{I}$ will be removed from S. This will clearly not cause the target to be removed from S. Moreover because $|\mathcal{I}| \leq \frac{1}{\hat{\gamma}^S}$, any concept for which even a single input in $\mathcal{I}$ yields 1 has the property that at least a $\hat{\gamma}^S$ fraction of inputs in $\mathcal{I}$ yield 1. Since $\mathcal{I}$ satisfies the semi-rich row condition for S, this means that we have eliminated at least

half the concepts in S. Thus, the algorithm will succeed with probability at least $(1 - \frac{1}{3 \log |C|})^{\log C}$ which is larger than $2/3$, and the theorem is proved. □

Recall that $Q(C)$ denotes the optimal query complexity over all quantum exact learning algorithms for concept class C. We can show that the query complexity of Algorithm 2 is never much worse than the optimal query complexity $Q(C)$:

Corollary 3.3.12. *For any concept class C, Algorithm 2 uses $O(nQ(C)^2 \log \log |C|)$ queries.*

Proof. This follows directly from Theorem 3.3.11 and the bound $Q(C) = \Omega(\frac{\log |C|}{n} + \frac{1}{\sqrt{\hat{\gamma}^C}})$ of Theorem 3.3.4. □

Since $|C| \leq 2^{2^n}$, the bound $O(nQ(C)^2 \log \log |C|)$ is always $O(n^2 Q(C)^2)$, and thus the query complexity of Algorithm 2 is always polynomially related to the query complexity of the optimal algorithm for any concept class C.

Discussion

Algorithm 2 can be viewed as a variant of the algorithm of [AIK$^+$04a] which learns any concept class C from $O(\sqrt{N \log |C| \log N} \log \log |C|)$ quantum membership queries. This algorithm repeatedly performs Grover search over the set of all inputs, with the goal each time of eliminating at least half of the remaining target concepts. Instead, our approach is to perform each Grover search only over sets which satisfy the semi-rich row condition for the remaining set of possible target concepts. By doing this, we are able to obtain an upper bound on query complexity in terms of $\hat{\gamma}^C$ for every such iteration.

We observe that our new bound of $O(\frac{\log |C| \log \log |C|}{\sqrt{\hat{\gamma}^C}})$ is stronger than the previously obtained upper bound of $O(\sqrt{N \log |C| \log N} \log \log |C|)$ from [AIK$^+$04a] as long as $\frac{\log |C|}{\hat{\gamma}^C} = o(N \log N)$. Thus, for any concept class C for which the $O(\frac{\log |C|}{\hat{\gamma}^C})$ upper bound of Theorem 3.3.3 on classical membership query algorithms is nontrivial (i.e. is less than N), our results give an improvement.

We note that independently Iwama *et al.* [AIK$^+$04b] have recently given a new algorithm for quantum exact learning that uses ideas similar to the construction of Algorithm 1; however the analysis is different and their results are incomparable to ours (their bounds depend only on the number

of concepts in C and not on the combinatorial parameter $\hat{\gamma}^C$). The main focus of [AIK$^+$04b] is on obtaining robust learning algorithms that can learn successfully using noisy oracle queries.

3.4 Relations between query complexity of quantum and classical exact learning

As noted in [SG04], combining Theorems 3.3.3 and 3.3.4 yields the following:

Corollary 3.4.1. *For any concept class C, we have $Q(C) \leq R(C) = O(nQ(C)^3)$.*

Can tighter bounds relating $R(C)$ and $Q(C)$ be given which hold for all concept classes C? While we have not been able to answer this question, here we make some simple observations and pose a question which we hope will stimulate further work.

We first observe that the factor n is required in the bound $R(C) = O(nQ(C)^3)$:

Lemma 3.4.2. *For any positive integer d there exists a concept class C over $\{0,1\}^n$ with $R(C) = \omega(1)$ which has $R(C) = \Omega(Q(C)^d)$.*

Proof. We assume $d > 1$. Recall that in the Bernstein-Vazirani problem, the target concept is an unknown parity function over some subset of the n Boolean variables $x_1, \ldots, x_n$; Bernstein and Vazirani showed [BV97] that for this concept class we have $R(C) = n$ whereas $Q(C) = 1$. We thus consider a concept class in which each concept c contains $n^{1/d}$ copies of the Bernstein-Vazirani problem (each instance of the problem is over $n^{(d-1)/d}$ variables) as follows: we view n-bit strings a, x as

$$a = (a_{1,1}, a_{1,2}, \ldots, a_{1,n^{(d-1)/d}}, a_{2,1}, a_{2,2}, \ldots, a_{2,n^{(d-1)/d}}, \ldots a_{n^{1/d},1}, \ldots, a_{n^{1/d},n^{(d-1)/d}})$$

$$x = (x_{1,1}, x_{1,2}, \ldots, x_{1,n^{(d-1)/d}}, x_{2,1}, x_{2,2}, \ldots, x_{2,n^{(d-1)/d}}, \ldots x_{n^{(1/d)},1}, \ldots, x_{n^{1/d},n^{(d-1)/d}})$$

The class C consists of the set of all 2^n concepts:

$$f_a(x) = \bigvee_{i=1}^{n^{1/d}} ((a_{i,1}, a_{i,2}, \ldots, a_{i,n^{(d-1)/d}}) \cdot (x_{i,1}, x_{i,2}, \ldots, x_{i,n^{(d-1)/d}}) \bmod 2)$$

i.e. $f_a(x)$ equals 1 if any of the $n^{1/d}$ parities corresponding to the substrings $a_{i,\cdot}$ take value 1 on the corresponding substring of x.

It is easy to see that n queries suffice for a classical algorithm, and by Theorem 3.3.2 we have $R(C) = \Omega(\log |C|)$, so $R(C) = \Theta(n)$. On the other hand, it is also easy to see that $Q = O(n^{\frac{1}{d}})$ since a quantum algorithm can learn by making $n^{1/d}$ successive runs of the Bernstein-Vazirani algorithm.

Finally, if $d = 1$ then as shown in [vD98] the set C of all 2^{2^n} concepts over $\{0,1\}^n$ has $Q(C) = \Theta(2^n)$ and $R(C) = \Theta(2^n)$. □

The bound $R(C) = O(nQ(C)^3)$ implies that the gap of Lemma 3.4.2 can only be achieved for concept classes C which have $R(C)$ small. However, it is easy to exhibit concept classes which have a factor n difference between $R(C)$ and $Q(C)$ for a wide range of values of $R(C)$:

Lemma 3.4.3. *For any k such that $n-k = \Theta(n)$, there is a concept class C with $R(C) = \Theta(n2^k)$ and $Q(C) = \Theta(2^k)$.*

Proof. The concept class C is defined as follows. A concept $c \in C$ corresponds to $(a^0, \ldots, a^{2^k-1})$, where each a^i is a $(n-k)$-bit string. The concept c maps input $x \in \{0,1\}^n$ to $(a^i \cdot y) \bmod 2$, where i is the number between 0 and $2^k - 1$ whose binary representation is the first k bits of x and y is the $(n-k)$-bit suffix of x. Since each concept in C is defined uniquely by 2^k many $(n-k)$-bit strings $a^0, \ldots, a^{2^k-1}$, there are $2^{2^k(n-k)}$ concepts in C.

Theorem 3.3.2 yields $R(C) = \Omega(2^k(n-k))$. It is easy to see that in fact $R(C) = \Theta(2^k(n-k))$: For each of the 2^k parities which one must learn (corresponding to the 2^k possible prefixes of an input), one can learn the $(n-k)$-bit parity with $n-k$ classical queries.

It is also easy to see that by running the Bernstein-Vazirani algorithm 2^k times (once for each different k-bit prefix), a quantum algorithm can learn an unknown concept from C exactly using 2^k queries, and thus $Q(C) = O(2^k)$. The $Q(C) = \Omega(\frac{\log |C|}{n})$ lower bound of Theorem 3.3.2 gives us $Q(C) = \Omega(\frac{n-k}{n} \cdot 2^k) = \Omega(2^k)$, and the lemma is proved. □

Based on these observations, we pose the following question:

Question 3.4.4. Does every concept class C satisfy $R(C) = O(nQ(C) + Q(C)^2)$?

Note that the example in Lemma 3.4.3 and the concept class of Grover search [Gro96]: $C = \{f_i, 0 \leq i < N : f_i(x) = \delta_{i,x}\}$ saturate this upper bound.

3.5 On learning a partition of a concept class

Definition 3.5.1. Let C be a concept class over $\{0,1\}^n$. A *partition* $\mathcal{P}$ *of* C is a collection of nonempty disjoint subsets $P_1, \ldots, P_k$ whose union yields C.

In this section we study a different problem, mentioned by Ambainis *et al.* [AIK$^+$04a], that is more relaxed than exact learning: given a partition $\mathcal{P}$ of C and a black-box (quantum or classical) oracle for an unknown target concept c in C, what is the query complexity of identifying the set P_i in $\mathcal{P}$ which contains c? It is easy to see that both the exact learning problem (in which $|\mathcal{P}| = |C|$) and the problem of computing some binary property of c (for which $|\mathcal{P}| = 2$) are special cases of this more general problem. One can view these problems in the following way: for the exact learning problem the algorithm must obtain all $\log |C|$ bits of information about the target concept, whereas for the problem of computing a property of c the algorithm must obtain a single bit of information. In a general instance of the partition problem, the algorithm must obtain $\log |\mathcal{P}|$ bits of information about the target concept.

Given a concept class C and a partition $\mathcal{P}$ of C, we will write $R_{\mathcal{P}}(C)$ to denote the optimal query complexity of any classical (randomized) algorithm for the partition problem which outputs the correct answer with probability at least $2/3$ for any target concept c. We similarly write $Q_{\mathcal{P}}(C)$ to denote the optimal complexity of any quantum query algorithm with the same success criterion.

As noted earlier, for the case $|\mathcal{P}| = |C|$ we know from Corollary 3.4.1 that the quantities $R_{\mathcal{P}}(C)$ and $Q_{\mathcal{P}}(C)$ are polynomially related for any concept class C, since $R_{\mathcal{P}}(C) = O(nQ_{\mathcal{P}}(C)^3)$. On the other extreme, if $|\mathcal{P}| = 2$ then concept classes are known for which $R_{\mathcal{P}}(C)$ and $Q_{\mathcal{P}}(C)$ are polynomially related (see e.g. [BBC$^+$01]), and concept classes are also known for which there is an exponential gap [Sim97]. It is thus natural to investigate the relationship between the size of $|\mathcal{P}|$ and the existence of a polynomial relationship between $R_{\mathcal{P}}(C)$ and $Q_{\mathcal{P}}(C)$.

In this section, we show that the number of sets in $|\mathcal{P}|$ alone (viewed as a function of $|C|$) often does not provide sufficient information to determine whether $R_{\mathcal{P}}(C)$ and $Q_{\mathcal{P}}(C)$ are polynomially

related. More precisely, in Section 3.5.1 we show that for *any* concept class C over $\{0,1\}^n$ and any value $2 \leq k \leq |C|$, there is a partition $\mathcal{P}$ of C with $|\mathcal{P}| = k$ for which we have $R_{\mathcal{P}}(C) = O(nQ_{\mathcal{P}}(C)^3)$. On the other hand, in Section 3.5.2 we show that for a wide range of values of $|\mathcal{P}|$ (again as a function of $|C|$), there are concept classes which have a superpolynomial separation between $R_{\mathcal{P}}(C)$ and $Q_{\mathcal{P}}(C)$. Thus, our results concretely illustrate that the structure of the partition (rather than the number of the sets in the partition) plays an important role in determining whether the quantum and classical query complexities are polynomially related.

3.5.1 Partition problems for which quantum and classical complexity are polynomially related

The following simple lemma extends the cardinality-based lower bounds of Theorem 3.3.2 and Theorem 3.3.4 for exact learning to the problem of learning a partition:

Lemma 3.5.2. *For any partition $\mathcal{P}$ of any concept class C over $\{0,1\}^n$, we have $R_{\mathcal{P}}(C) = \Omega(\log |\mathcal{P}|)$ and $Q_{\mathcal{P}}(C) = \Omega(\frac{\log |\mathcal{P}|}{n})$.*

Proof. Let $C' \subseteq C$ be a concept class formed by taking any single element from each subset in the partition $\mathcal{P}$. Learning $\mathcal{P}$ requires at least as many queries as exact learning the concept class C', and so the result follows from Theorem 3.3.2 and Theorem 3.3.4. □

To obtain a partition analogue of the other lower bounds of Theorems 3.3.2 and 3.3.4, we define the following combinatorial parameter which is an analogue of $\hat{\gamma}^C$:

Definition 3.5.3. Let $\mathcal{S}$ be the set of all subsets $C' \subseteq C$, $|C'| \geq 2$ which have the property that any subset $C'' \subseteq C'$ with $|C''| \geq \frac{3}{4}|C'|$ must intersect at least two subsets in $\mathcal{P}$. We define $\hat{\gamma}^C_{\mathcal{P}}$ to be $\hat{\gamma}^C_{\mathcal{P}} := \min_{C' \subseteq \mathcal{S}} \gamma^{C'}$.

Thus each subset C' in $\mathcal{S}$ has the property that the partition induced on C' by $\mathcal{P}$ contains no subset of size as large as $\frac{3}{4}|C'|$.

The next lemma shows that for each $C' \in \mathcal{S}$, the lower bounds for exact learning $R(C') = \Omega(\frac{1}{\gamma^{C'}})$ and $Q(C') = \Omega(\frac{1}{\sqrt{\gamma^{C'}}})$ which are implied by Theorems 3.3.2 and 3.3.4 extend to the

problem of learning a partition to yield $R_{\mathcal{P}}(C') = \Omega(\frac{1}{\gamma^{C'}})$ and $Q_{\mathcal{P}}(C') = \Omega(\frac{1}{\sqrt{\gamma^{C'}}})$. By considering the $C' \in \mathcal{S}$ which minimizes $\gamma^{C'}$, we obtain the strongest lower bound (this is the motivation behind Definition 3.5.3).

Lemma 3.5.4. *For any partition $\mathcal{P} = P_1, \ldots, P_k$ of the concept class C, we have $R_{\mathcal{P}}(C) = \Omega(\frac{1}{\hat{\gamma}_{\mathcal{P}}^C})$ and $Q_{\mathcal{P}}(C) = \Omega(\frac{1}{\sqrt{\hat{\gamma}_{\mathcal{P}}^C}})$.*

Proof. Let $C' \in \mathcal{S}$ be such that $\hat{\gamma}_{\mathcal{P}}^C = \gamma^{C'}$. We consider the problem of learning the partition induced by $\mathcal{P}$ over C', and shall prove the lower bound for this easier problem. We may assume without loss of generality that C' is 1-sensitive.

We first consider the classical case. We claim that there is a partition $\{S_1, S_2\}$ of C' with the property that each subset $(P_j \cap C')$ is contained entirely in exactly one of S_1, S_2 (i.e. S_1, S_2 is a "coarsening" of the partition induced by $\mathcal{P}$ over C') which satisfies $\min_{i=1,2} |S_i| > \frac{1}{4}|C'|$. To see this, we may start with $S_2 = \emptyset$, $S_1 = \cup_{i=1}^k (P_i \cap C') = C'$ and consider a process of "growing" S_2 by successively removing the smallest piece $P_j \cap C'$ from S_1 and adding it to S_2. W.l.o.g. we may suppose that $|P_j \cap C'| \leq |P_{j+1} \cap C'|$ for all j, so the pieces $P_j \cap C'$ are added to S_2 in order of increasing $j = 1, 2, \ldots$. Let t be the index such that adding $P_t \cap C'$ to S_2 causes $|S_2|$ to exceed $\frac{1}{4}|C'|$ for the first time. By Definition 3.5.3 it cannot be the case that adding $P_t \cap C'$ causes S_2 to become all of C' (since this would mean that $P_t \cap C'$ is a subset of size at least $\frac{3}{4}|C'|$ which intersects only P_t); thus it must be the case that after this t-th step S_1 is still nonempty. However, it also cannot be the case that after this t-th step we have $|S_1| < \frac{3}{8}|C'|$; for if this were the case, then after the t-th step we would have $|P_t \cap C'| > \frac{3}{8}|C'| > |S_1| = \cup_{j=t+1}^k (P_j \cap C')$ and this would violate the assumption that sets are added to S_2 in order of increasing size.

Since C' is 1-sensitive, the "worst case" for a learning algorithm is that each classical query to the target concept (some $c \in C'$) yields 0. By definition of $\gamma^{C'}$, each such query eliminates at most $\gamma^{C'} \cdot |C'|$ many possible target concepts from C'. Consequently, after $\lfloor \frac{1/4}{\gamma^{C'}} \rfloor - 1$ classical queries, the set of possible target concepts in C' is of size at least $\frac{3}{4}|C'|$, and so it must intersect both S_1 and S_2. It is thus impossible to determine with probability greater than $1/2$ whether c belongs to S_1 or S_2, and thus which piece P_i of $\mathcal{P}$ contains c. This gives the classical lower bound.

Our analysis for the quantum case requires some basic definitions and facts about quantum computing:

Definition 3.5.5. If $|\phi\rangle = \sum_z \alpha_z |z\rangle$ and $|\psi\rangle = \sum_z \beta_z |z\rangle$ are two superpositions of basis states, then the *Euclidean distance* between $|\phi\rangle$ and $|\psi\rangle$ is $\||\phi\rangle - |\psi\rangle\| = (\sum_z |\alpha_z - \beta_z|^2)^{1/2}$. The *total variation distance* between two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ is defined to be $\sum_x |\mathcal{D}_1(x) - \mathcal{D}_2(x)|$.

Fact 3.5.6. (See [BV97]) *Let $|\phi\rangle$ and $|\psi\rangle$ be two unit length superpositions which represent possible states of a quantum register. If the Euclidean distance $\||\phi\rangle - |\psi\rangle\|$ is at most ϵ, then performing the same observation on $|\phi\rangle$ and $|\psi\rangle$ induces distributions $\mathcal{D}_\phi$ and $\mathcal{D}_\psi$ which have total variation distance at most 4ϵ.*

For the quantum lower bound, suppose we have a quantum learning algorithm which makes at most $T = \lfloor \frac{1/4}{32\sqrt{\gamma^{C'}}} \rfloor - 1$ quantum membership queries. We will use the following result which combines Theorem 6 and Lemma 7 from [SG04] (those results are in turn based on Theorem 6.6 of [BBBV97]):

Lemma 3.5.7 (See [SG04])**.** *Consider any quantum algorithm $\mathcal{N}$ which makes T quantum membership queries. Let $|\phi_T^c\rangle$ denote the state of the quantum register after all T membership queries are performed in the algorithm, if the target concept is c. Then for any 1-sensitive set $C' \subseteq C$ of concepts with $|C'| \geq 2$ and any $\epsilon > 0$, there is a set $S \subseteq C'$ of cardinality at most $T^2|C'|\gamma^{C'}/\epsilon^2$ such that for all $c \in C' \setminus S$, we have $\||\phi_T^{\mathbf{0}}\rangle - |\phi_T^c\rangle\| \leq \epsilon$ (where $\mathbf{0}$ denotes the identically 0 concept).*

If we take $\epsilon = \frac{1}{32}$, then Lemma 3.5.7 implies that there exists a set $S \subseteq C'$ of cardinality less than $\frac{1}{4} \cdot |C'|$ such that for all $c \in C' \setminus S$ one has $\||\phi_T^{\mathbf{0}}\rangle - |\phi_T^c\rangle\| \leq \frac{1}{32}$. Consequently by Definition 3.5.3 there must exist two concepts $c_1, c_2 \in C' \setminus S$ with $\||\phi_T^{c_1}\rangle - |\phi_T^{c_2}\rangle\| \leq \frac{1}{16}$ which belong to different subsets P_i and P_j of $\mathcal{P}$. By Fact 3.5.6, the probability that our quantum learning algorithm outputs "i" can differ by at most $\frac{1}{4}$ when the target concept is c_1 versus c_2; but this contradicts the assumption that the algorithm is correct with probability at least $2/3$ on all target concepts. This proves the quantum lower bound. □

Before proving the main result of this section, we establish the following result which gives a sufficient condition for the quantum and classical complexities $Q_{\mathcal{P}}(C)$ and $R_{\mathcal{P}}(C)$ of learning a partition to be polynomially related. This result is a generalization of Corollary 3.4.1.

Corollary 3.5.8. *For a partition $\mathcal{P}$ over the concept class C, if the size of the largest subset P_i in $\mathcal{P}$ is less than $\frac{3/4}{\hat{\gamma}^C}$, then we have $R_{\mathcal{P}}(C) = O(nQ_{\mathcal{P}}(C)^3)$.*

Proof. Let C' be a subset of C for which $\gamma^{C'}$ equals $\hat{\gamma}^C$. We have that $|C'| \geq \frac{1}{\hat{\gamma}^C}$ by Definition 3.3.1. Thus any $\frac{3}{4}$ fraction of C' must intersect at least two subsets in $\mathcal{P}$, so C' must belong to $\mathcal{S}$. This forces $\hat{\gamma}^C_{\mathcal{P}} = \gamma^{C'} = \hat{\gamma}^C$. Moreover, we have that $|\mathcal{P}| \geq \frac{4}{3}\hat{\gamma}^C \cdot |C|$, and we know that $\frac{1}{N+1} \leq \hat{\gamma}^C \leq \frac{1}{2}$ by Lemma 3.3.8. Thus we have $\log|\mathcal{P}| \geq \log\frac{4}{3} - n + \log|C|$, and consequently $\frac{\log|\mathcal{P}|}{n} > \frac{\log|C|}{n} - 1$. Lemmas 3.5.2 and 3.5.4 yield $Q_{\mathcal{P}}(C) = \Omega(\frac{\log|\mathcal{P}|}{n} + \frac{1}{\sqrt{\hat{\gamma}^C_{\mathcal{P}}}}) = \Omega(\frac{\log|C|}{n} + \frac{1}{\sqrt{\hat{\gamma}^C}})$. Combining this with the bound $R_{\mathcal{P}}(C) = O(\frac{\log|C|}{\hat{\gamma}^C})$ (which clearly follows from Theorem 3.3.3 since the partition learning problem is no harder than the exact learning problem), we have that $R_{\mathcal{P}}(C)$ must be $O(nQ_{\mathcal{P}}(C)^3)$. □

We note here that we could have used any constant λ satisfying $\frac{2}{3} < \lambda < 1$ in Definition 3.5.3 in place of $3/4$, and obtained corresponding versions of Lemma 3.5.4 and the above corollary with λ in place of $3/4$.

Now we prove our main result of this subsection, showing that for *any* concept class C and any partition size bound $2 \leq k \leq |C|$ there is a partition of C into k pieces such that the classical and quantum query complexities are polynomially related:

Theorem 3.5.9. *Let C be any concept class and k any integer satisfying $2 \leq k \leq |C|$. Then there is a partition $\mathcal{P}$ of C with $|\mathcal{P}| = k$ for which we have $R_{\mathcal{P}}(C) = O(nQ_{\mathcal{P}}(C)^3)$.*

Proof. We will show that Algorithm 4 constructs a partition $\mathcal{P}$ with the desired properties. Algorithm 4 uses a slightly modified version of Algorithm 1, which we call Algorithm 3. Algorithm 3 differs from Algorithm 1 in that if the input $a_{\max}$ corresponds to a column which is flipped in the column flip on $C \setminus R$, then Algorithm 3 augments R by adding those concepts in the flipped version of $C \setminus R$ which yield 1 on $a_{\max}$ (note that by 1-sensitivity this is fewer than half of the concepts in $C \setminus R$), whereas Algorithm 1 adds those concepts which yield 1 on $a_{\max}$ in the unflipped (original) version of $C \setminus R$. Thus at each stage Algorithm 3 grows the set R by adding at most half of the remaining concepts in $C \setminus R$; we will need this property later. The analysis of Algorithm 1 carries over to show that the set $\mathcal{I}$ of inputs which Algorithm 3 constructs is of size at most $|\mathcal{I}| \leq \frac{1}{\hat{\gamma}^C}$.

Algorithm 3 A slightly modified version of Algorithm 1 to be used in generating a partition.

Require: C is 1-sensitive.

$R \leftarrow \emptyset, \mathcal{I} \leftarrow \emptyset, \mathcal{J} \leftarrow \emptyset$.
repeat
  Perform a column flip on $C \setminus R$ to make it 1-sensitive; call the resulting 1-sensitive matrix M.
  $a_{max} \leftarrow$ the input in $\{0,1\}^n \setminus \mathcal{I}$ at which the highest fraction of concepts in $C \setminus R$ yield 1.
  $\mathcal{I} \leftarrow \mathcal{I} \cup \{a_{\max}\}$.
  $R \leftarrow R \cup$ the set of concepts in M that yield 1 at input a_{max}.
  if the column corresponding to $a_{\max}$ in M was flipped relative to C **then**
    $\mathcal{J} \leftarrow \mathcal{J} \cup \{a_{\max}\}$.
  end if
until $|R| \geq |C|/2$.
Output $\leftarrow (\mathcal{I}, \mathcal{J})$.

At each iteration of the outer repeat loop, Algorithm 4 successively refines the partition $\mathcal{Q}$ until $|\mathcal{Q}| = k$. Let $C' \subseteq C$ be such that $\gamma^{C'} = \hat{\gamma}^C$. The first time Algorithm 4 passes through the inner repeat loop we will have $|C| = |S|$ and thus Algorithm 3 will be invoked on C'. We will write $C^\circ, C^\star$ to denote these sets S°, $S^\star$ of concepts that are formed out of C in this first iteration. The final partition $\mathcal{P}$ will ultimately be a refinement of the partition $\{C^\circ, C^\star\}$ obtained in this step; we will see later that this will force $\hat{\gamma}^C_{\mathcal{P}} = \hat{\gamma}^C$ (this is why the first iteration is treated differently than later iterations).

In addition to constructing the partition $\mathcal{P}$, the execution of Algorithm 4 should also be viewed as a "memoization" process in which various sets of inputs $\mathcal{I}(S), \mathcal{J}(S)$ and $\mathcal{K}(S)$ are defined to correspond to different sets of concepts S. These sets will be used during the execution of Algorithm 5 later. Roughly speaking, the division of S in each iteration depends only on the values on inputs in $\mathcal{I}(S)$, the set $\mathcal{J}(S)$ is used to keep track of the column flips Algorithm 3 performs, and the set $\mathcal{K}(S)$ keeps track of those inputs which need to be flipped to achieve 1-sensitivity.

We now explain the outer loop of Algorithm 4 in more detail. The algorithm works in a breadth-first fashion to successively refine the partition $\mathcal{Q}$, which is initially just $\{C\}$, into the final partition $\mathcal{P}$. After the first iteration of the outer loop, C has been partitioned into $\{C^\circ, C^\star\}$. Similarly, in the second iteration each of these sets is divided in two to give a four-way partition. The algorithm continues in this manner until the desired number of elements in the partition is reached. The main idea of the construction is that each division of a set S (after the first iteration) creates two pieces

S° and $S^\star$ of almost equal size as we shall describe below. Because degenerate divisions do not occur, we will see the algorithm will terminate after at most $O(\log k)$ iterations of the outer loop.

Recall from above that each invocation of Algorithm 3 in Algorithm 4 on a set S of concepts yields a set $\mathcal{I}(S)$ of at most $\frac{1}{\hat{\gamma}^S}$ many inputs. By flipping the output of concepts in S at inputs in $\mathcal{J}(S)$ in Step 14 of Algorithm 4, we ensure that the sets S° and $S^\star$ defined in steps 15 and 16 correspond precisely to the sets $C \setminus R$ and R of Algorithm 3 when it terminates. It thus follows from the termination condition of Algorithm 3 that $|S^\star|/|S| \geq \frac{1}{2}$. Recall also from the discussion in the first paragraph of this proof that the last iteration of the loop of Algorithm 3 adds at most half of the remaining concepts into the set R. Therefore we have that the set S° in Algorithm 4 must satisfy $|S^\circ|/|S| > \frac{1}{4}$. It follows from these bounds on $|S^\circ|/|S|$ and $|S^\star|/|S|$ that Algorithm 4 makes at most $O(\log k)$ many iterations through the outer loop.

It is therefore clear that at each iteration of the main loop of Algorithm 4 for which $|S| \geq 2$, each of the sets S° and $S^\star$ formed from S will be nonempty. This ensures that the algorithm will keep producing new elements in the partition until $|\mathcal{P}| = k$ is reached. The same argument shows that $C^\circ, C^\star$ are each nonempty and satisfy $|C^\circ \cap C'|/|C'| > \frac{1}{4}$ and $|C^\star \cap C'|/|C'| \geq \frac{1}{2}$. This implies that C' is an element of the set $\mathcal{S}$ of Definition 3.5.3: any subset $C'' \subseteq C'$ with $|C''| \geq \frac{3}{4}|C'|$ must intersect both C° and $C^\star$, and thus must intersect at least two subsets of $\mathcal{P}$ since $\mathcal{P}$ is a refinement of $\{C^\circ, C^\star\}$. Consequently we have $\hat{\gamma}^C_{\mathcal{P}} = \hat{\gamma}^C$.

To show that the partition $\mathcal{P}$ satisfies $R_{\mathcal{P}}(C) = O(nQ_{\mathcal{P}}(C)^3)$, we now give an analysis of the query complexity of learning $\mathcal{P}$ with both classical and quantum resources. As we will see, we need to give a classical upper bound and a quantum lower bound to obtain our goal.

In the classical case, we will show that Algorithm 5 makes $O(\frac{\log|\mathcal{P}|}{\hat{\gamma}^C})$ queries and successfully learns the partition $\mathcal{P}$. Using the sets $\mathcal{I}(S)$ which were defined by the execution of Algorithm 4, Algorithm 5 makes its way down the correct branch of the binary tree implicit in the successive refinements of Algorithm 4 to find the correct piece of the partition which contains c. More precisely, at the end of the t-th iteration of the outer loop of Algorithm 5, the set S which Algorithm 5 has just obtained will be identical to the piece c resides in of the partition constructed by Algorithm 4 at the end of the t-th iteration of its outer loop. As shown above, it takes $O(\log k) = O(\log|\mathcal{P}|)$ iterations until the subset which the target concept c lies in is reached. Moreover, by the same argument in

Algorithm 4 Constructing a partition for which $R_{\mathcal{P}}(C)$ and $Q_{\mathcal{P}}(C)$ are polynomially related.

$\mathcal{Q} \leftarrow \{C\}$
repeat
  $\mathcal{R} \leftarrow \emptyset$.
  repeat
    $S \leftarrow$ an element in $\mathcal{Q}$
    if $|S| \geq 2$ **then**
      if $|C| = |S|$ **then**
        Let $\mathcal{K}(S)$ denote the inputs which, if flipped, would make C' be 1-sensitive (C' is defined in the 2nd paragraph of the proof of Theorem 3.5.9). Flip the values of concepts in S at inputs in $\mathcal{K}(S)$.
        $(\mathcal{I}(S), \mathcal{J}(S)) \leftarrow$ The output of Algorithm 3 invoked with C'.
      else
        Let $\mathcal{K}(S)$ denote the inputs which, if flipped, would make S be 1-sensitive. Flip the values of concepts in S at inputs in $\mathcal{K}(S)$.
        $(\mathcal{I}(S), \mathcal{J}(S)) \leftarrow$ The output of Algorithm 3 invoked with input S.
      end if
      Flip the values of concepts in S at those inputs in $\mathcal{J}(S)$.
      $S^{\circ} \leftarrow \{$the concepts in S that yield 0 for each $x \in \mathcal{I}(S)\}$.
      $S^{\star} \leftarrow S \setminus S^{\circ}$.
      Flip the values of concepts in $S^{\circ}, S^{\star}$ for all elements in $\mathcal{J}(S)$.
      Flip the values of concepts in $S^{\circ}, S^{\star}$ for all elements in $\mathcal{K}(S)$.
      $\mathcal{R} \leftarrow \mathcal{R} \cup \{S^{\circ}, S^{\star}\}$.
    else
      $\mathcal{R} \leftarrow \mathcal{R} \cup \{S\}$.
    end if
    $\mathcal{Q} \leftarrow \mathcal{Q} \setminus \{S\}$.
  until $\mathcal{Q} = \emptyset$ OR $|\mathcal{Q}| + |\mathcal{R}| = k$.
  $\mathcal{Q} \leftarrow \mathcal{Q} \cup \mathcal{R}$.
until $|\mathcal{Q}| = k$.
$\mathcal{P} \leftarrow \mathcal{Q}$.

Lemma 3.3.10, Algorithm 3 always outputs a set of inputs $\mathcal{I}(S)$ with size at most $\frac{1}{\hat{\gamma}^S} \leq \frac{1}{\hat{\gamma}^C}$ when invoked on a set of concepts S. Therefore at each of these $O(\log |\mathcal{P}|)$ iterations Algorithm 5 makes at most $\frac{1}{\hat{\gamma}^C}$ many queries. Thus Algorithm 5 is a classical algorithm that learns $\mathcal{P}$ using $O(\frac{\log |\mathcal{P}|}{\hat{\gamma}^C})$ queries, so we have $R_{\mathcal{P}}(C) = O(\frac{\log |\mathcal{P}|}{\hat{\gamma}^C})$.

In the quantum case: since we have $\hat{\gamma}^C_{\mathcal{P}} = \hat{\gamma}^C$, by Lemma 3.5.4 any quantum algorithm learning $\mathcal{P}$ should perform $\Omega(\frac{1}{\sqrt{\hat{\gamma}^C}})$ quantum membership queries. Combining this result with that of Lemma 3.5.2, we have that $Q_{\mathcal{P}}(C) = \Omega(\frac{\log |\mathcal{P}|}{n} + \frac{1}{\sqrt{\hat{\gamma}^C}})$. Combining this inequality with the classical upper bound $R_{\mathcal{P}} = O(\frac{\log |\mathcal{P}|}{\hat{\gamma}^C})$ from Algorithm 5, we have that $R_{\mathcal{P}}(C) = O(nQ_{\mathcal{P}}(C)^3)$ for this partition $\mathcal{P}$, and we are done. □

Algorithm 5 A classical algorithm learning $\mathcal{P}$.

$S \leftarrow C$
repeat
  Flip the values of concepts in S at those inputs in $\mathcal{K}(S)$.
  Flip the values of concepts in S at those inputs in $\mathcal{J}(S)$.
  Classically query the given oracle implementing c at all elements in $\mathcal{I}(S)$.
  $Z \leftarrow$TRUE if c yields 0 for all elements in $\mathcal{I}(S)$. $Z \leftarrow$FALSE otherwise.
  if Z **then**
    $S \leftarrow$ {the concepts in S that yield 0 for each $x \in \mathcal{I}(S)$}.
  else
    $S \leftarrow$ {the concepts in S that yield 1 for at least one $x \in \mathcal{I}(S)$}.
  end if
  Flip the values of concepts in S at those inputs in $\mathcal{J}(S)$.
  Flip the values of concepts in S at those inputs in $\mathcal{K}(S)$.
until $S \in \mathcal{P}$.
Output$\leftarrow S$.

3.5.2 A partition problem with a large quantum-classical gap

In the previous subsection we showed that for any concept class and any partition size bound, there is a partition problem for which the classical and quantum query complexities are polynomially related. In this section, by adapting a result of Simon [Sim97] we show that for a wide range of values of the partition size bound, it is possible for the classical query complexity to be superpolynomially larger than the quantum query complexity:

Theorem 3.5.10. *Let $n = m + \log m$. For any value $1 \leq \ell < m$, there is a concept class C over $\{0,1\}^n$ with $|C| < 2^{m\ell-\ell^2+\ell+m2^{m-\ell}}$ and a partition $\mathcal{P}$ of C with $|\mathcal{P}| > 2^{m\ell-\ell^2-\ell}$ such that $R_{\mathcal{P}}(C) = \Omega(2^{m-\ell})$ and $Q_{\mathcal{P}}(C) =$poly(m)).*

Taking $\ell = m - \alpha(m)$ where $\alpha(m)$ is any function which is $\omega(\log m)$, we obtain $R_{\mathcal{P}}(C) = m^{\omega(1)}$ whereas $Q_{\mathcal{P}}(C) =$poly(m), for a superpolynomial separation between classical and quantum query complexity. Such a choice of ℓ gives $|\mathcal{P}| = 2^{\Omega(m)}$ whereas $|C|$ is roughly $2^{m \cdot 2^{\alpha(m)}}$. Note that the size of $|C|$ can be made to be $2^{\beta(m)}$ for any slightly superpolynomial function $\beta(m)$ via a suitable choice of $\alpha(m) = \omega(\log m)$. This means that viewed as a function of $|C|$, it is possible for $|\mathcal{P}|$ to be as large as $2^{(\log |C|)^{\varepsilon(|C|)}}$ for any function $\varepsilon(\cdot) = o(1)$ and still have the classical query complexity be superpolynomially larger than the quantum query complexity.

Proof of Theorem 3.5.10: We will use a result of Simon [Sim97] who considers functions f :

$\{0,1\}^m \to \{0,1\}^m$. Any such function $f : \{0,1\}^m \to \{0,1\}^m$ can equivalently be viewed as a function $\tilde{f} : (\{0,1\}^m \times \{1,\ldots,m\}) \to \{0,1\}$ where $\tilde{f}(x,j)$ equals $f(x)_j$, the j-th bit of $f(x)$. It is easy to see that we can simulate a call to an oracle for $f : \{0,1\}^m \to \{0,1\}^m$ by making m membership queries to the oracle for $\tilde{f}$, in both the classical and quantum case. This extra factor of m is immaterial for our bounds, so we will henceforth discuss functions f which map $\{0,1\}^m$ to $\{0,1\}^m$.

We view the input space $\{0,1\}^m$ as the vector space $\mathbb{F}_2^m$. Given a ℓ-dimensional vector subspace $V \subset \mathbb{F}_2^m$, we say that a function $f : \{0,1\}^m \to \{0,1\}^m$ is *V-invariant* if the following condition holds: $f(x) = f(y)$ if and only if $x = y \oplus v$ for some $v \in V$. Thus a V-invariant function f is defined by the $2^{m-\ell}$ distinct values it takes on representatives of the $2^{m-\ell}$ cosets of V. The concept class C is the class of all functions f which are V-invariant for some ℓ-dimensional vector subspace V of $\mathbb{F}_2^m$.

A simple counting argument shows that there are

$$N_{m,\ell} = \frac{(2^m-1)(2^m-2)(2^m-4)\cdots(2^m-2^{\ell-1})}{(2^\ell-1)(2^\ell-2)(2^\ell-4)\cdots(2^\ell-2^{\ell-1})}$$

many ℓ-dimensional subspaces of $\mathbb{F}_2^m$. This is because there are $(2^m-1)(2^m-2)(2^m-4)\cdots(2^m-2^{\ell-1})$ ways to choose an ordered list of ℓ linearly independent vectors to form a basis for V, and given any V there are $(2^\ell-1)(2^\ell-2)(2^\ell-4)\cdots(2^\ell-2^{\ell-1})$ ordered lists of vectors from V which could serve as a basis.

We define the partition $\mathcal{P}$ to divide C into $N_{m,\ell}$ equal-size subsets, one for each ℓ-dimensional vector subspace V; the subset of concepts corresponding to a given V is precisely those functions which are V-invariant. For any given ℓ-dimensional subspace V, the number of functions that are V-invariant is

$$I_{m,\ell} = 2^m(2^m-1)(2^m-2)\cdots(2^m-2^{m-\ell}+1)$$

since one can uniquely define such a function by specifying distinct values to be attained on each of the $2^{m-\ell}$ coset representatives.

Therefore we have $|C| = N_{m,\ell} \cdot I_{m,\ell}$, and it is easy to check that $2^{m\ell-\ell^2-\ell} \leq N_{m,\ell} \leq 2^{m\ell-\ell^2+\ell}$ and $I_{m,\ell} \leq 2^{m2^{m-\ell}}$. It remains only to analyze the quantum and classical query complexities.

For the quantum case, it follows easily from Simon's analysis of his algorithm in [Sim97] that for any V-invariant f, each iteration of Simon's algorithm (which requires a single quantum query to f) will yield a vector that is independently and uniformly drawn from the $(m-\ell)$-dimensional subspace $V^{\perp}$. A standard analysis shows that after $O(m)$ iterations, with very high probability we will have obtained a set of vectors that span $V^{\perp}$; from these it is easy to identify V and thus the piece of the partition to which f belongs.

For the classical case, an analysis much like that of ([Sim97], Section 3.2) can be used to show that any classical algorithm which correctly identifies the vector subspace V with high probability must make $2^{\Omega(m-\ell)}$ many queries; since the proof is similar to [Sim97] we only sketch the main ideas here. We say that a sequence of queries is *good* if it contains two distinct queries which yield the same output (i.e. two queries x, y which have $(x \oplus y) \in V$), and otherwise the sequence is *bad*. The argument of [Sim97] applied to our setting shows that if the target vector subspace is chosen uniformly at random, then any classical algorithm making $M = 2^{(m-\ell)/3}$ queries makes a good sequence of queries with very small probability. On the other hand, if a sequence of $M = 2^{(m-\ell)/3}$ queries is bad, then this restricts the possibilities for V by establishing a set S of $\binom{M}{2} < 2^{2(m-\ell)/3}$ many "forbidden" vectors from $\mathbb{F}_2^m$ which must not belong to the target vector space V (since for each pair of elements x, y in M we know that $(x \oplus y) \notin V$). Given a fixed nonzero vector $z \in \mathbb{F}_2^m$, we have that a random ℓ-dimensional vector space W contains z with probability $\frac{2^\ell - 1}{2^m - 1} < \frac{2^\ell}{2^m}$, and consequently the probability that a random W contains any element of S is at most $2^{2(m-\ell)/3} \cdot \frac{2^\ell}{2^m} = 2^{(\ell - m)/3}$, which is less than $1/2$ if $\ell < m - 6$ (and if $\ell \geq m - 6$ the bound of the theorem is trivially true). Thus at least half of the $N_{m,\ell}$ possible ℓ-dimensional vector subspaces are compatible with any given bad sequence of $2^{(m-\ell)/3}$ queries, so the classical algorithm cannot have identified the right subspace with non-negligible probability. □

3.6 Quantum versus classical PAC learning

3.6.1 The quantum PAC learning model

The quantum PAC learning model was introduced by Bshouty and Jackson in [BJ99]. A quantum PAC learning algorithm is defined analogously to a classical PAC algorithm, but instead of having

access to a random example oracle $\mathsf{EX}(c, \mathcal{D})$ it can (repeatedly) access a *quantum superposition* of labeled examples. The goal of constructing a classical Boolean circuit h which is an ϵ-approximator for c with probability $1 - \delta$ is unchanged. More precisely, for $\mathcal{D}$ a distribution over $\{0,1\}^n$, the *quantum example oracle associated to c under $\mathcal{D}$* $\mathsf{QEX}(c, \mathcal{D})$ is the quantum oracle whose query acts on the computational basis state $|0^n, 0\rangle$ as follows:

$$|0^n, 0\rangle \mapsto \sum_{x \in \{0,1\}^n} \sqrt{\mathcal{D}(x)} |x, c(x)\rangle.$$

We leave the action of a $\mathsf{QEX}(c, \mathcal{D})$ query undefined on other basis states, and we require that a quantum PAC learning algorithm may only invoke a $\mathsf{QEX}(c, \mathcal{D})$ query on the basis state $|0^n, 0\rangle$. We sometimes refer to a query to the quantum example oracle as *a quantum example*. It is easy to verify (see [BJ99]) that a $\mathsf{QEX}(c, \mathcal{D})$ oracle can simulate a classical $\mathsf{EX}(c, \mathcal{D})$ oracle.

As noted in Lemma 6 of [BJ99], we may assume without loss of generality (by renumbering qubits) that all $\mathsf{QEX}(c, \mathcal{D})$ queries in a quantum PAC learning algorithm occur sequentially at the beginning of the algorithm's execution and that the t-th query of $\mathsf{QEX}(c, \mathcal{D})$ affects the qubits $(t-1)(n+1)+1, (t-1)(n+1)+2, \ldots, t(n+1)$. After all T $\mathsf{QEX}(c, \mathcal{D})$ queries have been performed, the algorithm performs a fixed unitary transformation and then a measurement takes place. (See [BJ99, SG04] for more details on the quantum PAC learning model.) The *quantum sample complexity* is the number of queries T of QEX which the quantum PAC learning algorithm performs, i.e. the number of QEX gates in the quantum circuit corresponding to the quantum PAC learning algorithm.

The following definition plays an important role in the sample complexity of both classical and quantum PAC learning:

Definition 3.6.1. If C is a concept class over some domain X and $W \subseteq X$, we say that W is *shattered by C* if for every $W' \subseteq W$, there exists a $c \in C$ such that $W' = c \cap W$. The *Vapnik-Chervonenkis dimension of C*, VC-DIM(C), is the cardinality of the largest $W \subseteq X$ such that W is shattered by C.

3.6.2 Known results on quantum versus classical PAC learning

The classical sample complexity of PAC learning has been intensively studied and nearly matching upper and lower bounds are known:

Theorem 3.6.2.

- *[EHKV89] Any classical (ϵ, δ)-PAC learning algorithm for a non-trivial concept class C of VC dimension d must have classical sample complexity $\Omega(\frac{1}{\epsilon}\log\frac{1}{\delta} + \frac{d}{\epsilon})$.*
- *[BEHW89] Any concept class C of VC dimension d can be (ϵ, δ)-PAC learned by a classical algorithm with sample complexity $O(\frac{1}{\epsilon}\log\frac{1}{\delta} + \frac{d}{\epsilon}\log\frac{1}{\epsilon})$.*

Servedio and Gortler [SG04] gave a lower bound on the quantum sample complexity of PAC learning. They showed that for any concept class C of VC dimension d over $\{0,1\}^n$, if the distribution $\mathcal{D}$ is uniform over the d examples in some shattered set S, then even if the learning algorithm is allowed to make quantum membership queries on any superposition of inputs in the domain S, any algorithm which with high probability outputs a high-accuracy hypothesis with respect to $\mathcal{D}$ must make at least $\Omega(\frac{d}{n})$ many queries. Such membership queries can simulate $\mathsf{QEX}(c, \mathcal{D})$ queries since the support of $\mathcal{D}$ is S, and thus this gives a lower bound on the sample complexity of quantum PAC learning with a QEX oracle.

3.6.3 Improved lower bounds on quantum sample complexity of PAC Learning

In this section we give improved lower bounds on the sample complexity of quantum (ϵ, δ)-PAC learning algorithms for concept classes C of VC dimension d. These new bounds nearly match the classical lower bounds of [EHKV89].

We first note that the $\Omega(\frac{d}{n})$ lower bound of [SG04] can be easily strengthened to $\Omega(d)$:

Observation 3.6.3. *Let C be any concept class of VC dimension d and let $\mathcal{D}$ be the uniform distribution over a shattered set S of size d. Then any quantum learning algorithm which*

- *can make quantum membership queries on any superposition of inputs in the domain S, and*
- *with high probability outputs a hypothesis with error rate at most $\epsilon = \frac{1}{10}$*

must make at least $\frac{d}{100}$ queries (and consequently the sample complexity of PAC learning C with a QEX *oracle is $\Omega(d)$).*

Recall that in the exact learning model, the concept class C of all parity functions over n Boolean variables has VC dimension $d = n$ yet can be exactly learned with one call to a quantum membership oracle using the Bernstein-Vazirani algorithm [BV97]. In light of this, we feel that this improvement from $\Omega(\frac{d}{n})$ to $\Omega(d)$ is somewhat unexpected, and may even at first appear contradictory. The key to the apparent contradiction is that the Bernstein-Vazirani algorithm makes its membership query on a superposition of all 2^n inputs in $\{0,1\}^n$, not just the n inputs in a fixed shattered set S.

Proof. It suffices to slightly sharpen the proof of Theorem 4.2 from [SG04]. The key observation is that since queries always have zero amplitude on computational basis states outside of the shattered set S, the effective value of the domain size N is $|S| = d$ rather than $|\{0,1\}^n| = 2^n$. With this modification, at the end of the proof of Theorem 4.2 we obtain the inequality $N_0 = \sum_{i=0}^{2T} \binom{d}{i} \geq 2^{d/6}$ where T is the quantum query complexity of the algorithm (instead of the inequality $\sum_{i=0}^{2T} \binom{2^n}{i} \geq 2^{d/6}$ which appears in [SG04]). Now standard tail bounds on binomial coefficients (see e.g. Appendix 9 of [KV94]) show that $T > \frac{d}{100}$. □

We now give lower bounds on the quantum query complexity of (ϵ, δ)-PAC learning which depend on ϵ and δ. We require the following definition and fact:

Definition 3.6.4. A concept class C is said to be *trivial* if either C contains only one concept, or C contains exactly two concepts c_0, c_1 with each $x \in \{0,1\}^n$ belonging to exactly one of c_0, c_1.

Fact 3.6.5 (See [Shi00])**.** *Let $|\psi^{(0)}\rangle, |\psi^{(1)}\rangle$ represent states of a quantum system and Π be the linear projection associated to a measurement in the computational basis such that $\langle\psi^{(0)}|\Pi|\psi^{(0)}\rangle \geq 1-\delta$ and $\langle\psi^{(1)}|\Pi|\psi^{(1)}\rangle \leq \delta$ for some $\delta > 0$. Then we have $|\langle\psi^{(0)}|\psi^{(1)}\rangle| \leq 2\sqrt{\delta(1-\delta)}$.*

It is clear that a trivial concept class can be learned exactly from any single (classical) example. For nontrivial concept classes [BEHW89] gave a classical sample complexity lower bound of $\Omega(\frac{1}{\epsilon}\log\frac{1}{\delta})$. We now extend this bound to the quantum setting:

Lemma 3.6.6. *Any quantum algorithm with a* $\mathsf{QEX}(c, \mathcal{D})$ *oracle which* (ϵ, δ)*-learns a non-trivial concept class must have quantum sample complexity* $\Omega(\frac{1}{\epsilon}\log\frac{1}{\delta})$.

Proof. Since C is non-trivial, without loss of generality we may assume that there are two inputs x_0, x_1 and two concepts $c_0, c_1 \in C$ such that $c_0(x_0) = c_1(x_0) = 0$ while $c_0(x_1) = 0, c_1(x_1) = 1$.

Let $\mathcal{D}$ be the distribution where $\mathcal{D}(x_0) = 1 - 3\epsilon$ and $\mathcal{D}(x_1) = 3\epsilon$. Under this distribution, no hypothesis which is ϵ-accurate for c_0 can be ϵ-accurate for c_1 and vice versa.

Let $|\psi_T^{(i)}\rangle$ be the state of the system immediately after the T queries of $\mathsf{QEX}(c_i, \mathcal{D})$ are performed, and U be the fixed unitary transformation before the measurement. Then we have

$$\langle\psi_T^{(i)}|\underbrace{x_0, 0, x_0, 0, \ldots, x_0, 0}_{\text{repeated } T \text{ times}}, 0\ldots, 0\rangle = (1-3\epsilon)^{T/2}, \quad \text{for } i = 0, 1.$$

It is easy to see that any other computational basis state $|\ldots, x_1, b, \ldots\rangle$ which has nonzero amplitude in $|\psi_T^{(b)}\rangle$ must have zero amplitude in the other possible state $|\psi_T^{(1-b)}\rangle$, because c_0 and c_1 disagree on x_1. Consequently we have $\langle\psi_T^{(0)}|U^\dagger U|\psi_T^{(1)}\rangle = \langle\psi_T^{(0)}|\psi_T^{(1)}\rangle = (1-3\epsilon)^T$. When the algorithm terminates, the measurement outcome of the state $U|\psi_T^{(i)}\rangle$ will determine the result of the algorithm for concept c_i. Therefore, if $(1-3\epsilon)^T > 2\sqrt{\delta(1-\delta)}$ then Fact 3.6.5 dictates there is some output hypothesis which occurs with probability greater than δ whether the target is c_0 or c_1; but this cannot be the case for an (ϵ, δ)-PAC learning algorithm. Thus we must have $(1-3\epsilon)^{2T} \leq 4\delta$ yielding $T = \Omega(\frac{1}{\epsilon}\log\frac{1}{\delta})$. □

Ehrenfeucht *et al.* [EHKV89] obtained a $\Omega(\frac{d}{\epsilon})$ lower bound for classical PAC learning by considering a distribution $\mathcal{D}$ which distributes $\Theta(\epsilon)$ weight evenly over all but one of the elements in a shattered set. In other words under $\mathcal{D}$ one element in the shattered set has weight $1 - \Theta(\epsilon)$ and all the remaining $d-1$ elements has equal weight $\frac{\Theta(\epsilon)}{d-1}$. We use such a distribution to obtain the following quantum lower bound (no attempt has been made to optimize constants):

Theorem 3.6.7. *Let* C *be any concept class of VC dimension* $d+1$*. Let* $\delta = 1/5$*. Then we have that for sufficiently large* d *(i.e.* $d \geq 625$ *suffices) and any* $0 < \epsilon < \frac{1}{32}$*, any quantum algorithm with a* $\mathsf{QEX}(c, \mathcal{D})$ *oracle which* (ϵ, δ)*-learns* C *must have quantum sample complexity at least* $\frac{\sqrt{d}}{10000\epsilon}$.

Proof. Let $\{x_0, x_1, \ldots, x_d\}$ be a set of inputs which is shattered by C. We consider the distribution

$\mathcal{D}$, first introduced by [EHKV89], which has $\mathcal{D}(x_0) = 1-8\epsilon$ and $\mathcal{D}(x_i) = \frac{8\epsilon}{d}$ for $i = 1, \ldots, d$. Let $H(x) = -x \log x - (1-x)\log(1-x)$ denote the binary entropy function. As is noted in [SG04], there exists a set $s^1, \ldots, s^A$ of d-bit strings such that for all $i \neq j$ the strings s^i and s^j differ in at least $d/4$ positions where $A \geq 2^{d(1-H(1/4))} > 2^{d/6}$. For each $i = 1, \ldots, A$ let c_i be a concept such that

- $c_i(x_0) = 0$, and
- the d-bit string $(c_i(x_1), \ldots, c_i(x_d))$ is s^i.

The existence of such concepts follows from Definition 3.6.1. Since we have $\epsilon < \frac{1}{32}$, our quantum PAC learning algorithm should successfully distinguish between any two target concepts c_i and c_j with confidence at least $1 - \delta = \frac{4}{5}$. Moreover, without loss of generality we may suppose that $\epsilon < \frac{1}{100\sqrt{d}}$ since otherwise Observation 3.6.3 yields the required lower bound.

We shall make use of the following standard inequality:

$$(1-x)^\ell \geq 1 - x\ell, \text{if } x\ell < 1 \text{ for } \ell \in \mathbb{Z}^+, x \in \mathbb{R}^+. \tag{3.6.1}$$

Given a target concept c, we write $|\xi_{i_1,i_2,\ldots,i_t}\rangle$ to denote the basis state

$$|\xi_{i_1,i_2,\ldots,i_t}\rangle = |x_{i_1}, c(x_{i_1}), x_{i_2}, c(x_{i_2}), \ldots x_{i_t}, c(x_{i_t})\rangle.$$

We define the state $|\phi_t\rangle$ to be

$$|\phi_t\rangle = (1-8\epsilon)^{t/2}|\xi_{0,0,\ldots,0}\rangle + (1-8\epsilon)^{\frac{t-1}{2}}\sqrt{\frac{8\epsilon}{d}}\sum_{i=1}^{d}(|\xi_{i,0,0,\ldots,0}\rangle + |\xi_{0,i,0,\ldots,0}\rangle + \ldots + |\xi_{0,0,\ldots,i}\rangle) + \alpha|z\rangle.$$

Here $|z\rangle$ is some canonical basis state which is distinct from, and hence orthogonal to, all states of the form $|\xi_{i_1,i_2,\ldots,i_t}\rangle$, e.g. we could take $z = |x_1, c(x_1), x_1, 1-c(x_1), 0, 0, \ldots, 0\rangle$. The scalar α is a suitable normalizing coefficient so that the Euclidean norm of $|\phi_t\rangle$ is 1.

Let $|\psi_t\rangle$ denote the state of the quantum register after t invocations of $\mathsf{QEX}(c, \mathcal{D})$ have occurred. It is easy to see from the definition of the $\mathsf{QEX}(c, \mathcal{D})$ oracle that the amplitude of $|\psi_t\rangle$ on the basis state $|\xi_{0,\ldots,0}\rangle$ will be $(1-8\epsilon)^{t/2}$, and that for each of the td many basis states $|\xi_{0,0,\ldots,0,i,0,\ldots,0}\rangle$

(where i ranges over all t positions and ranges in value from 1 to d) the amplitude of $|\psi_t\rangle$ on this basis state will be $(1-8\epsilon)^{\frac{t-1}{2}}\sqrt{\frac{8\epsilon}{d}}$. We thus have that $\langle\psi_t|\phi_t\rangle = (1-8\epsilon)^t\left(1+\frac{8t\epsilon}{1-8\epsilon}\right)$.

If we let $t = \frac{1}{100\epsilon\sqrt{d}}$ (note that $t \geq 1$ by our assumption that $\epsilon < \frac{1}{100\sqrt{d}}$), we then have that $(1-8\epsilon)^t > (1-\frac{1}{12\sqrt{d}})$ by (3.6.1), and it is easy to check that $(1+\frac{8t\epsilon}{1-8\epsilon}) > 1+\frac{1}{12\sqrt{d}}$ from the bounds on ϵ and d in the theorem statement. We thus have that

$$\langle\psi_t|\phi_t\rangle > 1 - \frac{1}{144d}. \tag{3.6.2}$$

Now let us consider what happens if we replace each successive block of $t = \frac{1}{100\epsilon\sqrt{d}}$ invocations of the $\mathsf{QEX}(c,\mathcal{D})$ oracle in our PAC learning algorithm with the transformation

$$Q : |0^{t(n+1)}\rangle \mapsto |\phi_t\rangle.$$

If the learning algorithm makes a total of T calls to $\mathsf{QEX}(c,\mathcal{D})$ then we perform T/t replacements. After all T/t calls to Q in the modified algorithm, the initial state $|0\ldots0\rangle$ evolves into the state $|\varphi\rangle = \underbrace{|\phi_t\rangle\ldots|\phi_t\rangle}_{T/t \text{ times}}|0\ldots0\rangle$.

By equation (3.6.2), we have that $\langle\psi_T|\varphi\rangle > (1-\frac{1}{144d})^{T/t}$. If $T/t \leq d/100$ (i.e. if $T \leq \frac{\sqrt{d}}{10000\epsilon}$), then by (3.6.1) this lower bound is at least $\frac{143}{144}$, and this implies (since the original algorithm with T many $\mathsf{QEX}(c,\mathcal{D})$ calls was successful on each target c_i with probability at least $4/5$) that the modified algorithm which makes at most $d/100$ many calls to Q is successful with probability at least $2/3$. However, the exact same polynomial-based argument which underlies the $\Omega(d/n)$ lower bound for PAC learning proved in [SG04] (and the improved $\frac{d}{100}$ lower bound of Observation 3.6.3) implies that it is impossible for our modified algorithm, which makes at most $d/100$ many calls to Q, to succeed on each target c_i with probability at least $2/3$. (The crux of that proof is that each invocation of a black-box oracle for c increases the degree of the polynomial associated to the coefficient of each basis state by at most one. This property is easily seen to hold for Q as well – after r queries to the Q oracle, the coefficient of each basis state can be expressed as a degree-r polynomial in the indeterminates $c(x_1),\ldots,c(x_d)$). This proves that we must have $T/t > d/100$, which gives the conclusion of the theorem. □

Combining our results, we obtain the following quantum version of the classical $\Omega(\frac{1}{\epsilon}\log\frac{1}{\delta}+\frac{d}{\epsilon})$ bound:

Theorem 3.6.8. *Any quantum (ϵ,δ)-PAC learning algorithm for a concept class of VC dimension d must make at least $\Omega(\frac{1}{\epsilon}\log\frac{1}{\delta}+d+\frac{\sqrt{d}}{\epsilon})$ calls to the* QEX *oracle.*

Chapter 4

Learning Unions of $\omega(1)$-Dimensional Rectangles

This chapter is based on the article [AS06], chosen as the recipient of the E. M. Gold award by the committee of the 17th International Conference on Algorithmic Learning Theory. It will appear in the journal Theoretical Computer Science special issue ALT 2006.

4.1 Introduction

4.1.1 Motivation

The learnability of Boolean valued functions defined over the domain $[b]^n = \{0, 1, \ldots, b-1\}^n$ has long elicited interest in computational learning theory literature. In particular, much research has been done on learning various classes of "unions of rectangles" over $[b]^n$ (see e.g. [BK98, CH96, CM94, GGM94, Jac97, MW98]), where a rectangle is a conjunction of properties of the form "the value of attribute x_i lies in the range $[\alpha_i, \beta_i]$". One motivation for studying these classes is that they are a natural analogue of classes of DNF (Disjunctive Normal Form) formulae over $\{0,1\}^n$; for instance, it is easy to see that in the case $b = 2$ any union of s rectangles is simply a DNF with s terms.

Since the description length of a point $x \in [b]^n$ is $n \log b$ bits, a natural goal in learning functions over $[b]^n$ is to obtain algorithms which run in time $\mathrm{poly}(n \log b)$. Throughout this chapter we

refer to such algorithms with $\mathrm{poly}(n \log b)$ runtime as *efficient* algorithms. In this chapter we give efficient algorithms which can learn several interesting classes of unions of rectangles over $[b]^n$ in the model of uniform distribution learning with membership queries and in the uniform distribution quantum PAC learning model with quantum examples.

4.1.2 Previous results

In a breakthrough result a decade ago, Jackson [Jac97] gave the Harmonic Sieve algorithm and proved that it can learn any s-term DNF formula over n Boolean variables in $\mathrm{poly}(n, s)$ time. In fact, Jackson showed that the algorithm can learn any s-way majority of parities in $\mathrm{poly}(n, s)$ time; this is a richer set of functions which includes all s-term DNF formulae. The Harmonic Sieve algorithm works by boosting a Fourier-based weak learning algorithm, which is a modified version of an earlier algorithm due to Kushilevitz and Mansour [KM93].

In [Jac97] Jackson also described an extension of the Harmonic Sieve algorithm to the domain $[b]^n$. His main result for $[b]^n$ is an algorithm that can learn any union of s rectangles over $[b]^n$ in $\mathrm{poly}(s^{b \log\log b}, n)$ time; note that this runtime is $\mathrm{poly}(n, s)$ if and only if b is $\Theta(1)$ (and the runtime is clearly exponential in b for any s).

There has also been substantial work on learning various classes of unions of rectangles over $[b]^n$ in the more demanding model of exact learning from membership and equivalence queries. Some of the subclasses of unions of rectangles which have been considered in this setting are:

Subclasses where the dimension of each rectangle is $O(1)$: Beimel and Kushilevitz [BK98] established an algorithm learning any union of s $O(1)$-dimensional rectangles using equivalence queries only in $\mathrm{poly}(n, s, \log b)$ time steps.

Subclasses where the number of rectangles is limited: In [BK98] an algorithm is given which exactly learns any union of $O(\log n)$ many rectangles in $\mathrm{poly}(n, \log b)$ time using membership and equivalence queries. Earlier, Maass and Warmuth [MW98] gave an algorithm which uses only equivalence queries and can learn any union of $O(1)$ rectangles in $\mathrm{poly}(n, \log b)$ time.

Subclasses where the rectangles are disjoint: If no input $x \in [b]^n$ belongs to more than one

rectangle, then [BK98] can learn a union of s such rectangles in $\mathrm{poly}(n, s, \log b)$ time with membership and equivalence queries.

4.1.3 The techniques and results of this chapter

Because efficient learnability is established for unions of $O(\log n)$ arbitrary dimensional rectangles by [BK98] in a more demanding model, we are interested in achieving positive results when the number of rectangles is strictly larger. Thus the cases we study involve at least $\mathrm{poly}(\log(n \log b))$ and sometimes as many as $\mathrm{poly}(n \log b)$ rectangles.

We start by describing a new variant of the Harmonic Sieve algorithm for learning functions defined over $[b]^n$; we call this new algorithm the Generalized Harmonic Sieve, or GHS. The key difference between GHS and Jackson's algorithm for $[b]^n$ is that whereas Jackson's algorithm used a weak learning algorithm whose runtime is $\mathrm{poly}(b)$, the GHS algorithm uses a $\mathrm{poly}(\log b)$ time weak learning algorithm described in recent work of Akavia *et al.* [AGS03].

The GHS algorithm we describe runs under the assumptions of either of the following learning models:

- Learning with membership queries with respect to uniform distribution, as described in Section 4.3.
- Learning with uniform quantum examples with respect to uniform distribution (uniform distribution quantum PAC learning), as described in Section 4.7.

We then apply GHS to learn various classes of functions defined in terms of "b-literals" (see Section 4.2 for a precise definition; roughly speaking a b-literal is like a 1-dimensional rectangle). We first show the following result:

- (See Theorem 4.4.1) The concept class of s-MAJORITY of r-PARITY of b-literals where $s = \mathrm{poly}(n \log b)$, $r = O(\frac{\log(n \log b)}{\log \log(n \log b)})$ is efficiently learnable under GHS.

Learning this class has immediate applications for our goal of "learning unions of rectangles"; in particular, it follows that

- (See Theorem 4.6.3) The concept class of s-MAJORITY of r-dimensional rectangles is efficiently learnable under GHS provided $s = \mathrm{poly}(n \log b)$ and $r = O(\frac{\log(n \log b)}{\log \log(n \log b)})$.

This clearly implies efficient learnability for unions (as opposed to majorities) of s such rectangles as well.

We then employ a technique of restricting the domain $[b]^n$ to a much smaller set and adaptively expanding this set as required. This approach was used in the exact learning framework by Beimel and Kushilevitz [BK98]; by an appropriate modification we adapt the underlying idea to the uniform distribution membership query framework. Using this approach in conjunction with GHS we obtain almost a quadratic improvement in the dimension of the rectangles if the number of terms is guaranteed to be small:

- (See Theorem 4.6.5) The concept class of unions of $s = \mathrm{poly}(\log(n \log b))$ many r-rectangles where $r = O(\frac{\log^2(n \log b)}{(\log\log(n \log b) \log\log\log(n \log b))^2})$ is efficiently learnable.

Afterwards, we consider the case of disjoint rectangles (also studied by [BK98] as mentioned above), and improve the depth of our circuits by 1 provided that the rectangles connected to the same OR gate are disjoint:

- (See Corollary 4.6.8) The concept class of s-MAJORITY of t-OR of disjoint r-rectangles is efficiently learnable under GHS provided $s, t = \mathrm{poly}(n \log b)$ and $r = O(\frac{\log(n \log b)}{\log\log(n \log b)})$.

Finally in Section 4.7 we explore consequences of these results towards learning with quantum computation. We translate the derived positive learnability results to the uniform distribution quantum PAC learning model, in which no access to classical or quantum membership queries is permitted.

4.1.4 Organization of this chapter

In Section 4.2 we give preliminaries on the learning model, the classes of functions we will consider, and our main technical tools of boosting and the Fourier transform. In Section 4.3 we describe the Generalized Harmonic Sieve algorithm GHS which will be our main tool for learning unions of rectangles. In Section 4.4 we show that s-MAJORITY of r-PARITY of b-literals is efficiently learnable using GHS for suitable r, s; this concept class turns out to be quite useful for learning unions of rectangles. In Section 4.5 we improve over the results of Section 4.4 slightly if the number of terms is small, by adaptively selecting a small subset of $[b]$ in each dimension which is sufficient

for learning, and invoke GHS over the restricted domain. In Section 4.6 we explore the consequences of the results in Sections 4.4 and 4.5 for the ultimate goal of learning unions of rectangles. And finally in Section 4.7 we explore the implications of the results in the former sections towards quantum PAC learning.

4.2 Preliminaries

4.2.1 The learning model

In this chapter, we are interested in Boolean functions defined over the domain $[b]^n$, where $[b] = \{0, 1, \ldots, b-1\}$. We view Boolean functions as mappings into $\{-1, 1\}$ where -1 is associated with TRUE and 1 with FALSE.

In this chapter, a *concept class* $\mathfrak{C}$ is a collection of sets of Boolean functions $\{C_{n,b} \colon n > 0, b > 1\}$ such that if $f \in C_{n,b}$ then $f \colon [b]^n \to \{-1, 1\}$. Throughout this chapter we view both n and b as asymptotic parameters, and our goal is to exhibit algorithms that learn various classes $C_{n,b}$ in $\text{poly}(n, \log b)$ time. We now describe the uniform distribution membership query learning model that we will consider.

As defined earlier in Section 3.2.2, a *membership oracle* $\mathsf{MQ}(f)$ is an oracle which, when queried with input x, outputs the label $f(x)$ assigned by the target f to the input. Let $f \in C_{n,b}$ be an unknown member of the concept class and let $\mathcal{A}$ be a randomized learning algorithm which takes as input accuracy and confidence parameters ϵ, δ and can invoke $\mathsf{MQ}(f)$. We say that $\mathcal{A}$ *learns* $\mathfrak{C}$ *under the uniform distribution over* $[b]^n$ provided that given any $0 < \epsilon, \delta < 1$ and access to $\mathsf{MQ}(f)$, with probability at least $1 - \delta$ $\mathcal{A}$ outputs an ϵ-approximating hypothesis $h \colon [b]^n \to \{-1, 1\}$ (which need not belong to $\mathfrak{C}$) such that $\mathbf{Pr}_{x \in [b]^n}[f(x) = h(x)] \geq 1 - \epsilon$.

We are interested in computationally efficient learning algorithms. We say that $\mathcal{A}$ *learns* $\mathfrak{C}$ *efficiently* if for any target concept $f \in C_{n,b}$,

- $\mathcal{A}$ runs for at most $\text{poly}(n, \log b, 1/\epsilon, \log 1/\delta)$ steps;
- Any hypothesis h that $\mathcal{A}$ produces can be evaluated in at most $\text{poly}(n, \log b, 1/\epsilon, \log 1/\delta)$ time steps at any $x \in [b]^n$.

4.2.2 The functions we study

The reader might wonder which classes of Boolean valued functions over $[b]^n$ are interesting. In this chapter we study classes of functions that are defined in terms of "b-literals"; these include rectangles and unions of rectangles over $[b]^n$ as well as other richer classes. As described below, b-literals are a natural extension of Boolean literals to the domain $[b]^n$.

Definition 4.2.1. A function $\ell\colon [b] \to \{-1,1\}$ is a *basic b-literal* if for some $\sigma \in \{-1,1\}$ and some $\alpha \leq \beta$ with $\alpha, \beta \in [b]$ we have

$$\ell(x) = \begin{cases} \sigma & \text{if } \alpha \leq x \leq \beta \\ -\sigma & \text{otherwise.} \end{cases}$$

A function $\ell\colon [b] \to \{-1,1\}$ is a *b-literal* if there exists a basic b-literal ℓ' and some fixed $z \in [b]$, $\gcd(z,b) = 1$ such that for all $x \in [b]$ we have $\ell(x) = \ell'(xz \bmod b)$.

Basic b-literals are the most natural extension of Boolean literals to the domain $[b]^n$. General b-literals (not necessarily basic) were previously studied in [AGS03] and are also quite natural; for example, if b is odd then the *least significant bit* function $lsb(x)\colon [b] \to \{-1,1\}$ (defined by $lsb(x) = -1$ if and only if x is even) is a b-literal.

Definition 4.2.2. A function $f\colon [b]^n \to \{-1,1\}$ is a *k-rectangle* if it is an AND of k basic b-literals $\ell_1, \ldots, \ell_k$ over k distinct variables $x_{i_1}, \ldots, x_{i_k}$. If f is a k-rectangle for some k then we may simply say that f is a *rectangle*. A *union of s rectangles* $R_1, \ldots, R_s$ is a function of the form $f(x) = \text{OR}_{i=1}^{s} R_i(x)$.

The class of unions of s rectangles over $[b]^n$ is a natural generalization of the class of s-term DNF over $\{0,1\}^n$. Similarly MAJORITY of PARITY of basic b-literals generalizes the class of MAJORITY of PARITY of Boolean literals, a class which has been the subject of much research in learning theory and complexity theory (see e.g. [Jac97, Bru90, KS04]).

If G is a logic gate with potentially unbounded fan-in (e.g. MAJORITY, PARITY, AND, etc.) we write "s-G" to indicate that the fan-in of G is restricted to be at most s. Thus, for example, an "s-MAJORITY of r-PARITY of b-literals" is a MAJORITY of at most s functions $g_1, \ldots, g_s$, each

of which is a PARITY of at most r many b-literals. We will further assume that *any two b-literals which are inputs to the same gate depend on different variables.* This is a natural restriction to impose in light of our ultimate goal of learning unions of rectangles. Although our results hold without this assumption, it provides simplicity in the presentation.

4.2.3 Harmonic analysis of functions over $[b]^n$

We will make use of the Fourier expansion of complex valued functions over $[b]^n$.

Consider $f, g\colon [b]^n \to \mathbb{C}$ endowed with the inner product $\langle f, g\rangle = \mathbf{E}[f\overline{g}]$ and induced norm $\|f\| = \sqrt{\langle f, f\rangle}$. Let $\omega_b = e^{\frac{2\pi i}{b}}$ and for each $\alpha = (\alpha_1, \ldots, \alpha_n) \in [b]^n$, let $\chi_\alpha\colon [b]^n \to \mathbb{C}$ be defined as

$$\chi_\alpha(x_1, \ldots, x_n) = \omega_b^{\alpha_1 x_1 + \cdots + \alpha_n x_n}.$$

Let $\mathcal{B}$ denote the set of functions $\mathcal{B} = \{\chi_\alpha\colon \alpha \in [b]^n\}$. It is easy to verify the following properties:

- Elements in $\mathcal{B}$ are normal: for each $\alpha = (\alpha_1, \ldots, \alpha_n) \in [b]^n$, we have $\|\chi_\alpha\| = 1$.
- Elements in $\mathcal{B}$ are orthogonal: For $\alpha, \beta \in [b]^n$, we have $\langle \chi_\alpha, \chi_\beta\rangle = \begin{cases} 1 \text{ if } \alpha = \beta \\ 0 \text{ if } \alpha \neq \beta \end{cases}$.
- $\mathcal{B}$ constitutes an orthonormal basis for all functions $\{f\colon [b]^n \to \mathbb{C}\}$ considered as a vector space over $\mathbb{C}$. Thus every $f\colon [b]^n \to \mathbb{C}$ can be expressed uniquely as:

  $$f(x) = \sum_\alpha \hat{f}(\alpha)\chi_\alpha(x)$$

  which we refer to as the *Fourier expansion* or *Fourier transform* of f.

The values $\{\hat{f}(\alpha)\colon \alpha \in [b]^n\}$ are called the *Fourier coefficients* or the *Fourier spectrum* of f. As is well known, *Parseval's Identity* relates the values of the coefficients to the values of the function:

Lemma 4.2.3 (Parseval's Identity)**.** *For any $f\colon [b]^n \to \mathbb{C}$, we have $\sum_\alpha |\hat{f}(\alpha)|^2 = \mathbf{E}[|f|^2]$.*

We write $L_1(f)$ to denote $\sum_\alpha |\hat{f}(\alpha)|$.

4.2.4 Additional tools: weak hypotheses and boosting

Definition 4.2.4. Let $f\colon [b]^n \to \{-1,1\}$ and $\mathcal{D}$ be a probability distribution over $[b]^n$. A function $g\colon [b]^n \to \mathbb{R}$ is said to be a *weak hypothesis for f with advantage γ under $\mathcal{D}$* if $\mathbf{E}_{\mathcal{D}}[fg] \geq \gamma$.

The first *boosting* algorithm was described by Schapire [Sch90] in 1990; since then boosting has been intensively studied (see [FS99] for an overview). The basic idea is that by combining a sequence of weak hypotheses $h_1, h_2, \ldots$ (the i-th of which has advantage γ with respect to a carefully chosen distribution $\mathcal{D}_i$) it is possible to obtain a high accuracy final hypothesis h which satisfies $\mathbf{Pr}[h(x) = f(x)] \geq 1-\epsilon$. The following theorem gives a precise statement of the performance guarantees of a particular boosting algorithm, which we call Algorithm $\mathcal{B}$, due to Freund. Many similar statements are now known about a range of different boosting algorithms but this is sufficient for our purposes.

Theorem 4.2.5 (Boosting Algorithm [Fre95])**.** *Suppose that Algorithm $\mathcal{B}$ is given:*

- *$0 < \epsilon, \delta < 1$, and membership query access $\mathsf{MQ}(f)$ to $f\colon [b]^n \to \{-1,1\}$;*
- *access to an algorithm $\mathtt{WL}$ which has the following property: given a value δ' and access to $\mathsf{MQ}(f)$ and to $\mathsf{EX}(f,\mathcal{D})$ (the latter is an example oracle which generates random examples from $[b]^n$ drawn with respect to distribution $\mathcal{D}$), it constructs a weak hypothesis for f with advantage γ under $\mathcal{D}$ with probability at least $1-\delta'$ in time polynomial in n, $\log b$, $\log(1/\delta')$.*

Then Algorithm $\mathcal{B}$ behaves as follows:

- *It runs for $S = O(\log(1/\epsilon)/\gamma^2)$ stages and runs in total time polynomial in n, $\log b$, ϵ^{-1}, γ^{-1}, $\log(\delta^{-1})$.*
- *At each stage $1 \leq j \leq S$ it constructs a distribution $\mathcal{D}_j$ such that $L_\infty(\mathcal{D}_j) < \mathrm{poly}(\epsilon^{-1})/b^n$, and simulates $\mathsf{EX}(f,\mathcal{D}_j)$ for $\mathtt{WL}$ in stage j. Moreover, there is a "pseudo-distribution" $\tilde{\mathcal{D}}_j$ satisfying $\tilde{\mathcal{D}}_j(x) = c\mathcal{D}_j(x)$ for all x (where $c \in [1/2, 3/2]$ is some fixed value) such that $\tilde{\mathcal{D}}_j(x)$ can be computed in time polynomial in $n \log b$ for each $x \in [b]^n$.*
- *It outputs a final hypothesis $h = \mathrm{sgn}(h_1 + h_2 + \ldots + h_S)$ which ϵ-approximates f under the uniform distribution with probability $1-\delta$; here h_j is the output of $\mathtt{WL}$ at stage j invoked with simulated access to $\mathsf{EX}(f,\mathcal{D}_j)$.*

We will sometimes informally refer to distributions $\mathcal{D}$ which satisfy the bound $L_\infty(\mathcal{D}) < \frac{\mathrm{poly}(\epsilon^{-1})}{b^n}$ as *smooth* distributions.

In order to use boosting, it must be the case that there exists a suitable weak hypothesis with advantage γ. The "discriminator lemma" of Hajnal *et al.* [HMP$^+$93] (see also [Pis81]) can often be used to assert that the desired weak hypothesis exists:

Lemma 4.2.6 (The Discriminator Lemma [HMP$^+$93, Pis81])**.** *Let $\mathfrak{H}$ be a class of $\{1,-1\}$-valued functions over $[b]^n$ and let $f\colon [b]^n \to \{-1,1\}$ be expressible as $f = \textsc{Majority}(h_1,\ldots,h_s)$ where each $h_i \in \mathfrak{H}$ and $h_1(x)+\ldots+h_s(x) \neq 0$ for all x. Then for any distribution $\mathcal{D}$ over $[b]^n$ there is some h_i such that $|\mathbf{E}_{\mathcal{D}}[f h_i]| \geq 1/s$.*

4.3 The Generalized Harmonic Sieve algorithm

In this section our goal is to describe a variant of Jackson's Harmonic Sieve Algorithm and show that under suitable conditions it can efficiently learn certain functions $f\colon [b]^n \to \{-1,1\}$. As mentioned earlier, our aim is to attain $\mathrm{poly}(\log b)$ runtime dependence on b and consequently obtain efficient algorithms as described in Section 4.2.1. This goal precludes using Jackson's original Harmonic Sieve variant for $[b]^n$ since the runtime of his weak learner depends polynomially rather than polylogarithmically on b (see [Jac97, Lemma 15]).

As we describe below, this $\mathrm{poly}(\log b)$ runtime can be achieved by modifying the Harmonic Sieve over $[b]^n$ to use a weak learner due to Akavia *et al.* [AGS03] which is more efficient than Jackson's weak learner. We shall call the resulting algorithm "The Generalized Harmonic Sieve" algorithm, or GHS for short.

Recall that in the Harmonic Sieve over the Boolean domain $\{-1,1\}^n$, the weak hypotheses used are simply the Fourier basis elements over $\{-1,1\}^n$, which correspond to the Boolean-valued parity functions. For $[b]^n$, we will use the real component of the complex-valued Fourier basis elements $\{\chi_\alpha, \alpha \in [b]^n\}$ (as defined in Section 4.2.3) as our weak hypotheses.

The following theorem of Akavia *et al.* [AGS03, Theorem 5] will play a crucial role towards construction of the GHS algorithm.

Theorem 4.3.1 (See [AGS03])**.** *There is a learning algorithm that, given membership query access*

to $f\colon [b]^n \to \mathbb{C}$, $0 < \gamma$ and $0 < \delta < 1$, outputs a list L of indices such that with probability at least $1 - \delta$, we have $\{\alpha\colon |\hat{f}(\alpha)| > \gamma\} \subseteq L$ and $|\hat{f}(\beta)| \geq \frac{\gamma}{2}$ for every $\beta \in L$. The running time of the algorithm is polynomial in n, $\log b$, $\|f\|_\infty$, γ^{-1}, $\log(\delta^{-1})$.

Lemma 4.3.2 (Construction of the weak hypothesis)**.** *Given*

- *Membership query access $\mathsf{MQ}(f)$ to $f\colon [b]^n \to \{-1, 1\}$;*
- *A smooth distribution $\mathcal{D}$; more precisely, access to an algorithm computing $\tilde{\mathcal{D}}(x)$ in time polynomial in n, $\log b$ for each $x \in [b]^n$. Here $\tilde{\mathcal{D}}$ is a "pseudo-distribution" for $\mathcal{D}$ as in Theorem 4.2.5, i.e. there is a value $c \in [1/2, 3/2]$ such that $\tilde{\mathcal{D}}(x) = c\mathcal{D}(x)$ for all x.*
- *A value $0 < \gamma < 1/2$ such that there exists an element of the Fourier basis χ_τ satisfying $|\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\tau}]| > \gamma$,*

there is an algorithm that outputs a weak hypothesis for f with advantage $\gamma/4$ under $\mathcal{D}$ with probability $1 - \delta$ and runs in time polynomial in n, $\log b$, ϵ^{-1}, γ^{-1}, $\log(\delta^{-1})$.

Proof. Let $f_*(x) = b^n\tilde{\mathcal{D}}(x)f(x)$. Observe that

- Since $\mathcal{D}$ is smooth, $\|f_*\|_\infty < \mathrm{poly}(\epsilon^{-1})$.
- For any $\alpha \in [b]^n$, $\hat{f}_*(\alpha) = \mathbf{E}[f_*\overline{\chi_\alpha}] = \frac{1}{b^n}\sum_{x\in[b]^n} b^n\tilde{\mathcal{D}}(x)f(x)\overline{\chi_\alpha(x)} = \mathbf{E}_{\mathcal{D}}[cf\overline{\chi_\alpha}]$.

Therefore one can invoke the algorithm of Theorem 4.3.1 over $f_*(x)$ by simulating $\mathsf{MQ}(f_*)$ via $\mathsf{MQ}(f)$, each time with $\mathrm{poly}(n, \log b)$ time overhead, and obtain a list L of indices. Note that since we are guaranteed that there exists an index τ satisfying $|\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\tau}]| > \gamma$ implying $|\hat{f}_*(\tau)| \geq c\gamma$, we can invoke Theorem 4.3.1 in such a way that for any index β in its output, we know $|\hat{f}_*(\beta)| \geq c\gamma/2$.

It is easy to see that the algorithm runs in the desired time bound and outputs a nonempty list L. Let β be any element of L. Because $\hat{f}_*(\beta) = \mathbf{E}[b^n\tilde{\mathcal{D}}(x)f(x)\overline{\chi_\beta(x)}]$, one can approximate $\frac{\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\beta}]}{|\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\beta}]|} = \frac{\hat{f}_*(\beta)}{|\hat{f}_*(\beta)|} = e^{i\theta}$ using uniformly drawn random examples. Let $e^{i\theta'}$ be the approximation thus obtained.

By assumption we know that for random $x \in [b]^n$, the random variable $(b^n\tilde{\mathcal{D}}(x)f(x)\overline{\chi_\beta(x)})$ always takes a value whose magnitude is $O(\mathrm{poly}(\epsilon^{-1}))$ in absolute value. Using a straightforward

Chernoff bound argument, this implies that $|\theta - \theta'|$ can be made smaller than any constant using $\mathrm{poly}(n, \log b, \epsilon^{-1})$ time and random examples.

Now observe that we have

$$\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\beta}] = e^{i\theta}|\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\beta}]| \Rightarrow \mathbf{E}_{\mathcal{D}}[f\overline{e^{i\theta}\chi_\beta}] = |\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\beta}]| = c^{-1}|\hat{f}_*(\beta)| \geq \gamma/2.$$

Therefore for a sufficiently small value of $|\theta - \theta'|$, we have

$$\mathbf{E}_{\mathcal{D}}[f\Re\{\overline{e^{i\theta'}\chi_\beta}\}] = \Re\{\mathbf{E}_{\mathcal{D}}[f\overline{e^{i\theta'}\chi_\beta}]\} = \Re\{e^{i(\theta-\theta')} \underbrace{\mathbf{E}_{\mathcal{D}}[f\overline{e^{i\theta}\chi_\beta}]}_{\text{real valued and} \geq \gamma/2} \} \geq \gamma/4.$$

We conclude that $\Re\{\overline{e^{i\theta'}\chi_\beta}\}$ constitutes a weak hypothesis for f with advantage $\gamma/4$ under $\mathcal{D}$ with high probability. □

Rephrasing the statement of Lemma 4.3.2, now we know: As long as for any function f in the concept class it is guaranteed that under any smooth distribution $\mathcal{D}$ there is a Fourier basis element χ_β that has non-negligible correlation with f (i.e. $|\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\alpha}]| > \gamma$), then it is possible to efficiently identify and use such a Fourier basis element to construct a weak hypothesis.

Now one can invoke Algorithm $\mathcal{B}$ from Theorem 4.2.5 as in Jackson's original Harmonic Sieve: At stage j, we have a distribution $\mathcal{D}_j$ over $[b]^n$ for which $L_\infty(\mathcal{D}_j) < \mathrm{poly}(\epsilon^{-1})/b^n$. Thus one can pass the values of $\mathcal{D}_j$ to the algorithm in Lemma 4.3.2 and use this algorithm as `WL` in Algorithm $\mathcal{B}$ to obtain the weak hypothesis at each stage. Repeating this idea for every stage and combining the weak hypotheses generated for all the stages as described by Theorem 4.2.5, we have the GHS algorithm:

Corollary 4.3.3 (The Generalized Harmonic Sieve)**.** *Let $\mathfrak{C}$ be a concept class. Suppose that for any concept $f \in C_{n,b}$ and any distribution $\mathcal{D}$ over $[b]^n$ with $L_\infty(\mathcal{D}) < \mathrm{poly}(\epsilon^{-1})/b^n$ there exists a Fourier basis element χ_α such that $|\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\alpha}]| \geq \gamma$. Then $\mathfrak{C}$ can be learned in time* $\mathrm{poly}(n, \log b, \epsilon^{-1}, \gamma^{-1})$.

4.4 Learning MAJORITY of PARITY using GHS

In this section we identify classes of functions which can be learned efficiently using the GHS algorithm. Our main result is the following:

Theorem 4.4.1. *The concept class $\mathfrak{C}$ consisting of s-MAJORITY of r-PARITY of b-literals where $s = \mathrm{poly}(n \log b)$, $r = O(\frac{\log(n \log b)}{\log \log(n \log b)})$ is efficiently learnable using the GHS algorithm.*

To prove Theorem 4.4.1, we show that for any concept $f \in \mathfrak{C}$ and under any smooth distribution there must be some Fourier basis element which has high correlation with f; this is the essential step which lets us apply the Generalized Harmonic Sieve. We prove this in Section 4.4.2. In Section 4.4.3 we give an alternate argument which yields a Theorem 4.4.1 analogue but with a slightly different bound on r, namely $r = O(\frac{\log(n \log b)}{\log \log b})$.

4.4.1 Setting the stage

For ease of notation we will write $abs(\alpha)$ to denote $\min\{\alpha, b - \alpha\}$. We will use the following simple lemma from [AGS03]:

Lemma 4.4.2 (See [AGS03])**.** *For all $0 \leq \ell \leq b$, we have $|\sum_{y=0}^{\ell-1} \omega_b^{\alpha y}| < b/abs(\alpha)$.*

Corollary 4.4.3. *Let $f : [b] \to \{-1, 1\}$ be a basic b-literal, then*

- *If $\alpha = 0$, $|\hat{f}(\alpha)| < 1$.*
- *If $\alpha \neq 0$, $|\hat{f}(\alpha)| < \frac{2}{abs(\alpha)}$.*

Proof. The first inequality follows immediately from Lemma 4.2.3 (Parseval's Identity) because f is $\{1, -1\}$-valued. For the latter, note that

$$|\hat{f}(\alpha)| = |\mathbf{E}[f\overline{\chi_\alpha}]| = \frac{1}{b}\left|\sum_{x \in f^{-1}(1)} \chi_\alpha(x) - \sum_{x \in f^{-1}(-1)} \chi_\alpha(x)\right| \leq \frac{1}{b}\left|\sum_{x \in f^{-1}(1)} \chi_\alpha(x)\right| + \frac{1}{b}\left|\sum_{x \in f^{-1}(-1)} \chi_\alpha(x)\right|$$

where the inequality is simply the triangle inequality. It is easy to see that each of the sums on the R.H.S. above equals $\frac{1}{b}\,|\omega_b^{\alpha c}|\,|\sum_{y=0}^{\ell-1} \omega_b^{\alpha y}| = \frac{1}{b}|\sum_{y=0}^{\ell-1} \omega_b^{\alpha y}|$ for some suitable c and $\ell \leq b$, and hence Lemma 4.4.2 gives the desired result. □

The following easy lemma is useful for relating the Fourier transform of a b-literal to the corresponding basic b-literal:

Lemma 4.4.4. *For $f, g\colon [b] \to \mathbb{C}$ such that $g(x) = f(xz)$ where $\gcd(z, b) = 1$, we have $\hat{g}(\alpha) = \hat{f}(\alpha z^{-1})$.*

Proof.

$$\hat{g}(\alpha) = \mathbf{E}_x[g(x)\overline{\chi_\alpha(x)}] = \mathbf{E}_x[f(xz)\overline{\chi_\alpha(x)}] = \mathbf{E}_{xz^{-1}}[f(x)\overline{\chi_\alpha(xz^{-1})}]$$
$$= \mathbf{E}_{xz^{-1}}[f(x)\overline{\chi_{\alpha z^{-1}}(x)}] = \mathbf{E}_x[f(x)\overline{\chi_{\alpha z^{-1}}(x)}] = \hat{f}(\alpha z^{-1}).$$

□

A natural way to approximate a b-literal is by truncating its Fourier representation. We make the following definition:

Definition 4.4.5. Let k be a positive integer. For $f\colon [b] \to \{-1, 1\}$ a basic b-literal, the *k-restriction of f* is $\tilde{f}\colon [b] \to \mathbb{C}$,

$$\tilde{f}(x) = \sum_{abs(\alpha) \le k} \hat{f}(\alpha)\chi_\alpha(x).$$

More generally, for $f\colon [b] \to \{-1, 1\}$ a b-literal (so $f(x) = f'(xz)$ where f' is a basic b-literal) the *k-restriction of f* is $\tilde{f}\colon [b] \to \mathbb{C}$,

$$\tilde{f}(x) = \sum_{abs(\alpha z^{-1}) \le k} \hat{f}(\alpha)\chi_\alpha(x) = \sum_{abs(\beta) \le k} \widehat{f'}(\beta)\chi_\beta(x).$$

4.4.2 There exist highly correlated Fourier basis elements for functions in $\mathfrak{C}$ under smooth distributions

In this section we show that given any $f \in \mathfrak{C}$ and any smooth distribution $\mathcal{D}$, some Fourier basis element must have high correlation with f. We begin by bounding the error of the k-restriction of a basic b-literal:

Lemma 4.4.6. *For $f\colon [b] \to \{-1,1\}$ a b-literal and $\tilde{f}$ the k-restriction of f, we have $\mathbf{E}[|f-\tilde{f}|^2] = O(1/k)$.*

Proof. Without loss of generality assume f to be a basic b-literal. By an immediate application of Lemma 4.2.3 (Parseval's Identity) we obtain:

$$\mathbf{E}[|f-\tilde{f}|^2] = \sum_{abs(\alpha)>k} |\hat{f}(\alpha)|^2 \underbrace{=}_{\text{by Corollary 4.4.3}} \sum_{\alpha>k} O(1)/\alpha^2 = O(1/k).$$

□

Now suppose that f is an r-PARITY of b-literals $f_1,\ldots,f_r$. Since PARITY corresponds to multiplication over the domain $\{-1,1\}$, this means that $f = \prod_{i=1}^r f_i$. It is natural to approximate f by the product of the k-restrictions $\prod_{i=1}^r \tilde{f}_i$. The following lemma bounds the error of this approximation:

Lemma 4.4.7. *For $i = 1,\ldots,r$, let $f_i\colon [b] \to \{-1,1\}$ be a b-literal and let $\tilde{f}_i$ be its k-restriction. Then*

$$\mathbf{E}[|f_1(x_1)f_2(x_2)\ldots f_r(x_r) - \tilde{f}_1(x_1)\tilde{f}_2(x_2)\ldots\tilde{f}_r(x_r)|] < O(1)\cdot(e^{\frac{O(1)r}{\sqrt{k}}}-1).$$

Proof. First note that by the non-negativity of variance and Lemma 4.4.6, we have that for each $i = 1,\ldots,r$:

$$\mathbf{E}_{x_i}[|f_i(x_i)-\tilde{f}_i(x_i)|] \le \sqrt{\mathbf{E}_{x_i}[|f_i(x_i)-\tilde{f}_i(x_i)|^2]} = O(1/\sqrt{k}).$$

Therefore we also have for each $i = 1,\ldots,r$:

$$\mathbf{E}_{x_i}[|\tilde{f}_i(x_i)|] < \underbrace{\mathbf{E}_{x_i}[|\tilde{f}_i(x_i)-f_i(x_i)|]}_{<O(\frac{1}{\sqrt{k}})} + \underbrace{\mathbf{E}_{x_i}[|f_i(x_i)|]}_{=1} = 1 + \frac{O(1)}{\sqrt{k}}.$$

For any $(x_1, \ldots, x_r)$ we can bound the difference in the lemma as follows:

$$
\begin{aligned}
&|f_1(x_1)\ldots f_r(x_r) - \tilde{f}_1(x_1)\ldots\tilde{f}_r(x_r)| \\
&\leq |f_1(x_1)\ldots f_r(x_r) - f_1(x_1)\ldots f_{r-1}(x_{r-1})\tilde{f}_r(x_r)| \\
&+ |f_1(x_1)\ldots f_{r-1}(x_{r-1})\tilde{f}_r(x_r) - \tilde{f}_1(x_1)\ldots\tilde{f}_r(x_r)| \\
&\leq \underbrace{|f_1(x_1)\ldots f_{r-1}(x_{r-1})|}_{=1}|f_r(x_r) - \tilde{f}_r(x_r)| + |\tilde{f}_r(x_r)||f_1(x_1)\ldots f_{r-1}(x_{r-1}) - \tilde{f}_1(x_1)\ldots\tilde{f}_{r-1}(x_{r-1})|
\end{aligned}
$$

Therefore we can express the expectation in question as follows:

$$
\begin{aligned}
&\mathbf{E}_{(x_1,\ldots,x_r)\in[b]^r}[|f_1(x_1)f_2(x_2)\ldots f_r(x_r) - \tilde{f}_1(x_1)\tilde{f}_2(x_2)\ldots\tilde{f}_r(x_r)|] \\
&\leq \underbrace{\mathop{\mathbf{E}}_{x_r}[|f_r(x_r) - \tilde{f}_r(x_r)|]}_{=O(\frac{1}{\sqrt{k}})} + \underbrace{\mathop{\mathbf{E}}_{x_r}[|\tilde{f}_r(x_r)|]}_{=1+\frac{O(1)}{\sqrt{k}}} \cdot \mathbf{E}_{(x_1,\ldots,x_{r-1})}[|f_1(x_1)\ldots f_{r-1}(x_{r-1}) - \tilde{f}_1(x_1)\ldots\tilde{f}_{r-1}(x_{r-1})|]
\end{aligned}
$$

and repeat this argument successively until the base case $\mathbf{E}_{x_1}[|f_1(x_1) - \tilde{f}_1(x_1)|] \leq O(\frac{1}{\sqrt{k}})$ is reached. Thus for some $K = O(1)$, $1 < L = 1 + \frac{O(1)}{\sqrt{k}}$, we have

$$
\begin{aligned}
\mathbf{E}[|f_1(x_1)\ldots f_r(x_r) - \tilde{f}_1(x_1)\ldots\tilde{f}_r(x_r)|] &\leq \frac{K\sum_{i=0}^{r-1} L^i}{\sqrt{k}} = \frac{K(L^r-1)}{(L-1)\sqrt{k}} \\
&\leq O(1)\cdot((1+\frac{O(1)}{\sqrt{k}})^r - 1) < O(1)\cdot(e^{\frac{O(1)r}{\sqrt{k}}} - 1). \quad \square
\end{aligned}
$$

from which the lemma follows. $\square$

Now we are ready for the main theorem asserting the existence (under suitable conditions) of a highly correlated Fourier basis element. The basic approach of the following proof is reminiscent of the main technical lemma from [JKS02].

Theorem 4.4.8. *Let τ be a parameter to be specified later and $\mathfrak{C}$ be the concept class consisting of s-*MAJORITY *of r-*PARITY *of b-literals where $s = \mathrm{poly}(\tau)$ and $r = O(\frac{\log(\tau)}{\log\log(\tau)})$. Then for any $f \in C_{n,b}$ and any distribution $\mathcal{D}$ over $[b]^n$ with $L_\infty(\mathcal{D}) = \mathrm{poly}(\tau)/b^n$, there exists a Fourier basis element χ_α such that*

$$|\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\alpha}]| > \Omega(1/\mathrm{poly}(\tau)).$$

Proof. Assume f is a MAJORITY of $h_1, \ldots, h_s$ each of which is a r-PARITY of b-literals. Then

Lemma 4.2.6 implies that there exists h_i such that $|\mathbf{E}_{\mathcal{D}}[fh_i]| \geq 1/s$. Let h_i be PARITY of the b-literals $\ell_1, \ldots, \ell_r$.

Since s and $b^n \cdot L_\infty(\mathcal{D})$ are both at most $\mathrm{poly}(\tau)$ and $r = O(\frac{\log(\tau)}{\log\log(\tau)})$, Lemma 4.4.7 implies that there are absolute constants C_1, C_2 such that if we consider the k-restrictions $\tilde{\ell}_1, \ldots, \tilde{\ell}_r$ of $\ell_1, \ldots, \ell_r$ for $k = C_1 \cdot \tau^{C_2}$, we will have

$$\mathbf{E}[|h_i - \prod_{j=1}^{r} \tilde{\ell}_j|] \leq 1/(2sb^n L_\infty(\mathcal{D}))$$

where the expectation on the left hand side is with respect to the uniform distribution over $[b]^n$. This in turn implies that

$$\mathbf{E}_{\mathcal{D}}[|h_i - \prod_{j=1}^{r} \tilde{\ell}_j|] \leq 1/2s.$$

Let us write h' to denote $\prod_{j=1}^{r} \tilde{\ell}_j$. We then have

$$\begin{aligned}
|\mathbf{E}_{\mathcal{D}}[f\overline{h'}]| &\geq |\mathbf{E}_{\mathcal{D}}[f\overline{h_i}]| - |\mathbf{E}_{\mathcal{D}}[f\overline{(h_i - h')}]| \\
&\geq |\mathbf{E}_{\mathcal{D}}[f\overline{h_i}]| - \mathbf{E}_{\mathcal{D}}[|f\overline{(h_i - h')}|] = |\mathbf{E}_{\mathcal{D}}[fh_i]| - \mathbf{E}_{\mathcal{D}}[|h_i - h'|] \\
&\geq 1/s - 1/2s = 1/2s.
\end{aligned}$$

Now observe that we additionally have

$$|\mathbf{E}_{\mathcal{D}}[f\overline{h'}]| = |\mathbf{E}_{\mathcal{D}}[f\overline{\sum_{\alpha} \hat{h'}(\alpha)\chi_\alpha}]| = |\sum_{\alpha} \overline{\hat{h'}(\alpha)} \mathbf{E}_{\mathcal{D}}[f\overline{\chi_\alpha}]| \leq L_1(h') \max_{\alpha} |\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\alpha}]|$$

Moreover, for each $j = 1, \ldots, r$ we have the following (where we write ℓ'_j to denote the basic b-literal associated with the b-literal ℓ_j):

$$L_1(\tilde{\ell}_j) = \sum_{abs(\alpha) \leq k} |\widehat{\ell'}_j(\alpha)| \underbrace{=}_{\text{by Corollary 4.4.3}} 1 + \sum_{\alpha=1}^{k} O(1)/\alpha = O(\log k).$$

Therefore, for some absolute constant $c > 0$ we have $L_1(h') \leq \prod_{j=1}^{r} L_1(\tilde{\ell}_j) \leq (c \log k)^r$, where the first inequality holds since the L_1 norm of a product is at most the product of the L_1 norms.

Combining inequalities, we obtain

$$\max_{\alpha} |\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\alpha}]| \geq 1/(2s(c\log k)^r) = \Omega(1/\mathrm{poly}(\tau))$$

which is the desired result. □

Since we are interested in algorithms with runtime $\mathrm{poly}(n, \log b, \epsilon^{-1})$, setting $\tau = n\epsilon^{-1}\log b$ in Theorem 4.4.8 and combining its result with Corollary 4.3.3, gives rise to Theorem 4.4.1.

4.4.3 The second approach

A different analysis, similar to that which Jackson uses in the proof of [Jac97, Fact 14], gives us an alternate bound to Theorem 4.4.8:

Lemma 4.4.9. *Let $\mathfrak{C}$ be the concept class consisting of s-*MAJORITY *of r-*PARITY *of b-literals. Then for any $f \in C_{n,b}$ and any distribution $\mathcal{D}$ over $[b]^n$, there exists a Fourier basis element χ_α such that*

$$|\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\alpha}]| = \Omega(1/s(\log b)^r).$$

Proof. Assume f is a MAJORITY of $h_1, \ldots, h_s$ each of which is a r-PARITY of b-literals. Then Lemma 4.2.6 implies that there exists h_i such that $|\mathbf{E}_{\mathcal{D}}[fh_i]| \geq 1/s$. Let h_i be PARITY of the b-literals $\ell_1, \ldots, \ell_r$. Now observe:

$$\begin{aligned} 1/s \leq |\mathbf{E}_{\mathcal{D}}[fh_i]| = |\mathbf{E}_{\mathcal{D}}[f\overline{h_i}]| &= |\mathbf{E}_{\mathcal{D}}[f\overline{\sum_{\alpha} \hat{h}_i(\alpha)\chi_\alpha}]| \\ &= |\sum_{\alpha} \overline{\hat{h}_i(\alpha)}\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\alpha}]| \\ &\leq L_1(h_i)\max_{\alpha} |\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\alpha}]| \end{aligned}$$

Also note that for $j = 1, \ldots, r$ we have the following (where as before we write ℓ'_j to denote the basic b-literal associated with the b-literal ℓ_j):

$$L_1(\ell_j) \underbrace{=}_{\text{by Lemma 4.4.4}} \sum_{\alpha} |\hat{\ell'_j}(\alpha)| \underbrace{=}_{\text{by Corollary 4.4.3}} 1 + \sum_{\alpha=1}^{b-1} O(1)/\alpha = O(\log b).$$

Therefore for some constant $c > 0$ we have $L_1(h_i) \leq \prod_{j=1}^{r} L_1(\ell_j) = O((\log b)^r)$, from which we obtain:

$$\max_{\alpha} |\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\alpha}]| = \Omega(1/s(\log b)^r)$$

□

Combining this result with that of Corollary 4.3.3 we obtain the following result:

Theorem 4.4.10. *The concept class $\mathfrak{C}$ consisting of s-*MAJORITY *of r-*PARITY *of b-literals can be learned in time* $\mathrm{poly}(s, n, (\log b)^r)$ *using the* GHS *algorithm.*

As an immediate corollary we obtain the following close analogue of Theorem 4.4.1:

Theorem 4.4.11. *The concept class $\mathfrak{C}$ consisting of s-*MAJORITY *of r-*PARITY *of b-literals where $s = \mathrm{poly}(n \log b)$, $r = O(\frac{\log(n \log b)}{\log \log b})$ is efficiently learnable using the* GHS *algorithm.*

4.5 Locating sensitive elements and learning with GHS on a restricted grid

In this section we consider an extension of the GHS algorithm which lets us achieve slightly better bounds when we are dealing only with basic b-literals. Following an idea from [BK98], the new algorithm works by identifying a subset of "sensitive" elements from $[b]$ for each of the n dimensions. Although we will be invoking Theorem 4.4.11 inside our algorithm, an important feature of the developed algorithm is that it could be built on top of any other uniform distribution learning algorithm for rectangles in order to reduce its complexity.

Definition 4.5.1 (See [BK98])**.** A value $\sigma \in [b]$ is called *i-sensitive* with respect to $f\colon [b]^n \to \{-1, 1\}$ if there exist values $c_1, c_2, \ldots, c_{i-1}, c_{i+1}, \ldots, c_n \in [b]$ such that

$$f(c_1, \ldots, c_{i-1}, \sigma - 1, c_{i+1}, \ldots, c_n) \neq f(c_1, \ldots, c_{i-1}, \sigma, c_{i+1}, \ldots, c_n).$$

A value σ is called *sensitive* with respect to f if σ is i-sensitive for some i. If there is no i-sensitive value with respect to f, we say index i is *trivial*.

The main idea is to run GHS over a restricted subset of the original domain $[b]^n$, which is the grid formed by the sensitive values and a few more additional values, and therefore lower the algorithm's complexity.

Definition 4.5.2. A *grid in* $[b]^n$ is a set $\mathcal{S} = L_1 \times L_2 \times \cdots \times L_n$ with $0 \in L_i \subseteq [b]$ for each i. We refer to the elements of $\mathcal{S}$ as *corners*. The *region covered by a corner* $(x_1, \ldots, x_n) \in \mathcal{S}$ is defined to be the set $\{(y_1, \ldots, y_n) \in [b]^n : \forall i, x_i \leq y_i < \lceil x_i \rceil\}$ where $\lceil x_i \rceil$ denotes the smallest value in L_i which is larger than x_i (by convention $\lceil x_i \rceil := b$ if no such value exists). The *area covered by the corner* $(x_1, \ldots, x_n) \in \mathcal{S}$ is therefore defined to be $\prod_{i=1}^{n} (\lceil x_i \rceil - x_i)$. A *refinement* of $\mathcal{S}$ is a grid in $[b]^n$ of the form $L'_1 \times L'_2 \times \cdots \times L'_n$ where each $L_i \subseteq L'_i$.

Lemma 4.5.3. *Let $\mathcal{S}$ be a grid $L_1 \times L_2 \times \cdots \times L_n$ in $[b]^n$ such that each $|L_i| \leq \ell$. Let $\mathcal{I}_{\mathcal{S}}$ denote the set of indices for which $L_i \neq \{0\}$. If $|\mathcal{I}_{\mathcal{S}}| \leq \kappa$, then $\mathcal{S}$ admits a refinement $\mathcal{S}' = L'_1 \times L'_2 \times \cdots \times L'_n$ such that*

1. *All of the sets L'_i which contain more than one element have the same number of elements: $\mathbf{L_{max}}$, which is at most $\ell + C\kappa\ell$, where $C = \frac{b}{\kappa\ell} \cdot \frac{1}{\lfloor b/4\kappa\ell \rfloor} \geq 4$.*
2. *Given a list of the sets $L_1, \ldots, L_n$ as input, a list of the sets $L'_1, \ldots, L'_n$ can be generated by an algorithm with a running time of $O(n\kappa\ell \log b)$.*
3. *$L'_i = \{0\}$ whenever $L_i = \{0\}$.*
4. *Any ϵ fraction of the corners in $\mathcal{S}'$ cover a combined area of at most $2\epsilon b^n$.*

Proof. Consider Algorithm 7 which, given $\mathcal{S} = L_1 \times L_2 \times \cdots \times L_n$, generates $\mathcal{S}'$.

The purpose of the code between lines 19–23 is to make every $L'_i \neq \{0\}$ contain equal number of elements. Therefore the algorithm keeps track of the number of elements in the largest L'_i in a variable called $\mathbf{L_{max}}$ and eventually adds more (arbitrary) elements to those $L'_i \neq \{0\}$ which have fewer elements.

It is clear that the algorithm satisfies Property 3 above.

Now consider the state of Algorithm 7 at line 18. Let i be such that $|L'_i| = \mathbf{L_{max}}$. Clearly L'_i includes the elements in L_i which are at most ℓ many. Moreover every new element added to L'_i in the loop spanning lines 9-12 covers a section of $[b]$ of width τ, and thus $b/\tau = C\kappa\ell$ elements can

Algorithm 7 Computing a refinement of the grid $\mathbb{S}$ with the desired properties.

$\mathbf{L_{max}} \leftarrow 0$.
for all $1 \leq i \leq n$ **do**
  if $L_i = \{0\}$ **then**
    $L'_i \leftarrow \{0\}$.
  else
    Consider $L_i = \{x_0^i, x_1^i \ldots, x_{\ell-1}^i\}$, where $x_0^i < x_1^i < \cdots < x_{\ell-1}^i$ (Also let $x_\ell^i = b$).
    $L'_i \leftarrow L_i$.
    $\tau \leftarrow \lfloor b/4\kappa\ell \rfloor$.
    for all $r = 0, \ldots, \ell - 1$ **do**
      if $|x_{r+1}^i - x_r^i| > \tau$ **then**
        $L'_i \leftarrow L'_i \cup \{x_r^i + \tau, x_r^i + 2\tau, \ldots\}$ (up to and including the largest $x_r^i + j \cdot \tau$ which is less than x_{r+1}^i)
      end if
    end for
    if $|L'_i| > \mathbf{L_{max}}$ **then**
      $\mathbf{L_{max}} \leftarrow |L'_i|$.
    end if
  end if
end for
for all $1 \leq i \leq n$ with $|L'_i| > 1$ **do**
  while $(|L'_i| < \mathbf{L_{max}})$ **do**
    $L'_i \leftarrow L'_i \cup \{$an arbitrary element from $[b]\}$.
  end while
end for
$\mathbb{S}' \leftarrow L'_1 \times L'_2 \times \cdots \times L'_n$.

be added. Thus $\mathbf{L_{max}} \leq \ell + C\kappa\ell$. At the end of the algorithm every L'_i contains either 1 element (which is $\{0\}$) or $\mathbf{L_{max}}$ elements. This gives us Property 1. Note that $C \geq 4$ by construction.

It is easy to verify that it satisfies Property 2 as well (the $\log b$ factor in the runtime is present because the algorithm works with $(\log b)$-bit integers).

Property 1 and the bound $|\mathcal{I}_{\mathbb{S}}| \leq \kappa$ together give that the number of corners in $\mathbb{S}$ is at most $(\ell + C\kappa\ell)^\kappa$. It is easy to see from the algorithm that the area covered by each corner in $\mathbb{S}'$ is at most $\frac{b^n}{(C\kappa\ell)^\kappa}$ (again using the bound on $|\mathcal{I}_{\mathbb{S}}|$). Therefore any ϵ fraction of the corners in $\mathbb{S}'$ cover an area of at most:

$$\epsilon(\ell + C\kappa\ell)^\kappa \times \frac{b^n}{(C\kappa\ell)^\kappa} = \epsilon(1 + \frac{1}{C\kappa})^\kappa \times b^n \underbrace{<}_{C \geq 4} e^{1/3}\epsilon b^n < 2\epsilon b^n.$$

This gives Property 4. □

The following lemma is easy and useful; similar statements are given in [BK98]. Note that the lemma critically relies on the b-literals being basic.

Lemma 4.5.4. *Let $f\colon [b]^n \to \{-1,1\}$ be expressed as an s-*MAJORITY *of* PARITY *of basic b-literals. Then for each index $1 \leq i \leq n$, there are at most $2s$ i-sensitive values with respect to f.*

Proof. A literal ℓ on variable x_i induces two i-sensitive values. The lemma follows directly from our assumption (see the end of Section 4.2.2) that for each variable x_i, each of the s PARITY gates has no more than one incoming literal which depends on x_i. □

Algorithm 8 is our extension of the GHS algorithm. It essentially works by repeatedly running GHS on the target function f but restricted to a small (relative to $[b]^n$) grid. To upper bound the number of steps in each of these invocations we will be referring to the result of Theorem 4.4.11. After each execution of GHS, the hypothesis defined over the grid is extended to $[b]^n$ in a natural way and is tested for ϵ-accuracy. If h is not ϵ-accurate, then a point where h is incorrect is used to identify a new sensitive value and this value is used to refine the grid for the next iteration. The bound on the number of sensitive values from Lemma 4.5.4 lets us bound the number of iterations. Our theorem about Algorithm 8's performance is the following:

Theorem 4.5.5. *Let concept class $\mathfrak{C}$ consist of s-*MAJORITY *of r-*PARITY *of basic b-literals such that $s = \mathrm{poly}(n \log b)$ and each $f \in C_{n,b}$ has at most $\kappa(n,b)$ non-trivial indices and at most $\ell(n,b)$ i-sensitive values for each $i = 1,\ldots,n$. Then $\mathfrak{C}$ is efficiently learnable if $r = O(\frac{\log(n \log b)}{\log\log \kappa\ell})$.*

Proof. We assume $b = \omega(\kappa\ell)$ without loss of generality. Otherwise one immediately obtains the result with a direct application of GHS through Theorem 4.4.11.

We clearly have $\kappa \leq n$ and $\ell \leq 2s$. By Lemma 4.5.4 there are at most $\kappa\ell = O(ns)$ sensitive values. We will show that the algorithm finds a new sensitive value at each iteration and terminates before all sensitive values are found. Therefore the number of iterations will be upper bounded by $O(ns)$. We will also show that each iteration runs in $\mathrm{poly}(n, \log b, \epsilon^{-1})$ steps. This will establish the desired result.

Let's first establish that step 6 takes at most $\mathrm{poly}(n, \log b, \epsilon^{-1})$ steps. To observe this it is sufficient to combine the following facts:

Algorithm 8 An improved algorithm for learning MAJORITY of PARITY of basic b-literals.

1: $L_1 \leftarrow \{0\}, L_2 \leftarrow \{0\}, \ldots, L_n \leftarrow \{0\}$.
2: **loop**
3: $\mathcal{S} \leftarrow L_1 \times L_2 \times \cdots \times L_n$.
4: $\mathcal{S}' \leftarrow$ the output of refinement algorithm with input $\mathcal{S}$.
5: One can express $\mathcal{S}' = L'_1 \times L'_2 \times \cdots \times L'_n$. If $L_i \neq \{0\}$ then $L'_i = \{x^i_0, x^i_1 \ldots, x^i_{(\mathbf{L_{max}}-1)}\}$. Let $x^i_0 < x^i_1 < \cdots < x^i_{t-1}$ and let $\tau_i : \mathbb{Z}_{\mathbf{L_{max}}} \to L'_i$ be the translation function such that $\tau_i(j) = x^i_j$. If $L_i = L'_i = \{0\}$ then τ_i is the function simply mapping 0 to 0.
6: Invoke GHS over $f|_{\mathcal{S}'}$ with accuracy $\epsilon/8$. This is done by simulating $\mathsf{MQ}(f|_{\mathcal{S}'}(x_1, \ldots, x_n))$ with $\mathsf{MQ}(f(\tau_1(x_1), \tau_2(x_2), \ldots, \tau_n(x_n)))$. Let the output of the algorithm be g.
7: Let h be a hypothesis function over $[b]^n$ such that $h(x_1, \ldots, x_n) = g(\tau_1^{-1}(\lfloor x_1 \rfloor), \ldots, \tau_n^{-1}(\lfloor x_n \rfloor))$ ($\lfloor x_i \rfloor$ denotes largest value in L'_i less than or equal to x_i).
8: **if** h ϵ-approximates f **then**
9: Output h and terminate.
10: **end if**
11: Perform random membership queries until an element $(x_1, \ldots, x_n) \in [b]^n$ is found such that $f(\lfloor x_1 \rfloor, \ldots, \lfloor x_n \rfloor) \neq f(x_1, \ldots, x_n)$.
12: Find an index $1 \leq i \leq n$ such that

$$f(\lfloor x_1 \rfloor, \ldots, \lfloor x_{i-1} \rfloor, x_i, \ldots, x_n) \neq f(\lfloor x_1 \rfloor, \ldots, \lfloor x_{i-1} \rfloor, \lfloor x_i \rfloor, x_{i+1}, \ldots, x_n)$$

This requires $O(\log n)$ membership queries using binary search.
13: Find a value σ such that $\lfloor x_i \rfloor + 1 \leq \sigma \leq x_i$ and

$$f(\lfloor x_1 \rfloor, \ldots, \lfloor x_{i-1} \rfloor, \sigma - 1, x_{i+1}, \ldots, x_n) \neq f(\lfloor x_1 \rfloor, \ldots, \lfloor x_{i-1} \rfloor, \sigma, x_{i+1}, \ldots, x_n)$$

This requires $O(\log b)$ membership queries using binary search.
14: $L_i \leftarrow L_i \cup \{\sigma\}$.
15: **end loop**

- Due to the construction of Algorithm 7 for every non-trivial index i of f, L'_i has fixed cardinality $= \mathbf{L_{max}}$. Therefore GHS could be invoked over the restriction of f onto the grid, $f|_{\mathcal{S}'}$, without any trouble.

- If f is s-MAJORITY of r-PARITY of basic b-literals, then the function obtained by restricting it onto the grid: $f|_{\mathcal{S}'}$ could be expressed as t-MAJORITY of u-PARITY of basic L-literals where $t \leq s$, $u \leq r$ and $L \leq O(\kappa\ell)$ (due to the 1^{st} property of the refinement).

- Due to Theorem 4.4.11, running GHS over a grid with alphabet size $O(\kappa\ell)$ in each non-trivial index takes $\mathrm{poly}(n, \log b, \epsilon^{-1})$ time if the dimension of the rectangles are $r = O(\frac{\log(n \log b)}{\log \log \kappa\ell})$. The key idea here is that running GHS over this $\kappa\ell$-size alphabet lets us replace the "b" in

Theorem 4.4.11 with "$\kappa\ell$".

To check whether if h ϵ-approximates f at step 8, we may draw $O(1/\epsilon) \cdot \log(1/\delta)$ uniform random examples and use the membership oracle to empirically estimate h's accuracy on these examples. Standard bounds on sampling show that if the true error rate of h is less than (say) $\epsilon/2$, then the empirical error rate on such a sample will be less than ϵ with probability $1 - \delta$. Observe that if all the sensitive values are recovered by the algorithm, h will ϵ-approximate f with high probability. Indeed, since g $(\epsilon/8)$-approximates $f|_{\mathcal{S}'}$, Property 4 of the refinement guarantees that misclassifying the function at $\epsilon/8$ fraction of the corners could at most incur an overall error of $2\epsilon/8 = \epsilon/4$. This is because when all the sensitive elements are recovered, for every corner in $\mathcal{S}'$, h either agrees with f or disagrees with f in the entire region covered by that corner. Thus h will be an $\epsilon/4$ approximator to f with high probability. This establishes that the algorithm must terminate within $O(ns)$ iterations of the outer loop.

Locating another sensitive value occurs at steps 11, 12 and 13. Note that h is not an ϵ-approximator to f because the algorithm moved beyond step 8. Even if we were to correct all the mistakes in g this would alter at most $\epsilon/8$ fraction of the corners in the grid $\mathcal{S}'$ and therefore $\epsilon/4$ fraction of the values in h – again due to the 4^{th} property of the refinement and the way h is generated. Therefore for at least $3\epsilon/4$ fraction of the domain we ought to have $f(\lfloor x_1 \rfloor, \ldots, \lfloor x_n \rfloor) \neq f(x_1, \ldots, x_n)$ where $\lfloor x_i \rfloor$ denotes largest value in L'_i less than or equal to x_i. Thus the algorithm requires at most $O(1/\epsilon)$ random queries to find such an input in step 11.

Thus we have observed that steps 6, 8, 11, 12, 13 take at most $\mathrm{poly}(n, \log b, \epsilon^{-1})$ steps. Therefore each iteration of Algorithm 8 runs in $\mathrm{poly}(n, \log b, \epsilon^{-1})$ steps as claimed.

We note that we have been somewhat cavalier in our treatment of the failure probabilities for various events (such as the possibility of getting an inaccurate estimate of h's error rate in step 9, or not finding a suitable element $(x_1, \ldots, x_n)$ soon enough in step 11). A standard analysis shows that all these failure probabilities can be made suitably small so that the overall failure probability is at most δ within the claimed runtime. □

4.6 Applications to learning unions of rectangles

In this section we apply the results we have obtained in Sections 4.4 and 4.5 to obtain results on learning unions of rectangles and related classes.

4.6.1 Learning majorities and unions of many low-dimensional rectangles

The following lemma will let us apply our algorithm for learning MAJORITY of PARITY of b-literals to learn MAJORITY of AND of b-literals:

Lemma 4.6.1. *Let $f\colon \{-1,1\}^n \to \{-1,1\}$ be expressible as an s-MAJORITY of r-AND of Boolean literals. Then f is also expressible as a $O(ns^2)$-MAJORITY of r-PARITY of Boolean literals.*

We note that Krause and Pudlák gave a related but slightly weaker bound in [KP98]; they used a probabilistic argument to show that any s-MAJORITY of AND of Boolean literals can be expressed as an $O(n^2s^4)$-MAJORITY of PARITY. Our boosting-based argument below closely follows that of [Jac97, Corollary 13].

Proof of Lemma 4.6.1: Let f be the MAJORITY of $h_1, \ldots, h_s$ where each h_i is an AND gate of fan-in r. By Lemma 4.2.6, given any distribution $\mathcal{D}$ there is some AND function h_j such that $|\mathbf{E}_{\mathcal{D}}[fh_j]| \geq 1/s$. It is not hard to show that the L_1-norm of any AND function is at most 4 (see, e.g., [KM93, Lemma 5.1] for a somewhat more general result), so we have $L_1(h_j) \leq 4$. Now the argument from the proof of Lemma 4.4.9 shows that there must be some parity function χ_a such that $|\mathbf{E}_{\mathcal{D}}[f\chi_a]| \geq 1/4s$, where the variables in χ_a are a subset of the variables in h_j – and thus χ_a is a parity of at most r literals. Consequently, we can apply the boosting algorithm of [Fre95] stated in Theorem 4.2.5, choosing the weak hypothesis to be a PARITY with fan-in at most r at each stage of boosting, and be assured that each weak hypothesis has advantage at least $1/4s$ at every stage of boosting. If we boost to accuracy $\epsilon = \frac{1}{2^n+1}$, then the resulting final hypothesis will have zero error with respect to f and will be a MAJORITY of $O(\log(1/\epsilon)/s^2) = O(ns^2)$ many r-PARITY functions. Note that while this argument does not lead to a computationally efficient construction of the desired MAJORITY of r-PARITY, it does establish its existence, which is all we need. □

Remark 4.6.2. Any union (OR) of s many r-rectangles can be expressed as an $O(s)$-MAJORITY of r-rectangles as well.

Theorem 4.4.1 and Lemma 4.6.1 together give us the following:

Theorem 4.6.3. *The concept class $\mathfrak{C}$ consisting of functions expressible as s-MAJORITY of r-rectangles (or, more generally, of s-MAJORITY of r-AND of b-literals which need not necessarily be basic) where $s = \mathrm{poly}(n \log b)$, $r = O(\frac{\log(n \log b)}{\log \log(n \log b)})$ is efficiently learnable under* GHS.

4.6.2 Learning unions of fewer rectangles of higher dimension

We now show that the number of rectangles s and the dimension bound r of each rectangle can be traded off against each other in Theorem 4.6.3 to a limited extent. We state the results below for the case $s = \mathrm{poly}(\log(n \log b))$, but one could obtain analogous results for a range of different choices of s.

We require the following lemma:

Lemma 4.6.4. *Any s-term r-*DNF *can be expressed as an $r^{O(\sqrt{r} \log s)}$-*MAJORITY *of $O(\sqrt{r} \log s)$-*PARITY *of Boolean literals.*

Proof. [KS04, Corollary 13] states that any s-term r-DNF can be expressed as an $r^{O(\sqrt{r} \log s)}$-MAJORITY of $O(\sqrt{r} \log s)$-ANDs. By considering the Fourier representation of an AND, it is clear that each t-AND in the MAJORITY can be replaced by at most $2^{O(t)}$ many t-PARITYs, corresponding to the parities in the Fourier representation of the AND. This gives the lemma. □

Now we can prove our theorem, which gives us roughly a quadratic improvement in the dimension r of rectangles over Theorem 4.6.3 when $s = \mathrm{poly}(\log(n \log b))$.

Theorem 4.6.5. *The concept class $\mathfrak{C}$ consisting of unions of $s = \mathrm{poly}(\log(n \log b))$ many r-rectangles where $r = O(\frac{\log^2(n \log b)}{(\log \log(n \log b) \log \log \log(n \log b))^2})$ is efficiently learnable via Algorithm 8.*

Proof. First note that by Lemma 4.5.4, any function in $C_{n,b}$ can have at most $\kappa = O(rs) = \mathrm{poly}(\log(n \log b))$ non-trivial indices, and at most $\ell = O(s) = \mathrm{poly}(\log(n \log b))$ many i-sensitive values for all $i = 1, \ldots, n$. Now use Lemma 4.6.4 to express any function in $C_{n,b}$ as an s'-MAJORITY of r'-PARITY of basic b-literals where $s' = r^{O(\sqrt{r} \log s)} = \mathrm{poly}(n \log b)$ and $r' = O(\sqrt{r} \log s) = O(\frac{\log(n \log b)}{\log \log \log(n \log b)})$. Finally, apply Theorem 4.5.5 to obtain the desired result. □

Remark 4.6.6. It is possible to obtain a similar result for learning $\mathrm{poly}(\log(n \log b))$ union of $O(\frac{\log^2(n \log b)}{(\log \log(n \log b))^4})$-AND of b-literals if one were to invoke Theorem 4.4.1.

4.6.3 Learning majorities of unions of disjoint rectangles

A set $\{R_1, \ldots, R_s\}$ of rectangles is said to be *disjoint* if every input $x \in [b]^n$ satisfies at most one of the rectangles. Learning unions of disjoint rectangles over $[b]^n$ was studied by [BK98], and is a natural analogue over $[b]^n$ of learning "disjoint DNF" which has been well studied in the Boolean domain (see e.g. [Kha94, ABK$^+$98]).

We observe that when disjoint rectangles are considered Theorem 4.6.3 extends to the concept class of majority of unions of disjoint rectangles; enabling us to improve the depth of our circuits by 1. This extension relies on the following easily verified fact:

Fact 4.6.7. *If $f_1, \ldots, f_t$ are functions from $[b]^n$ to $\{-1, 1\}^n$ such that each x satisfies at most one f_i, then the function* OR$(f_1, \ldots, f_t)$ *satisfies*

$$L_1(\mathrm{OR}(f_1, \ldots, f_t)) = O(L_1(f_1) + \cdots + L_1(f(t))).$$

This fact lets us apply the argument behind Theorem 4.4.8 without modification, and we obtain the following:

Corollary 4.6.8. *The concept class $\mathfrak{C}$ consisting of s-*MAJORITY *of t-*OR *of disjoint r-rectangles where $s, t = \mathrm{poly}(n \log b)$, $r = O(\frac{\log(n \log b)}{\log \log(n \log b)})$ is efficiently learnable under* GHS.

Remark 4.6.9. Only the rectangles connected to the same OR gate must be disjoint in order to invoke Corollary 4.6.8.

4.7 Learning unions of rectangles using uniform quantum examples

In this section we are interested in exploring the consequences of the earlier results in this chapter towards learning with quantum computation. Obviously if we permit our quantum learning algorithm access to quantum membership queries (as defined earlier in Section 3.3), we can replicate all the results in this chapter. This is because quantum computation with quantum membership queries

is stronger than classical computation with classical membership queries. However we will adopt the uniform distribution quantum PAC learning model in which no access to classical or quantum membership queries is permitted.

Recall that the quantum PAC learning model is defined in full generality in Section 3.6. This model, introduced by Bshouty and Jackson in [BJ99], is a natural quantum extension of the classical PAC model introduced by Valiant [Val84]. Some considerable portion of classical learning theory literature is based on membership queries as classical PAC learnable classes are rather limited. In contrast, the uniform distribution quantum PAC model allows access to uniform quantum examples only – which can't simulate classical membership queries (see Lemma 5.2.1).

4.7.1 Sampling from the Fourier spectrum of $\{-1,1\}$-valued functions over $[b]^n$ using uniform quantum examples

Let $f\colon \{0,1\}^n \to \{-1,1\}$ be a Boolean function. [BJ99, Section 5.1] describes an algorithm QSAMP, which has its roots in an idea from [BV97], that uses uniform quantum examples and can sample from the Fourier spectrum of functions defined over Boolean domains. More precisely, QSAMP uses one call to $\mathsf{QEX}(f)$ and its output is distributed as depicted on the left hand side of Table 4.1.

QSAMP(f) given $f\colon \{0,1\}^n \to \{-1,1\}$ produces the following output:		GQSAMP(f) given $f\colon [b]^n \to \{-1,1\}$ is desired to produce the following output:	
$S \subseteq [n]$,	with probability $\frac{\lvert\hat{f}(S)\rvert^2}{2}$,	$\alpha \in [b]^n$,	with probability $\frac{\lvert\hat{f}(\alpha)\rvert^2}{2}$,
FAIL,	with probability $1/2$.	FAIL,	with probability $1/2$.

Table 4.1: Output distribution of QSAMP vs. GQSAMP.

In this section our goal is to extend the idea of Bshouty and Jackson to obtain an analogous quantum subroutine GQSAMP defined for functions over $[b]^n$. In particular, given any $f\colon [b]^n \to \{-1,1\}$, we want each call to GQSAMP to use one quantum example and the distribution of its output to be as depicted on the right hand side of Table 4.1.

As described by [Dam01, Section B.2], a b-level quantum register, which can assume a superposition of the basis states $\{|x\rangle, x \in [b]\}$, can be implemented using $\lceil \log b \rceil$ qubits. Multidimensional quantum registers arise naturally in our discussion as our functions are defined over $[b]^n$. Clearly,

the elements $(x_1, \ldots, x_n) \in [b]^n$ are in one to one correspondence with computational basis states of n b-level quantum registers.

First we define the uniform quantum example oracle associated to a function $f : [b]^n \to \{-1, 1\}$.

Definition 4.7.1. Given $f : [b]^n \to \{-1, 1\}$, *the uniform quantum example oracle* $\mathsf{QEX}(f)$ is the quantum oracle whose query acts over a quantum register of n b-level quantum subregisters and one qubit. A $\mathsf{QEX}(f)$ query can only be invoked in the quantum state $|0^n\rangle|0\rangle$ over which its action is defined as follows:

$$\text{For } f : [b]^n \to \{-1, 1\}, \mathsf{QEX}(f)\colon |0^n\rangle|0\rangle \mapsto \frac{1}{b^{n/2}} \sum_{(x_1,\ldots,x_n)\in[b]^n} |x_1, \ldots, x_n\rangle|\frac{1 - f(x_1, \ldots, x_n)}{2}\rangle.$$

We sometimes refer to a query to the uniform quantum example oracle as a *quantum example*. This definition of a quantum example extends that of [BJ99] (as explained in detail in Section 3.6), originally defined for Boolean valued functions over $\{0, 1\}^n$ analogously to Boolean valued functions over $[b]^n$. Clearly by the term "uniform" we mean the underlying distribution $\mathcal{D}$ (as in the definition of QEX in Section 3.6) is fixed as the uniform distribution over $[b]^n$.

Now follows the main result of this section.

Lemma 4.7.2. *There exists a quantum algorithm* `GQSAMP` *that, given a quantum example oracle* $\mathsf{QEX}(f)$ *for* $f : [b]^n \to \{-1, 1\}$*, returns* $\alpha \in [b]^n$ *with probability* $\frac{|\hat{f}(\alpha)|^2}{2}$ *and* FAIL *with probability* $1/2$.

Proof. First, define the following unitary transformation acting on the computational basis states of n b-level quantum subregisters:

$$T\colon |x_1, x_2, \ldots, x_n\rangle \mapsto |\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n}\rangle, \text{ where } \overline{x_i} \text{ denotes the element } y \in [b], y \equiv -x_i \mod b.$$

The Quantum Fourier Transform over $\mathbb{Z}_b^n$ (see [Dam01, MZ03, Sho97]) is defined on n b-level quantum subregisters as follows:

$$F_{\mathbb{Z}_b^n}\colon |x_1, \ldots, x_n\rangle \mapsto \frac{1}{b^{n/2}} \sum_{(y_1,\ldots,y_n)\in[b]^n} \omega_b^{\sum x_i y_i} |y_1, \ldots, y_n\rangle.$$

Now consider the following algorithm over n b-level quantum subregisters and one qubit:

1. Start with a uniform quantum example $\mathsf{QEX}(f)$ over $|0^n\rangle|0\rangle$. The quantum state after this step will be:

$$\mathsf{QEX}(f)|0^n\rangle|0\rangle = \left(\frac{1}{b^{n/2}} \sum_{x=(x_1,\ldots,x_n)\in[b]^n} |x\rangle|\frac{1-f(x)}{2}\rangle\right)$$

2. Apply the unitary transformation $T \otimes I$ where I denotes the identity operator on the last qubit. This will have the following effect on our registers:

$$(T\otimes I)\left(\frac{1}{b^{n/2}} \sum_{x\in[b]^n} |x\rangle|\frac{1-f(x)}{2}\rangle\right) = \left(\frac{1}{b^{n/2}} \sum_{x\in[b]^n} |\overline{x_1},\ldots,\overline{x_n}\rangle|\frac{1-f(x)}{2}\rangle\right)$$

   The purpose of this transformation is to create the effect of conjugation in

$$\mathbf{E}_{x\in[b]^n}[f(x)\overline{\chi_\alpha(x)}] = \hat{f}(\alpha)$$

   to ultimately set $\frac{\hat{f}(\alpha)}{\sqrt{2}}$ as the amplitude of the state $|\alpha\rangle|1\rangle$.

3. Apply the unitary transformation $F_{\mathbb{Z}_b^n} \otimes F_{\mathbb{Z}_2}$, i.e. the Quantum Fourier Transform over $\mathbb{Z}_b^n$ on the first n b-level quantum subregisters and the Quantum Fourier Transform $F_{\mathbb{Z}_2}$ over $\mathbb{Z}_2$ on the last qubit. The effect will be:

$$\begin{aligned}
&(F_{\mathbb{Z}_b^n}\otimes F_{\mathbb{Z}_2})\left(\frac{1}{b^{n/2}} \sum_{x\in[b]^n} |\overline{x_1},\ldots,\overline{x_n}\rangle|\frac{1-f(x)}{2}\rangle\right)\\
&= \frac{1}{\sqrt{2}b^n} \sum_{\substack{x,\ \alpha\in[b]^n\\ z\in\{0,1\}}} \omega_b^{-\sum x_i\alpha_i}(-1)^{z\cdot\frac{1-f(x)}{2}}|\alpha\rangle|z\rangle\\
&= \frac{1}{\sqrt{2}b^n}\left(\sum_{x,\alpha\in[b]^n} (\overline{\omega_b})^{\sum x_i\alpha_i} f(x)|\alpha\rangle|1\rangle + \sum_{x,\alpha\in[b]^n} (\overline{\omega_b})^{\sum x_i\alpha_i}|\alpha\rangle|0\rangle\right)\\
&= \frac{1}{\sqrt{2}} \sum_{\alpha\in[b]^n} \mathbf{E}_{x\in[b]^n}[f(x)\overline{\chi_\alpha(x)}]|\alpha\rangle|1\rangle + \frac{1}{\sqrt{2}} \sum_{\alpha\in[b]^n} \mathbf{E}_{x\in[b]^n}[\overline{\chi_\alpha(x)}]|\alpha\rangle|0\rangle\\
&= \frac{1}{\sqrt{2}} \sum_{\alpha\in[b]^n} \hat{f}(\alpha)|\alpha\rangle|1\rangle + \frac{1}{\sqrt{2}}|0^n\rangle|0\rangle.
\end{aligned}$$

4. Measure the last qubit. Output FAIL if it collapses to 0, otherwise measure the first n b-level registers and output its value $\alpha \in [b]^n$.

This gives us the desired GQSAMP algorithm due to the above calculation. □

4.7.2 Locating heavy Fourier indices of real valued functions over $[b]^n$ using uniform quantum examples

The idea in this section is due to Bshouty and Jackson [BJ99], we rephrase it for our purposes and provide it here for completeness.

In this section, our goal is to derive an analogous result to Lemma 4.3.2. However unlike Lemma 4.3.2 or its basic underlying tool, Theorem 4.3.1, we are given access to uniform quantum examples only – no membership queries are allowed. Ultimately our construction will give rise to a GHS algorithm running under quantum PAC model assumptions with respect to the uniform distribution.

Recall that the underlying key component of the weak hypothesis construction of GHS in Lemma 4.3.2 is the ability to locate one index β such that $|\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\beta}]| \geq \gamma/2$. Indeed, Theorem 4.3.1 provides a list of indices satisfying this requirement, although only one index is essentially sufficient for our purposes. Equivalently, because $\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\beta}] = \mathbf{E}[b^n\mathcal{D}f\overline{\chi_\beta}]$, this problem is basically the same problem as locating the heavy Fourier elements of $(b^n\mathcal{D}f)\colon [b]^n \to \mathbb{R}$ which gives this section its name.

The following lemma, translating Theorem 4.3.1 into quantum PAC model assumptions, will become our fundamental tool towards this objective. Note that the statement is directly reminiscent of Theorem 4.3.1 if we consider the input function as $g(x) = \zeta(x)f(x)$ – apart from one major difference: the requirement of uniform quantum examples.

Lemma 4.7.3. *There is a quantum algorithm that, given*

- *Uniform quantum examples* $\mathsf{QEX}(f)$*, as defined in Section 4.7.1, where* $f\colon [b]^n \to \{-1, 1\}$*;*
- $\zeta(x)\colon [b]^n \to \mathbb{R}^+ \cup \{0\}$*, by an algorithm computing* $\zeta(x)$ *in time polynomial in* n, $\log b$ *for each* $x \in [b]^n$*; and*
- $0 < \gamma$ *and* $0 < \delta < 1$*,*

outputs a list L such that with probability at least $1-\delta$, we have $\{\alpha\colon |\widehat{(\zeta f)}(\alpha)| > \gamma\} \subseteq L$ and $|\widehat{(\zeta f)}(\beta)| \geq \frac{\gamma}{2}$ for every $\beta \in L$. The running time of the algorithm is polynomial in n, $\log b$, $\|\zeta\|_\infty$, γ^{-1}, $\log(\delta^{-1})$.

Proof. This argument is due to [BJ99]. The key idea is that the problem of constructing a list containing the heavy Fourier indices of ζf can be reduced to locating the heavy Fourier indices of a list of $\{-1,1\}$-valued functions: $\theta_1 f, \theta_2 f, \ldots, \theta_d f$, where $d = O(\log(\|\zeta\|_\infty \gamma^{-1}))$. And then the algorithm of Section 4.7.1 for $\{-1,1\}$-valued functions can be used for this purpose over each $\theta_i f$.

Let

$$d = \lceil \log(3\|\zeta\|_\infty \gamma^{-1}) \rceil \text{ and } \theta(x) = \lfloor 2^d \frac{\zeta(x)}{\|\zeta\|_\infty} \rfloor 2^{-d},$$

$$\text{in other words: } \theta(x) \text{ is truncation of } \frac{\zeta(x)}{\|\zeta\|_\infty} \text{ to its most significant } d \text{ bits.}$$

The error induced by this truncation will not be too large in the following sense:

$$|\mathbf{E}[f\theta\overline{\chi_\alpha}]| \geq \frac{|\mathbf{E}[\zeta f \overline{\chi_\alpha}]|}{\|\zeta\|_\infty} - \frac{\gamma}{3\|\zeta\|_\infty} \quad \text{for all } \alpha \in [b]^n. \tag{4.7.1}$$

Therefore

$$\text{if } |\widehat{(\zeta f)}(\alpha)| > \gamma, \text{ for some } \alpha \in [b]^n, \text{ then } |\mathbf{E}[f\theta\overline{\chi_\alpha}]| > \frac{2\gamma}{3\|\zeta\|_\infty}. \tag{4.7.2}$$

By definition, for all $x \in [b]^n, 0 \leq \theta(x) \leq 1$ and consequently $\theta(x)$ can be written as follows:

$$\theta(x) = \theta_1(x)2^{-1} + \theta_2(x)2^{-2} + \ldots + \theta_d(x)2^{-d} + r(x)2^{-d} \tag{4.7.3}$$

$$\text{where for each } i = 1, \ldots, d, \theta_i\colon [b]^n \to \{-1,1\} \text{ and } r\colon [b]^n \to \{-1,0,1\}.$$

Observe that using the algorithm computing $\zeta(x)$, it is possible to compute $\theta(x)$ and thus each $\theta_i(x)$ in time polynomial in n, $\log b$ for every $x \in [b]^n$.

Let's multiply both sides of equation (4.7.3) with $f\overline{\chi_\alpha}$ and take the expectation over all $x \in [b]^n$. Because of the definition of d and fr is a $\{-1,0,1\}$-valued function, $|\mathbf{E}[fr\overline{\chi_\alpha}]|2^{-d} \leq \frac{\gamma}{3\|\zeta\|_\infty}$. Thus

we obtain:

$$|\mathbf{E}[f\theta\overline{\chi_\alpha}]| \leq \max_i |\mathbf{E}[f\theta_i\overline{\chi_\alpha}]| + \frac{\gamma}{3\|\zeta\|_\infty}. \tag{4.7.4}$$

Combined with the observation in (4.7.2), the inequality (4.7.4) implies:

$$\text{If } |\widehat{(\zeta f)}(\alpha)| > \gamma, \text{ for some } \alpha \in [b]^n, \text{ then there exists } 1 \leq i \leq d, \text{ such that } |\mathbf{E}[f\theta_i\overline{\chi_\alpha}]| > \frac{\gamma}{3\|\zeta\|_\infty}. \tag{4.7.5}$$

Now consider the following algorithm:

1. For each $1 \leq i \leq d$, calculate a list L'_i containing $\{\alpha \colon |\mathbf{E}[f\theta_i\overline{\chi_\alpha}]| > \frac{\gamma}{3\|\zeta\|_\infty}\}$ with probability $1 - \frac{\delta}{3d}$.

   In order to compute such a list L'_i one can proceed as follows: Start with a uniform quantum example $\mathsf{QEX}(f)$, apply the unitary transformation mapping the state

   $$|x, \frac{1-f(x)}{2}\rangle \mapsto |x, \frac{1-\theta_i(x)f(x)}{2}\rangle$$

   and then follow the steps 2–4 of `GQSAMP` algorithm in Section 4.7.1 to sample from the Fourier spectrum of $\theta_i f$. Let's call this algorithm $\mathcal{A}_i$.

   Now note that:

   - By Parseval's Identity there are at most $\mathrm{poly}(\|\zeta\|_\infty, \gamma^{-1})$ indices α for which

     $$|\widehat{(f\theta_i)}(\alpha)| = |\mathbf{E}[f\theta_i\overline{\chi_\alpha}]| > \frac{\gamma}{3\|\zeta\|_\infty}.$$

   - Moreover for any such α, the probability of obtaining α as a result of each iteration of the algorithm $\mathcal{A}_i$ is $\frac{|\widehat{(f\theta_i)}(\alpha)|^2}{2} \geq (\frac{\gamma^2}{18\|\zeta\|_\infty^2})$.

   Therefore in order to run into all such indices with probability at least $1 - \frac{\delta}{3d}$, it is sufficient to invoke the algorithm $\mathcal{A}_i$ $\mathrm{poly}(\|\zeta\|_\infty, \gamma^{-1}, \log(\delta^{-1}))$ times. This will give rise to such a list L'_i with probability $1 - \frac{\delta}{3d}$.

2. Set $L' = \cup_i L'_i$. Due to the observation in (4.7.5), one has $\{\alpha \colon |\widehat{(\zeta f)}(\alpha)| > \gamma\} \subseteq L'$.

3. For every $\beta \in L'$, approximate $|\widehat{(\zeta f)}(\beta)|$ to accuracy $\gamma/4$ with success probability $1 - \frac{\delta}{3|L'|}$. This could be achieved with random examples in $\mathrm{poly}(n, \log b, \|\zeta\|_\infty, \gamma^{-1}, \log(\delta^{-1}))$ steps in total. Discard from L' those β for which the approximation to $|\widehat{(\zeta f)}(\beta)|$ is less than $3\gamma/4$.

4. Output the list of remaining elements L.

This algorithm establishes the result. □

Consequently, we can reexpress Lemma 4.3.2 in terms of uniform quantum examples which gives rise to the GHS algorithm under quantum PAC learning model with respect to uniform distribution.

Lemma 4.7.4 (Construction of the weak hypothesis using uniform quantum examples)**.** *Given*

- *Access to uniform quantum examples* $\mathsf{QEX}(f)$ *where* $f\colon [b]^n \to \{-1, 1\}$*;*
- *A smooth distribution* $\mathcal{D}$*; more precisely, access to an algorithm computing* $\tilde{\mathcal{D}}(x)$ *in time polynomial in* n*,* $\log b$ *for each* $x \in [b]^n$*. Here* $\tilde{\mathcal{D}}$ *is a "pseudo-distribution" for* $\mathcal{D}$*, i.e. there is a value* $c \in [1/2, 3/2]$ *such that* $\tilde{\mathcal{D}}(x) = c\mathcal{D}(x)$ *for all* x*.*
- *A value* $0 < \gamma < 1/2$ *such that there exists an element of the Fourier basis* χ_τ *satisfying* $|\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\tau}]| > \gamma$*,*

there is a quantum algorithm that outputs a weak hypothesis for f *with advantage* $\gamma/4$ *under* $\mathcal{D}$ *with probability* $1 - \delta$ *and runs in time polynomial in* n*,* $\log b$*,* ϵ^{-1}*,* γ^{-1}*,* $\log(\delta^{-1})$*.*

Proof. The proof is essentially the same as Lemma 4.3.2. The only difference is that instead of invoking Theorem 4.3.1 using membership queries, we invoke 4.7.3 using quantum examples with $\zeta(x) = b^n\tilde{\mathcal{D}}(x)$ to locate a heavy Fourier index of $f_*(x) = b^n\tilde{\mathcal{D}}(x)f(x) = \zeta(x)f(x)$. Since the rest of the proof of Lemma 4.3.2 relies on random examples to f, these can easily be simulated using uniform quantum examples $\mathsf{QEX}(f)$. □

Corollary 4.7.5 (The Generalized Harmonic Sieve with uniform quantum examples)**.** *Let* $\mathfrak{C}$ *be a concept class. Suppose that for any concept* $f \in C_{n,b}$ *and any distribution* $\mathcal{D}$ *over* $[b]^n$ *with* $L_\infty(\mathcal{D}) < \mathrm{poly}(\epsilon^{-1})/b^n$ *there exists a Fourier basis element* χ_α *such that* $|\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\alpha}]| \geq \gamma$*. Then*

$\mathfrak{C}$ *can be learned by a quantum algorithm using* $\mathrm{poly}(n, \log b, \epsilon^{-1}, \gamma^{-1})$ *time including queries to the uniform quantum example oracle* QEX *(i.e. there exist uniformly generated quantum circuits in this amount of time as described in Section 2.1).*

Note the similarity between Corollary 4.7.5 and Corollary 4.3.3. The main difference is that Corollary 4.7.5 uses uniform quantum examples as opposed to membership queries to extract information about the concept. Clearly Corollary 4.7.5 also requires quantum computation – but only for the construction of the weak hypothesis, not for the boosting.

Let us go back to our main application of learning unions of rectangles in Section 4.6.

At this point, thanks to Corollary 4.7.5, we can immediately build over the results of Section 4.4. Consequently, Theorems 4.4.1 and 4.6.3 as well as Corollary 4.6.8 carry over, as they are based on the results of Section 4.4, to give us equivalent learnability results in the uniform distribution quantum PAC model.

The only remaining result of Section 4.4 is Theorem 4.6.5. Note that the idea of locating sensitive elements and the underlying techniques in Section 4.5 relies on membership queries. However, we are interested in algorithms using uniform quantum examples only for uniform distribution quantum PAC learning. Consequently, we cannot build over the results of Section 4.5 as we did earlier in the proof of Theorem 4.6.5. Nevertheless, by using Theorem 4.4.1 directly as pointed out by Remark 4.6.6, we can obtain a similar albeit slightly coarser result in the uniform distribution quantum PAC model.

Chapter 5

Quantum Algorithms for Testing and Learning Juntas

5.1 Introduction

5.1.1 Motivation

A common theme in the literature of quantum computational learning theory is that these works study: quantum learning / testing algorithms that use purely quantum oracles (such as the quantum membership oracle of Section 3.3 or the quantum example oracle of Section 3.6). For instance, [BJ99] modifies the Harmonic Sieve algorithm of [Jac97] so that it uses only uniform quantum examples to learn DNF formulas. [BFNR03] considers the problem of quantum property testing using quantum membership queries to give an exponential separation between classical and quantum testers for certain concept classes. In Chapter 3 (based on the article [AS05]), we considered the information theoretic requirements of exact learning and partition learning using quantum membership queries as well as PAC learning using quantum examples. Many other articles including [SG04, AIK$^+$04a, HMP$^+$03] further extend this list.

As the problem of building large scale quantum computers remains a major challenge, it is natural to question the technical feasibility of large scale implementation of the quantum oracles considered in the literature. It is desirable to minimize the number of quantum oracle queries required by quantum algorithms. Thus motivated, in this chapter we are interested in designing

algorithms in a new framework: algorithms with access to both quantum and classical sources of information (with a goal of minimizing the quantum resources required).

5.1.2 The results of this chapter

All of our positive results are based off of a quantum subroutine due to [BJ99], which we will refer to as a FS (Fourier Sample) oracle call. As explained in Section 5.2, a call to the FS oracle yields a set drawn according to the Fourier spectrum of the target Boolean function. As demonstrated by [BJ99], such an oracle can be implemented using $O(1)$ uniform quantum examples from a $\mathsf{QEX}(f, \mathfrak{U})$ oracle. In fact, our algorithms will be purely classical apart from the FS oracle access. This approach allows us to abstract away from the intricacies of quantum computation, and renders our results useful towards any setting in which such a subroutine can be provided to the user. Hence, in a sense learning and testing with FS oracle queries can be regarded as a new distinct model, which may possibly be weaker than the uniform distribution Quantum PAC model.

All our algorithms can be implemented within the (uniform distribution) quantum PAC model first proposed by [BJ99]. Recall that the quantum PAC learning model is defined in full generality in Section 3.6. This model is a natural quantum extension of the classical PAC model introduced by Valiant [Val84]. Consequently no access to classical or quantum membership queries are permitted or used by our algorithms. However, calls to the FS oracle and classical random examples are allowed as they can be simulated by $O(1)$ uniform quantum examples. Some considerable portion of classical learning theory literature is based on membership queries as classical PAC learnable classes are rather limited. In contrast, the quantum PAC model under uniform distribution allow access to uniform quantum examples only – which can't efficiently simulate classical membership queries in general [BJ99].

We are primarily interested in the information theoretic requirements (i.e. the number of queries or oracle calls needed) of the learning and testing problems that we discuss. We give upper and lower bounds for a range of learning and testing problems.

Our first result, in Section 5.3, is a k-junta testing algorithm which uses $O(k/\epsilon)$ FS oracle calls. Our algorithm uses fewer queries than the best known classical junta testing algorithm due to Fischer *et al.* [FKR$^+$04], which uses $O((k \log k)^2 \epsilon^{-1})$ membership queries. However, since the

best lower bound known for membership query based junta testing (due to Chockler and Gutfreund [CG04]) is $\Omega(k)$, our result does not rule out the possibility that there might exist a classical membership query algorithm with the same query complexity.

To complement our FS based testing algorithm, we establish a new lower bound: Any FS based k-junta testing algorithm requires $\Omega(\sqrt{k})$ calls to the FS oracle. This shows that our our testing algorithm is not too far from optimal.

Next we consider algorithms that can both make FS queries and also access classical random examples. In Section 5.4 we give an algorithm for learning k-juntas over $\{-1,1\}^n$ that uses $O(\epsilon^{-1}k\log k)$ FS queries and $O(\log(\epsilon^{-1})2^k)$ random examples. Since any classical learning algorithm requires $\Omega(2^k+\log n)$ examples (even if it is allowed to use membership queries), this result illustrates that it is possible to reduce the classical query complexity substantially (in particular, to eliminate the dependence on n) if the learning algorithm is also permitted to have some very limited quantum information. Moreover most of the consumption of our algorithm is from classical random examples which are considered quite "cheap" relative to quantum examples. From another perspective, our result shows that for learning k-juntas, almost all the quantum examples used by the algorithm of Bshouty and Jackson [BJ99] can in fact be converted into ordinary classical random examples.

Finally, in Section 5.5 we present a learning algorithm for polynomially t-sparse functions using $O(\frac{t}{\epsilon}\log\frac{t}{\epsilon})$ FS oracle calls and random examples. As shown by [KM93], some important and natural function classes such as decision trees and functions with bounded L_1 norm are polynomially sparse. We give an $\Omega(n\sqrt{t}+t)$ lower bound on the number of classical membership queries that are required to learn t-sparse functions; since our quantum algorithm's sample complexity has no dependence on n, it goes beyond what is possible for a classical learning algorithm using membership queries. Our algorithm also compares favorably to using the [BJ99] algorithm for learning t-sparse functions in terms of the number of quantum examples used.

5.1.3 Organization of this chapter

In Section 5.2 we describe the models and problems we will consider and present some useful preliminaries from Fourier analysis and probability. Sections 5.3 and 5.4 give our results on testing

and learning juntas respectively. Section 5.5 extends our results on learning juntas to a more general setting of learning functions whose Fourier representations are "polynomially sparse."

5.2 Preliminaries

5.2.1 The problems and the models

In this chapter, a *concept* c over $\{-1,1\}^n$ is a Boolean function $c : \{-1,1\}^n \to \{-1,1\}$, where -1 stands for TRUE and 1 stands for FALSE. A *concept class* $\mathfrak{C} = \cup_{n\geq 1} C_n$ is a set of concepts where C_n consists of those concepts in $\mathfrak{C}$ whose domain is $\{-1,1\}^n$. For ease of notation throughout the chapter we will omit the subscript in C_n and simply write C to denote a collection of concepts over $\{-1,1\}^n$.

The concept class we will chiefly be interested in is the class of *k-juntas*. A Boolean function $f\colon \{-1,1\}^n \to \{-1,1\}$ is a k-junta if f depends only on k out of its n input variables.

The problems:

We are interested in the following computational problems:

Learning under the uniform distribution: Given any *target concept* $c \in C$, an *ϵ-learning algorithm for concept class C* under the uniform distribution outputs a *hypothesis* h with probability at least $2/3$ which agrees with c on at least $1-\epsilon$ fraction of the inputs. This is a widely studied problem in learning theory literature both in classical (see for instance [KM93, Jac97]) and in quantum (see [BJ99]) versions.

Property testing: Let c be any Boolean function $c : \{-1,1\}^n \to \{-1,1\}$. A *property testing algorithm for concept class C* is an algorithm which, given access to c, behaves as follows:

- If $c \in C$ then the algorithm outputs ACCEPT with probability at least $2/3$.
- If c is *ϵ-far* from any concept in C, i.e. for any concept $f \in C$, c and f differ on at least ϵ fraction of the inputs, then the algorithm outputs REJECT with probability at least $2/3$.

Knowing that a concept depends on only a small number of variables can be especially useful in the context of learning. The notion of property testing was first developed by [GGR98] and [RS96]. Quantum property testing was first studied by Buhrman *et al.* [BFNR03], which first gave an example of an exponential separation between classical and quantum testers.

Note that a learning or testing algorithm for C "knows" the class C but does not know the identity of the target concept $c \in C$. We are interested in *time efficient* algorithms for both of these problems: Algorithms running in $\mathrm{poly}(n, 2^k, \epsilon^{-1})$ time for k-juntas and $\mathrm{poly}(n, t, \epsilon^{-1})$ time for polynomially t-sparse functions.

Classical oracles:

Naturally, in order for the learning and testing algorithms to gather information about the target concept they need a an information source called an *oracle*. The number of times an oracle is queried by an algorithm is referred to as the *query complexity*. Sometimes the algorithms will be allowed access to more than one type of oracle in our discussion.

Throughout this chapter the following oracles providing classical information will be discussed:

Membership oracle MQ**:** As defined earlier in Section 3.2.2, a *membership oracle* $\mathsf{MQ}(f)$ is an oracle which, when queried with input x, outputs the label $f(x)$ assigned by the target f to the input.

Uniform random examples EX**:** A query $\mathsf{EX}(f)$ of the random example oracle returns an ordered pair $\langle x, f(x)\rangle$ where x is drawn uniformly random from the set of all inputs. This is the same oracle as $\mathsf{EX}(f, \mathfrak{U})$ as defined in Section 3.2.2, where $\mathfrak{U}$ denotes the uniform distribution over $\{-1, 1\}^n$.

Clearly single call to an MQ oracle can simulate the random example oracle EX. Indeed EX oracle queries are considered "cheap" compared to membership queries. For example, in many settings one can get random examples but can't see a desired input (e.g. weather, stock market). Note that the set of concept classes that are efficiently PAC learnable with uniform random examples only is rather limited. In contrast, there are known efficient algorithms using membership queries

for learning important function classes such as DNF formulas [Jac97]. Recall that classical PAC learnability was discussed earlier in Section 3.2.2.

Quantum oracles:

We will consider the following quantum oracles. These are the quantum generalizations of membership queries and uniform random examples respectively.

Quantum membership queries QMQ**:** The quantum membership oracle $\mathsf{QMQ}(f)$ is the quantum oracle whose query acts on the computational basis states as follows:

$$\mathsf{QMQ}(f)\colon |x, b\rangle \mapsto |x, b \odot f(x)\rangle, \text{ where } x \in \{-1,1\}^n \text{ and } b \in \{-1,1\}.$$

Uniform quantum examples QEX**:** The uniform quantum example oracle $\mathsf{QEX}(f)$ is the quantum oracle whose query acts on the computational basis state $|1^n, 1\rangle$ as follows:

$$\mathsf{QEX}(f)\colon |1^n, 1\rangle \mapsto \sum_{x\in\{-1,1\}^n} \frac{1}{2^{n/2}} |x, f(x)\rangle.$$

The action of a $\mathsf{QEX}(f)$ query is undefined on other basis states, and an algorithm may only invoke the $\mathsf{QEX}(f)$ query on the basis state $|1^n, 1\rangle$.

Note that the above definitions for $\mathsf{QMQ}(f)$ and $\mathsf{QEX}(f)$ are essentially equivalent to $\mathsf{QMQ}(f)$ in Section 3.3 and $\mathsf{QEX}(f, \mathfrak{U})$ in Section 3.6 respectively. There is only one notational difference: since our concepts are functions $f\colon \{-1,1\}^n \to \{-1,1\}$, we name the computational basis states of a qubit as $|1\rangle, |-1\rangle$ instead of $|0\rangle, |1\rangle$. Under this new convention, the action of the queries to $\mathsf{QMQ}(f)$ and $\mathsf{QEX}(f)$ are expressed as above.

It is clear that a QMQ oracle can simulate QEX and MQ oracles, and a QEX oracle can simulate a EX oracle.

The quantum PAC learning model, as defined in full generality in Section 3.6, was introduced by Bshouty and Jackson in [BJ99]. As is the case with classical PAC learning vs. learning with membership queries, this model, allowing access to QEX queries only, is weaker than learning with quantum membership queries under uniform distribution. In fact, the following fact due to

[BJ99, Theorem 5] demonstrates that even the classical membership queries cannot be efficiently simulated by quantum examples.

Lemma 5.2.1 (See [BJ99])**.** *For* DNF *formulas membership queries* MQ *cannot be simulated by a polynomial size quantum network using* QEX *oracle queries.*

5.2.2 Harmonic analysis of functions over $\{-1,1\}^n$

We will make use of the Fourier expansion of real valued functions over $\{-1,1\}^n$. We write $[n]$ to denote the set of variables $\{x_1, x_2, \ldots, x_n\}$.

Consider the set of real valued functions over $\{-1,1\}^n$ endowed with the inner product

$$\langle f, g\rangle = \mathbf{E}[fg] = \frac{1}{2^n}\sum_x f(x)g(x)$$

and induced norm $\|f\| = \sqrt{\langle f, f\rangle}$. For each $S \subseteq [n]$, let χ_S be the parity function $\chi_S(x) = \prod_{x_i \in S} x_i$. It is a well known fact that the 2^n functions $\{\chi_S(x), S \subseteq [n]\}$ form an orthonormal basis for the vector space of real valued functions over $\{-1,1\}^n$ with the above inner product. Consequently, every $f\colon \{-1,1\}^n \to \mathbb{R}$ can be expressed uniquely as:

$$f(x) = \sum_{S\subseteq[n]} \hat{f}(S)\chi_S(x)$$

which we refer to as the *Fourier expansion* or *Fourier transform* of f. Alternatively, the values $\{\hat{f}(S)\colon S \subseteq [n]\}$ are called the *Fourier coefficients* or the *Fourier spectrum* of f. Recall that analogous notions were introduced in Section 4.2.3 for functions over $[b]^n$.

Parseval's Identity relates the values of the coefficients to the values of the function:

Lemma 5.2.2 (Parseval's Identity)**.** *For any $f\colon \{-1,1\}^n \to \mathbb{R}$, we have $\sum_{S\subseteq[n]} |\hat{f}(S)|^2 = \mathbf{E}[f^2]$. Thus for a Boolean valued function $\sum_{S\subseteq[n]} |\hat{f}(S)|^2 = 1$.*

We write $L_1(f)$ to denote $\sum_{S\subseteq[n]} |\hat{f}(S)|$. We will use the following simple and well-known fact:

Fact 5.2.3 (See [KM93]). *For any $f\colon \{-1,1\}^n \to \{-1,1\}$ and any $g : \{-1,1\}^n \to \mathbf{R}$, we have*

$$\mathbf{Pr}_x[f(x) \neq \mathrm{sgn}(g(x))] \leq \mathbf{E}_x[(f(x) - g(x))^2] = \sum_{S\subseteq[n]} |\hat{f}(S) - \hat{g}(S)|^2$$

Recall that the *influence* of a variable x_i on a Boolean function f is the probability (taken over a uniform random input x for f) that f changes its value when the i-th bit of x is flipped, i.e.

$$\mathrm{Inf}_i(f) = \mathbf{Pr}_x[f(x_i \leftarrow -1) \neq f(x_i \leftarrow 1)].$$

It is well known (see e.g. [KKL88]) that $\mathrm{Inf}_i(f) = \sum_{S\ni x_i} |\hat{f}(S)|^2$.

5.2.3 Additional tools

Fact 5.2.4 (Data Processing Inequality). *Let X_1, X_2 be two random variables over the same domain. For any (possibly randomized) algorithm $\mathcal{A}$, one has that*

$$\|\mathcal{A}(X_1) - \mathcal{A}(X_2)\|_1 \leq \|X_1 - X_2\|_1.$$

Let S_1, S_2 be random variables corresponding to sequences of draws taken from two different distributions over the same domain. By the above inequality, if $\|S_1 - S_2\|_1$ is known to be small, then the probability of success must be small for any algorithm designed to distinguish if the draws are made according to S_1 or S_2.

We will also use standard Hoeffding and Chernoff bounds as introduced in Chapter 2.

5.2.4 The Fourier sampling oracle: FS

Definition 5.2.5. The *Fourier sampling oracle* $\mathsf{FS}(f)$ is the classical oracle whose query returns a subset of variables S with probability $|\hat{f}(S)|^2$, where $\hat{f}(S)$ denotes the corresponding Fourier coefficient as defined in Section 5.2.2. Note that due to Parseval's Identity given by Lemma 5.2.2, $\sum_{S\subseteq[n]} |\hat{f}(S)|^2 = 1$.

This oracle will play an important role in our algorithms.

In [BJ99] Bshouty and Jackson develops a constant size quantum network QSAMP, which has its roots in an idea by [BV97]. See Section 4.7.1 for a detailed description of the QSAMP network's behaviour. QSAMP allows sampling from the Fourier spectrum of a Boolean function using $O(1)$ QEX oracle queries:

Lemma 5.2.6 (See [BJ99])**.** *For any Boolean function f, it is possible to simulate a draw from the* $\mathsf{FS}(f)$ *oracle with probability* $1-\delta$ *using* $O(\log \delta^{-1})$ $\mathsf{QEX}(f)$ *queries.*

Proof. Recall the behavior of QSAMP in [BJ99] explained in Section 4.7.1. When QSAMP terminates successfully, it returns a draw from FS oracle. As described in Section 4.7.1, QSAMP is successful $1/2$ of the time, and thus by repeating the QSAMP algorithm $O(\log \delta^{-1})$ times it is possible to obtain a draw from the FS oracle with probability $1-\delta$. □

All the algorithms we describe are otherwise classical algorithms with FS queries.

5.3 Testing juntas

Fischer *et al.* studied the problem of testing juntas given black-box access (i.e., classical membership query access) to the unknown function f using harmonic analysis and probabilistic methods. They gave several different algorithms giving rise to upper bounds *with no dependence on* n, the most efficient of which yields the following:

Theorem 5.3.1 (See [FKR$^+$04, Theorem 6])**.** *There is an algorithm that tests the property of being a k-junta using $O((k \log k)^2 \epsilon^{-1})$ membership queries.*

The algorithm giving rise to the above test works by randomly partitioning the coordinates into subsets and finding those subsets with non-negligible influence by considering them in blocks.

Fischer *et al.* also gave a lower bound on the number of queries required for testing juntas, which was subsequently improved by Chockler *et al.* to the following:

Theorem 5.3.2 (See [CG04])**.** *Any algorithm that tests whether f is a k-junta must use $\Omega(k)$ membership queries.*

We emphasize that that both of these results concern algorithms with classical membership query access.

5.3.1 A testing algorithm using $O(k/\epsilon)$ FS oracle calls

In this section we describe a new testing algorithm that uses the FS oracle and prove the following theorem about its performance:

Theorem 5.3.3. *There is an algorithm that tests the property of being a k-junta using $O(k/\epsilon)$ calls to the* FS *oracle.*

As described in Section 5.2, the algorithm can thus be implemented using $O(k/\epsilon)$ uniform quantum examples from $\mathsf{QEX}(f)$.

Proof. Consider the following algorithm $\mathcal{A}$ which has FS oracle access to an unknown function $f\colon \{-1,1\}^n \to \{-1,1\}$. Algorithm $\mathcal{A}$ first makes $10(k+1)/\epsilon$ calls to the FS oracle; let $\mathcal{S}$ denote the union of all the sets of variables received as responses to these oracle calls. Algorithm $\mathcal{A}$ then outputs "ACCEPT" if $|\mathcal{S}| \leq k$ and outputs "REJECT" if $|\mathcal{S}| > k$.

It is clear that if f is a k-junta then $\mathcal{A}$ outputs "ACCEPT" with probability 1. To prove correctness of the test it suffices to show that if f is ϵ-far from any k-junta then $\mathbf{Pr}[\mathcal{A}$ outputs "REJECT"$] \geq \frac{2}{3}$.

The argument is similar to the standard analysis of the coupon collector's problem. Let us view the set $\mathcal{S}$ as growing incrementally step by step as successive calls to the FS oracle are performed.

Let X_i be a random variable which denotes the number of FS queries that take place starting immediately after the $(i-1)$-st new variable is added to $\mathcal{S}$, up through the draw when the i-th new variable is added to $\mathcal{S}$. By assumption, if the $(i-1)$-st and i-th new variables are obtained in the same draw then $X_i = 0$. (For example, if the first three queries to the FS oracle are $\{1,2,4\}$, $\{2,4\}$, $\{1,4,5,6\}$, then we would have $X_1 = 1$, $X_2 = 0$, $X_3 = 0$, $X_4 = 2$, $X_5 = 0$.)

Since f is ϵ-far from any k-junta, we know that for any set $\mathcal{T}$ of $k' \leq k$ variables, it must be the case that

$$\sum_{S \subseteq \mathcal{T}} \hat{f}(S)^2 \leq 1 - \epsilon$$

(since otherwise if we set $g = \sum_{S \subseteq \mathcal{T}} \hat{f}(S)\chi_S$, $h = \mathrm{sgn}(g)$ and use Fact 5.2.3, we would have

$$\mathbf{Pr}_x[f(x) \neq h(x)] \leq \mathbf{E}_x[(f(x) - g(x))^2] = \sum_{S \not\subseteq \mathcal{T}} \hat{f}(S)^2 < \epsilon$$

which contradicts the fact that f is ϵ-far from any k-junta). It follows that for each $1 \leq i \leq k$, if at the current stage of the construction of $\mathcal{S}$ we have $|\mathcal{S}| = i$, then the probability that the next FS query yields a new variable outside of $\mathcal{S}$ is at least ϵ. Consequently we have $\mathbf{E}[X_i] \leq \frac{1}{\epsilon}$ for each $1 \leq i \leq k+1$, and hence

$$\mathbf{E}[X_1 + \cdots + X_{k+1}] \leq \frac{(k+1)}{\epsilon}.$$

By Markov's inequality, the probability that $X_1 + \cdots + X_{k+1} \leq 10(k+1)/\epsilon$ is at least $9/10$, and therefore with probability at least $9/10$ it will be the case after $10(k+1)/\epsilon$ draws that $|\mathcal{S}| > k$ and the algorithm will consequently output "REJECT." □

Note that the $O(k/\epsilon)$ uniform quantum examples required for Algorithm $\mathcal{A}$ improves on the $O(k^2/\epsilon^2)$ query complexity of the best known classical algorithm. However our result does not conclusively show that QEX queries are more powerful than classical membership queries for this problem since it is conceivable that there could exist an as yet undiscovered $O(k/\epsilon)$ classical membership query algorithm.

5.3.2 Lower bounds for the FS oracle based testing

A first approach

As a first attempt to obtain a lower bound on the number of FS oracle calls required to test k-juntas, it is natural to consider the approach of Chockler *et al.* from [CG04]. To prove Theorem 5.3.2, Chockler *et al.* show that any classical algorithm which can successfully distinguish between the following two probability distributions over black-box functions must use $\Omega(k)$ queries:

- **Scenario I:** The distribution $\mathcal{D}_{k,n}^{(0)}$ is uniform over the set of all Boolean functions over n variables which do not depend on variables $k+2, \ldots, n$.
- **Scenario II:** The distribution $\mathcal{D}_{k,n}^{(1)}$ is defined as follows: to draw a function f from this distribution, first an index i is chosen uniformly from $1, \ldots, k+1$, and then f is chosen uniformly from among those functions that do not depend on variables $k+2, \ldots, n$ or on variable i.

The following observation shows that this approach will not yield a strong lower bound for algorithms that have access to a FS oracle:

Observation 5.3.4. *With $O(\log k)$ queries to a* FS *oracle, it is possible to determine w.h.p. whether a function f is drawn from Scenario I or Scenario II.*

Proof. It is easy to see that a function drawn from Scenario I is simply a random function on the first $k+1$ variables. The Fourier spectrum of random Boolean functions is studied in [OS03], where it is shown that sums of squares of Fourier coefficients of random Boolean functions are tightly concentrated around their expected value. In particular, Proposition 6 of [OS03] directly implies that for any fixed variable $x_i, i \in 1, \ldots, k+1$, we have:

$$\mathbf{Pr}_{f \leftarrow \mathcal{D}^{(0)}_{k,n}} \left[\left| \sum_{S \ni x_i} |\hat{f}(S)|^2 - \frac{1}{2} \right| > \frac{1}{6} \right] < \exp(-2^{k+1}/144).$$

Thus with overwhelmingly high probability, if f is drawn from Scenario I then each FS query will "expose" variable i with probability at least $1/3$. It follows that after $O(\log k)$ queries all $k+1$ variables will have been exposed; so by making $O(\log k)$ FS queries and simply checking whether or not $k+1$ variables have been exposed, one can determine w.h.p. whether f is drawn from Scenario I or Scenario II. □

Thus we must adopt a more sophisticated approach to prove a strong lower bound on FS oracle algorithms.

An $\Omega(\sqrt{k})$ lower bound for FS oracle algorithms

Our main result in this section is the following theorem:

Theorem 5.3.5. *Any algorithm that has* FS *oracle access to an unknown f must use $\Omega(\sqrt{k})$ oracle calls to test whether f is a k-junta.*

Proof. Let k be such that $k = r + 2^{r-1}$ for some positive integer r. We let R denote 2^r. The *Addressing function* on $r+R$ variables has r "addressing variables," which we shall denote $x_1, \ldots, x_r$, and $R = 2^r$ "addressee variables" which we denote $z_0, \ldots, z_{R-1}$. The output of

the function is the value of variable $z_{\mathbf{x}}$ where the "address" $\mathbf{x}$ is the element of $\{0, \ldots, R-1\}$ whose binary representation is given by $x_1 \ldots x_r$. Figure 5.1 depicts a decision tree that computes the general Addressing function and Figure 5.2 with $r = 3$. Formally, the Addressing function $\textsc{Addressing} : \{-1,1\}^{r+R} \to \{-1,1\}$ is defined as follows:

$\textsc{Addressing}(x_1, x_2, \ldots, x_r, z_0, z_1, \ldots, z_{R-1}) = z_{\mathbf{x}}$,
where $\mathbf{x} = (\frac{1-x_1}{2}) \circ (\frac{1-x_2}{2}) \circ \ldots \circ (\frac{1-x_r}{2})$ in binary form and $\circ$ is binary concatenation.

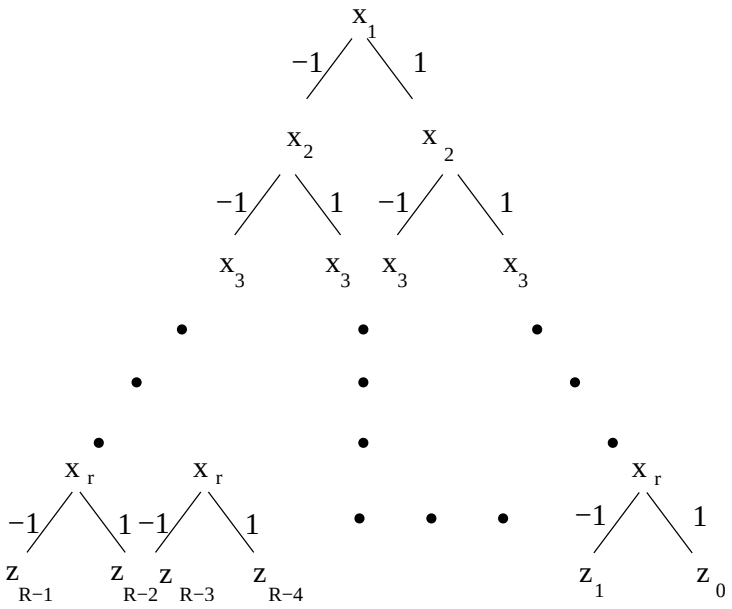

Figure 5.1: The decision tree for the Addressing function.

Intuitively, the Addressing function will be useful for us because as we will see the Fourier spectrum is "spread out" over the R addressee variables; this will make it difficult to distinguish the Addressing function (which is not a k-junta since $k = r + R/2$) from a variant which is a k-junta.

Let $x_1, \ldots, x_r, y_0, \ldots, y_{n-r-1}$ be the n variables that our Boolean functions are defined over. We now define two distributions $\mathcal{D}_{\textsc{Reject}}$, $\mathcal{D}_{\textsc{Accept}}$ over functions on these variables.

The distribution $\mathcal{D}_{\textsc{Reject}}$ is defined as follows: to make a draw from $\mathcal{D}_{\textsc{Reject}}$,

1. First uniformly choose a subset T of R variables from $\{y_0, \ldots, y_{n-r-1}\}$;

2. Next, replace the variables $z_0, \ldots, z_{R-1}$ in the function

$$\textsc{Addressing}(x_1, \ldots, x_r, z_0, \ldots, z_{R-1})$$

with the variables in T (choosing the variables from T in a uniformly random order). Return the resulting function.

Note that step (2) in the description of making a draw from $\mathcal{D}_{\text{REJECT}}$ above corresponds to placing the variables in T uniformly at the leaves of the decision tree for ADDRESSING (see Figure 5.1).

Equivalently, if we write f_τ to denote the following function *over n variables*

$$f_\tau(x_1, \ldots, x_r, y_0, \ldots, y_{n-r-1}) = \textsc{Addressing}(x_1, x_2, \ldots, x_r, y_{\tau(0)}, y_{\tau(1)}, \ldots, y_{\tau(R-1)}); \tag{5.3.1}$$

a draw from $\mathcal{D}_{\text{REJECT}}$ is a function chosen uniformly at random from the set $C_{\text{REJECT}} = \{f_\tau\}$ where τ ranges over all permutations of $\{0, \ldots, n-r-1\}$.

It is clear that every function in C_{REJECT} (the support of $\mathcal{D}_{\text{REJECT}}$) depends on $r + R$ variables and thus is not a k-junta. In fact, every function in C_{REJECT} is far from being a k-junta:

Lemma 5.3.6. *Every f that has nonzero probability under $\mathcal{D}_{\text{REJECT}}$ is $1/6$-far from any k-junta.*

Proof. Fix any such f and let g be any k-junta. It is clear that at least $R/2 - r$ of the "addressee" variables of f are not relevant variables for g. For a $\frac{R/2-r}{R} > 1/3$ fraction of all inputs to f, the value of f is determined by one of these addressee variables; on such inputs the error rate of g relative to f will be precisely $1/2$. □

Considering the contribution to the Fourier spectrum from each leaf of the decision tree, we obtain the following expression for the Fourier representation of each f_τ in C_{REJECT}:

$$f_\tau(x_1, \ldots, x_r, y_0, \ldots, y_{n-r-1}) = \sum_{\mathbf{i}=i_1 i_2 \ldots i_r=0}^{R-1} y_{\tau(\mathbf{i})} \left(\frac{1+(-1)^{i_1} x_1}{2}\right)\left(\frac{1+(-1)^{i_2} x_2}{2}\right) \ldots \left(\frac{1+(-1)^{i_r} x_r}{2}\right) \tag{5.3.2}$$

$$= \frac{1}{2^r} \sum_{\mathbf{i}=0}^{R-1} \sum_{X \subseteq \{x_1,\ldots,x_r\}} (-1)^{(\sum_{x_j \in X} i_j)} y_{\tau(\mathbf{i})} \chi_X. \tag{5.3.3}$$

Note that whenever $\frac{1-x_1}{2} = i_1, \frac{1-x_2}{2} = i_2, \ldots, \frac{1-x_r}{2} = i_r$, the sum on the RHS of Equation (5.3.2) has precisely one non-zero term which is $y_{\tau(\mathbf{i})}$. This is because the rest of the terms are annihilated since in each of these terms there is some index j such that $\frac{1-x_j}{2} = 1 - i_j$ which makes $(\frac{1+(-1)^{i_j} x_j}{2}) = 0$. Consequently this sum gives rise to exactly the Addressing function in Equation (5.3.1) which is defined as f_τ and consequently the equality in Equation (5.3.2) follows.

Now we turn to $\mathcal{D}_{\text{ACCEPT}}$.

The distribution $\mathcal{D}_{\text{ACCEPT}}$ is defined as follows: to make a draw from $\mathcal{D}_{\text{ACCEPT}}$,

1. First uniformly choose a subset T of $R/2$ variables from $\{y_0, \ldots, y_{n-r-1}\}$;

2. Next, replace the variables $z_0, \ldots, z_{R/2-1}$ in the function

$$\text{ADDRESSING}(x_1, \ldots, x_r, z_0, \ldots, z_{R-1})$$

   with the variables in T (choosing the variables from T in a uniformly random order).

3. Finally, for each $\mathbf{i} = 0, \ldots, R/2 - 1$ do the following: if variable y_j was used to replace variable $z_{\mathbf{i}}$ in the previous step, let $s_{\mathbf{i}}$ be a fresh uniform random ± 1 value and replace variable $z_{R-1-\mathbf{i}}$ with $s_{\mathbf{i}} y_j$. Return the resulting function.

Observe that for any integer $0 \leq \mathbf{i} < R/2$ with binary expansion $\mathbf{i} = i_1 \circ i_2 \circ \cdots \circ i_r$, we have that the binary expansion of $R - 1 - \mathbf{i}$ is $\overline{i_1} \circ \overline{i_2} \circ \cdots \circ \overline{i_r}$. Thus steps (2) and (3) in the description of making a draw from $\mathcal{D}_{\text{ACCEPT}}$ may be restated as follows in terms of the decision tree representation for ADDRESSING:

2′. Place the variables $y_j \in T$ randomly among the leaves of the decision tree with index less than $R/2$.

3′. For each variable $y_j \in T$ placed at the leaf with index $\mathbf{i} = i_1 \circ i_2 \circ \cdots \circ i_r < R/2$ above, throw a ± 1 valued coin $s_{\mathbf{i}}$ and place $s_{\mathbf{i}} y_j$ at the antipodal leaf location with index: $\bar{\mathbf{i}} = \overline{i_1} \circ \overline{i_2} \circ \cdots \circ \overline{i_r} = R - 1 - \mathbf{i}$.

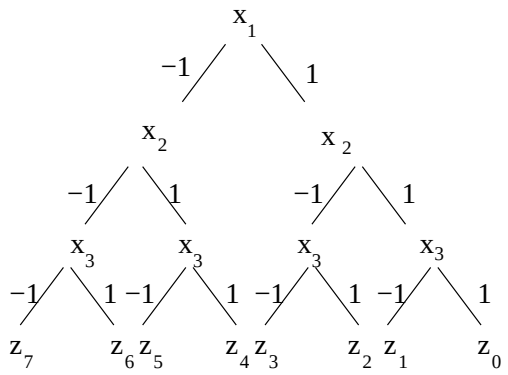

Figure 5.2: The decision tree for the Addressing function with $r = 3$.

Equivalently, if we write $g_{\tau,s}$ to denote the following function *over* n *variables*

$$g_{\tau,s}(x_1,\ldots,x_r,y_0,\ldots,y_{n-r-1}) =$$
$$\textsc{Addressing}(x_1,\ldots,x_r,y_{\tau(0)},\ldots,y_{\tau(R/2-1)},s_{(R/2-1)}y_{\tau(R/2-1)},\ldots,s_0y_{\tau(0)}); \tag{5.3.4}$$

a draw from $\mathcal{D}_{\textsc{Accept}}$ is a function chosen uniformly at random from the set $C_{\textsc{Accept}} = \{g_{\tau,s}\}$ where τ ranges over all permutations of $\{0,\ldots,n-r-1\}$ and s ranges over all of $\{-1,1\}^{R/2}$. It is clear that every function in $C_{\textsc{Accept}}$ depends on at most $r + R/2 = k$ variables, and thus is indeed a k-junta.

By considering the contribution to the Fourier spectrum from each pair of leaves $\mathbf{i}, \bar{\mathbf{i}}$ of the decision tree, we obtain the following expression for the Fourier expansion of each function in the support of $\mathcal{D}_{\textsc{Accept}}$:

$$\begin{aligned} g_{\tau,\mathbf{s}}(x_1,\ldots,x_r,y_0,\ldots,y_{n-r-1}) = &\sum_{\mathbf{i}=i_1i_2\ldots i_r=0}^{R/2-1} y_{\tau(\mathbf{i})}\left(\frac{1+(-1)^{i_1}x_1}{2}\right)\left(\frac{1+(-1)^{i_2}x_2}{2}\right)\ldots\left(\frac{1+(-1)^{i_r}x_r}{2}\right) \\ &+ \sum_{\mathbf{i}=0}^{R/2-1} s_{\mathbf{i}}y_{\tau(\mathbf{i})}\left(\frac{1+(-1)^{\overline{i_1}}x_1}{2}\right)\left(\frac{1+(-1)^{\overline{i_2}}x_2}{2}\right)\ldots\left(\frac{1+(-1)^{\overline{i_r}}x_r}{2}\right) \end{aligned} \tag{5.3.5}$$

$$\begin{aligned} [\text{Since } (-1)^{\overline{i_j}} = -(-1)^{i_j}] \quad &= \frac{1}{2^{r-1}} \sum_{\mathbf{i}=0}^{R/2-1} \begin{cases} \displaystyle\sum_{X\subseteq\{x_1,\ldots,x_r\},|X| \text{ even}} (-1)^{\left(\sum_{x_j\in X} i_j\right)} y_{\tau(\mathbf{i})}\chi_X & \text{if } s_{\mathbf{i}} = 1; \\ \displaystyle\sum_{X\subseteq\{x_1,\ldots,x_r\},|X| \text{ odd}} (-1)^{\left(\sum_{x_j\in X} i_j\right)} y_{\tau(\mathbf{i})}\chi_X & \text{if } s_{\mathbf{i}} = -1. \end{cases} \end{aligned} \tag{5.3.6}$$

Just as in the Equation (5.3.2), whenever $\frac{1-x_1}{2} = i_1, \frac{1-x_2}{2} = i_2, \ldots, \frac{1-x_r}{2} = i_r$, the sum on the RHS of Equation (5.3.5) has precisely one non-zero term which is $y_{\tau(\mathbf{i})}$ if $\mathbf{i} < R/2$ and $s_{R-1-\mathbf{i}} y_{\tau(R-1-\mathbf{i})}$ if $\mathbf{i} \geq R/2$. Therefore this sum gives rise to exactly the Addressing function in Equation (5.3.4) which is defined as $g_{\tau,s}$ and consequently the equality in Equation (5.3.5) follows.

It follows that for each $g_{\tau,\mathbf{s}}$ in the support of $\mathcal{D}_{\text{ACCEPT}}$ and for any fixed y_j, all elements of the set $\{S\colon y_j \in S \text{ and } \widehat{g_{\tau,\mathbf{s}}}(S) \neq 0\}$ will have the same parity. Moreover, given a function drawn from $\mathcal{D}_{\text{ACCEPT}}$, for every distinct y_j this odd/even parity is independent and uniformly random.

Now we are ready to prove Theorem 5.3.5. Recall that a FS oracle query returns S with probability $|\hat{f}(S)|^2$ for every subset S of input variables to the function.

Let us define a set $\mathcal{T}$ of "typical" outcomes from FS oracle queries. Fix any $N = o(\sqrt{k})$, and let $\mathcal{T}$ denote the set of all sequences $\{(y_{j_1}, X_1), \ldots, (y_{j_N}, X_N)\}$ of length N which have the property that *no y_i occurs more than once among $y_{j_1}, \ldots, y_{j_N}$*.

Note that for any fixed $f_\tau \leftarrow \mathcal{D}_{\text{REJECT}}$, every non-zero Fourier coefficient $\widehat{f_\tau}(S)$ satisfies $|\widehat{f_\tau}(S)|^2 = \frac{1}{2^{2r}} = \frac{1}{R^2}$ due to Equation (5.3.3). Therefore after f_τ is drawn, for any fixed y_j the probability of receiving a response of the form (y_j, X) as the outcome of a FS query is either

$= 0$, if f_τ is not a function of y_j, i.e. $j \notin \{\tau(0), \ldots, \tau(R-1)\}$; or

$= \frac{1}{R}$, if $j \in \{\tau(0), \ldots, \tau(R-1)\}$. This is because each of the $2^r = R$ responses (y_j, X) occurs with probability $\frac{1}{R^2}$.

Similarly, for any fixed $g_{\tau,\mathbf{s}} \leftarrow \mathcal{D}_{\text{ACCEPT}}$, every non-zero Fourier coefficient $\widehat{g_{\tau,\mathbf{s}}}(S)$ satisfies $|\widehat{g_{\tau,\mathbf{s}}}(S)|^2 = \frac{1}{2^{2r-2}} = \frac{4}{R^2}$ due to Equation (5.3.6). Therefore after $g_{\tau,\mathbf{s}}$ is drawn, for any fixed y_j the probability of receiving a response of the form (y_j, X) as the outcome of a FS query is either

$= 0$, if $g_{\tau,\mathbf{s}}$ is not a function of y_j, i.e. $j \notin \{\tau(0), \ldots, \tau(R/2-1)\}$; or

$= \frac{2}{R}$, if $j \in \{\tau(0), \ldots, \tau(R/2-1)\}$. This is because each of the $2^{r-1} = R/2$ responses (y_j, X) occurs with probability $\frac{4}{R^2}$.

Now let us consider the probability of obtaining a sequence from $\mathcal{T}$ under each scenario.

- If the function is drawn from $\mathcal{D}_{\text{REJECT}}$: the probability is at least

$$1(1-1/R)(1-2/R)\ldots(1-N/R) > 1-o(1) \quad \text{[by the Birthday Paradox]}.$$

- If the function is from $\mathcal{D}_{\text{ACCEPT}}$: the probability is at least

$$1(1-2/R)(1-4/R)\ldots(1-2N/R) > 1-o(1) \quad \text{[by the Birthday Paradox]}$$

Now the crucial observation is that whether the function is drawn from $\mathcal{D}_{\text{REJECT}}$ or from $\mathcal{D}_{\text{ACCEPT}}$, each sequence in $\mathcal{T}$ is equiprobable by symmetry in the construction. To see this, simply consider the probability of receiving a fixed (y_j, X) for some new y_j in the next FS query of an unknown function drawn from either one of these distributions. Using the above calculations for $|\hat{f}(y_j, X)|^2$, one can directly calculate that these probabilities are equal in either scenario. Alternatively, for a function drawn from $\mathcal{D}_{\text{ACCEPT}}$ one can observe that since each successive y_j is "new", a fresh random bit determines whether the support is an (y_j, X) with $|X|$ odd or even; once this is determined, the choice of X is uniform from all subsets with the correct parity. Thus the overall draw of (y_j, X) is uniform over all X's. Considering that the subset of relevant variables $T, |T| = R/2$ is uniformly chosen from $\{y_0, \ldots, y_{n-r-1}\}$, this gives the equality of the probabilities for each (y_j, X) with a new y_j when the function is drawn from $\mathcal{D}_{\text{ACCEPT}}$. The argument for the case of $\mathcal{D}_{\text{REJECT}}$ is clear.

Consequently the statistical difference between the distributions corresponding to the sequence of outcomes of the N FS oracle calls under the two distributions is at most $o(1)$. Now Fact 5.2.4 implies that no algorithm making only N oracle calls can distinguish between these two scenarios with high probability. This gives us the result. □

Intuitively, under either distribution on functions, each element of a sequence of N FS oracle calls will "look like" a uniform random draw X from subsets of $\{x_1, \ldots, x_r\}$ and j from $\{0, \ldots, n-r-1\}$ where j and X are independent. Note that this argument breaks down at $N = \Theta(\sqrt{R})$. This is because if the algorithm queried the FS oracle $\omega(\sqrt{R})$ times it will start to see some y_i's more than once (again by the birthday paradox). But when the functions are drawn

from $\mathcal{D}_{\text{ACCEPT}}$ the corresponding X_i's will always have a fixed parity for a given y_i whereas for functions drawn from $\mathcal{D}_{\text{REJECT}}$ the parity will be random each time. This will provide the algorithm with sufficient evidence to distinguish w.h.p. between these two scenarios.

5.4 Learning juntas

5.4.1 Known results

The problem of learning an unknown k-junta has been well studied in the computational learning theory literature, see e.g. [MOS04, AR03, Blu03]. The following classical lower bound will be a yardstick against which we will measure our results.

Lemma 5.4.1. *Any classical membership query algorithm for learning k-juntas to accuracy $1/5$ must use $\Omega(2^k + \log n)$ membership queries.*

Proof. Consider the restricted problem of learning an unknown variable $x_1, \ldots, x_n$. Since any two variables disagree on half of all inputs, any $1/5$-learning algorithm can be easily modified into an algorithm that exactly learns an unknown variable with no more queries. It is well known that any set of n concepts required $\Omega(\log n)$ queries for any exact learning algorithm that uses membership queries only, see e.g. [BCG$^+$96]. This gives the $\Omega(\log n)$ lower bound.

For the $\Omega(2^k)$ lower bound, we may suppose that the the algorithm "knows" that the junta has relevant variables $x_1, \ldots, x_k$. Even in this case, if fewer than $\frac{1}{2}2^k$ membership queries are made the learner will have no information about at least $1/2$ of the function's output values. A straightforward application of the Chernoff bound shows that it is very unlikely for such a learner's hypothesis to be $1/5$-accurate, if the target junta is a uniform random function over the relevant variables. This establishes the result. □

Learning juntas from uniform random examples $\mathsf{EX}(f)$ is a seemingly difficult computational problem. Simple algorithms based on exhaustive search can learn from $O(2^k \log n)$ examples but require $\Omega(n^k)$ runtime. The fastest known algorithm in this setting, due to Mossel *et al.*, uses $(n^k)^{\frac{\omega}{\omega+1}}$ examples and runs in $(n^k)^{\frac{\omega}{\omega+1}}$ examples time, where $\omega < 2.376$ is the matrix multiplication exponent [MOS04].

Bshouty and Jackson [BJ99] gave an algorithm using uniform quantum examples from the QEX oracle to learn general DNF formulas. Their algorithm uses $\tilde{O}(ns^6\epsilon^{-8})$ calls to QEX to learn an s-term DNF over n variables to accuracy ϵ. Since any k-junta is expressible as a DNF with at most 2^{k-1} terms, their result immediately yields the following statement.

Theorem 5.4.2 (See [BJ99])**.** *There exists an ϵ-learning quantum algorithm for k-juntas using $\tilde{O}(n2^{6k}\epsilon^{-8})$ quantum examples under the uniform distribution quantum PAC model.*

Note that [BJ99] did not try to optimize the quantum query complexity of their algorithms in the special case of learning juntas. In contrast, our goal is to obtain a more efficient algorithm for juntas.

The lower bound of [AS05, Observation 6.3] for learning with quantum membership queries for an arbitrary concept class of can be rephrased for the purpose of learning k-juntas as follows.

Fact 5.4.3 (See [AS05])**.** *Any algorithm for learning k-juntas to accuracy $\epsilon = 1/10$ with quantum membership queries must use $\Omega(2^k)$ queries.*

Proof. Since we are proving a lower bound we may assume that the algorithm is told in advance that the junta depends on variables $x_1, \ldots, x_k$. Consequently we may assume that the algorithm makes all its queries with nonzero amplitude only on inputs of the form $|x, 1^{n-k}\rangle$. Now [AS05, Observation 6.3] states that any quantum algorithm which makes queries only over a shattered set (as is the set of inputs $\{|x, 1^{n-k}\rangle\}_{x\in\{-1,1\}^k}$ for the class of k-juntas) must make at least VC-DIM(C)/100 QMQ queries to learn with error rate at most $\epsilon = 1/10$; here VC-DIM(C) is the Vapnik-Chervonenkis dimension of concept class C. Since the VC dimension of the class of all Boolean functions over variables $x_1, \ldots, x_k$ is 2^k, the result follows. □

This shows that a QMQ oracle cannot provide sufficient information to learn a k-junta using $o(2^k)$ queries to high accuracy. It is worth noting that there are other similar learning problems known where an N-query QMQ algorithm can exactly identify a target concept whose description length is $\omega(N)$ bits. For instance, a single FS oracle call (which can be implemented by a single QMQ query) can potentially give up to k bits of information; if the concept class C is the class of all 2^k parity functions over the first k variables, then any concept in the class can be exactly learned by a single FS oracle call.

Note that all the results we have discussed in this subsection concern algorithms with access to only one type of oracle; this is in contrast with the algorithm we present in the next section.

5.4.2 A new learning algorithm

The motivating question for this section is: "Is it possible to reduce the classical query/sample complexity drastically for the problem of junta learning if the learning algorithm is also permitted to have very limited quantum information?" We will give an affirmative answer to this question by describing a new algorithm that uses both FS queries (i.e. quantum examples) and classical uniform random examples.

Lemma 5.4.4. *Let $f\colon \{-1,1\}^n \to \{-1,1\}$ be a function whose value depends on the set of variables $\mathcal{I}$. Then there is an algorithm querying the* FS *oracle $O(\epsilon^{-1}\log|\mathcal{I}|)$ times which w.h.p. outputs a list of variables such that*

- *the list contains all the variables x_i for which $\mathrm{Inf}_i(f) \geq \epsilon$; and*
- *all the variables x_j in the list have non-zero influence: $\mathrm{Inf}_j(f) > 0$.*

Proof. The algorithm simply queries the FS oracle $N = O(\epsilon^{-1}\log|\mathcal{I}|)$ many times and outputs the union of all the sets of variables received as responses to these queries.

If $\mathrm{Inf}_i(f) \geq \epsilon$ then the probability that x_i never occurs in any response obtained from the N FS oracle calls is at most $(1-\epsilon)^N \leq \frac{1}{10|\mathcal{I}|}$. The union bound now yields that with probability at least $9/10$, every x_i with $\mathrm{Inf}_i(f) \geq \epsilon$ is output by the algorithm. □

Theorem 5.4.5. *There is an efficient algorithm ϵ-learning k-juntas with $O(\epsilon^{-1}k\log k)$ queries of the* FS *oracle and $O(\log(\epsilon^{-1})2^k)$ random examples.*

Proof. We claim Algorithm 12 satisfies these requirements.

Assume we are given a Boolean function f whose value depends on the set of variables $\mathcal{I}$ with $|\mathcal{I}| \leq k$. By Lemma 5.4.4, $O(\epsilon^{-1}k\log k)$ queries of the FS oracle will reveal all variables with influence at least $(\epsilon/10k)$ with high probability during Stage 1.

Assuming the algorithm of Lemma 5.4.4 was successful, we group the variables as follows:

Algorithm 12 The junta learning algorithm.

Input: $\epsilon > 0, \mathsf{FS}(f), \mathsf{EX}(f)$.
Stage 1:
Construct a set containing all variables of f with an influence at least $(\epsilon/10k)$ using the algorithm in Lemma 5.4.4. Let $\mathcal{A}$ be the final result.
$\forall \mathbf{a} \in \{-1,1\}^{|\mathcal{A}|}, encountered(\mathbf{a}) \leftarrow$ FALSE.
Stage 2:
repeat
  $\langle x, f(x)\rangle \leftarrow$ Draw from $\mathsf{EX}(f)$. Let $x|_{\mathcal{A}}$ denote the projection of x onto the variables in $\mathcal{A}$.
  if $encountered(x|_{\mathcal{A}}) =$ FALSE **then**
    $value(x|_{\mathcal{A}}) \leftarrow f(x), encountered(x|_{\mathcal{A}}) \leftarrow$ TRUE.
  end if
until For at least $(1-\epsilon/3)$ fraction of all $\mathbf{a} \in \{-1,1\}^{|\mathcal{A}|}, encountered(\mathbf{a}) =$ TRUE.
Output the hypothesis:

$$H(x) = \begin{cases} value(x|_{\mathcal{A}}) & \text{if } encountered(x|_{\mathcal{A}}) = \text{TRUE} \\ \text{TRUE} & \text{otherwise.} \end{cases}$$

Group	Description
$\mathcal{A}$	The set of variables encountered in Stage 1.
$\mathcal{B}$	The set of relevant variables $\mathcal{I} \setminus \mathcal{A}$.
$\mathcal{C}$	The remaining $n - \lvert\mathcal{I}\rvert$ variables the function does not depend on.

Note that $|\mathcal{A}| + |\mathcal{B}| \leq k$ by Lemma 5.4.4 and by the assumption that f is a k-junta.

We reorder the variables of f so that the new order is $\mathcal{A}, \mathcal{B}, \mathcal{C}$ for notational simplicity, i.e. f is now considered to be over $(a_1, \ldots, a_{|\mathcal{A}|}, b_1, \ldots, b_{|\mathcal{B}|}, c_1, \ldots, c_{|\mathcal{C}|})$. We will denote an assignment to these variables by $(\mathbf{a}, \mathbf{b}, \mathbf{c})$.

In Stage 2 the algorithm draws random examples until at least $(1-\epsilon/3)$ fraction of all assignments to the variables in $\mathcal{A}$ are observed. Let us call this set of assignments by $\mathcal{S}$, and for every $\mathbf{a} \in \mathcal{S}$, let us denote the first example $\langle x, f(x)\rangle$ drawn in Stage 2 for which $x|_{\mathcal{A}} = \mathbf{a}$ by $x = (\mathbf{a}, \mathbf{b}^{\mathbf{a}}, \mathbf{c}^{\mathbf{a}})$. At the end of the algorithm, the following hypothesis is produced as the output:

$$H(\mathbf{a}, *, *) = \begin{cases} f(\mathbf{a}, \mathbf{b}^{\mathbf{a}}, \mathbf{c}^{\mathbf{a}}) & \text{if } \mathbf{a} \in \mathcal{S} \\ \text{TRUE} & \text{otherwise.} \end{cases}$$

In other words, the value of the hypothesis only depends on the setting of the variables in $\mathcal{A}$. Ob-

serve the probability that any given setting of a fixed set of variables in $\mathcal{A}$ has not been seen can be made less than $\epsilon/50$ using $O(\log(\epsilon^{-1})2^k)$ uniform random examples. Therefore the linearity of expectation implies that after $O(\log(\epsilon^{-1})2^k)$ random examples, the expected fraction of unseen assignments is $< \epsilon/50$. Thus by Markov's Inequality the fraction of unseen assignments will be $\leq \epsilon/3$ w.h.p. Hence Stage 2 will terminate w.h.p. after $O(\log(\epsilon^{-1})2^k)$ random examples. Consequently, the whole algorithm terminates with high probability with the desired query consumption. All we need to verify is that the hypothesis constructed is ϵ-accurate.

The hypothesis H is ϵ-accurate with high probability:

We introduce some notation: Let $\mathbb{B} = \{-1, 1\}$; and given two strings $u, v \in \mathbb{B}^\ell$, let $u \odot v$ denote the bitwise multiplication between u, v; and let $|u|$ denote the total number of -1's in u. Also let $\mathbf{1}_W$ denote the indicator function that takes value 1 if W holds and value 0 if W is false.

We start with the following fact:

Fact 5.4.6. *For any $s \in \mathbb{B}^{|\mathcal{B}|}$, we have* $\frac{1}{2^n} \sum_{\mathbf{a}\in\mathbb{B}^{|\mathcal{A}|}} \sum_{\mathbf{b}\in\mathbb{B}^{|\mathcal{B}|}} \sum_{\mathbf{c}\in\mathbb{B}^{|\mathcal{C}|}} \mathbf{1}_{[f(\mathbf{a},\mathbf{b}\odot s,\mathbf{c})\neq f(\mathbf{a},\mathbf{b},\mathbf{c})]} < \epsilon/10.$

Proof. Given any string $s \in \mathbb{B}^{|\mathcal{B}|}$, clearly there exists a sequence of $|s| + 1$ strings:

$$1^{|\mathcal{B}|} = u^1, u^2, \ldots, u^{|s|+1} = s, \text{ where } u^i \in \mathbb{B}^{|\mathcal{B}|}, \text{ and for } i = 1, \ldots, s, \ |u^i \odot u^{i+1}| = 1.$$

Therefore,

$$\begin{aligned}
\text{For any } s \in \mathbb{B}^{|\mathcal{B}|}, \quad & \frac{1}{2^n} \sum_{\mathbf{a}\in\mathbb{B}^{|\mathcal{A}|}} \sum_{\mathbf{b}\in\mathbb{B}^{|\mathcal{B}|}} \sum_{\mathbf{c}\in\mathbb{B}^{|\mathcal{C}|}} \mathbf{1}_{[f(\mathbf{a},\mathbf{b}\odot s,\mathbf{c})\neq f(\mathbf{a},\mathbf{b},\mathbf{c})]} \\
& \leq \frac{1}{2^n} \sum_{\mathbf{a}\in\mathbb{B}^{|\mathcal{A}|}} \sum_{\mathbf{b}\in\mathbb{B}^{|\mathcal{B}|}} \sum_{\mathbf{c}\in\mathbb{B}^{|\mathcal{C}|}} \sum_{i=1}^{|s|} \mathbf{1}_{[f(\mathbf{a},\mathbf{b}\odot u^{i+1},\mathbf{c})\neq f(\mathbf{a},\mathbf{b}\odot u^i,\mathbf{c})]} \\
& = \sum_{i=1}^{|s|} \underbrace{\left(\frac{1}{2^n} \sum_{\mathbf{a}\in\mathbb{B}^{|\mathcal{A}|}} \sum_{\mathbf{b}\in\mathbb{B}^{|\mathcal{B}|}} \sum_{\mathbf{c}\in\mathbb{B}^{|\mathcal{C}|}} \mathbf{1}_{[f(\mathbf{a},\mathbf{b}\odot u^i\odot u^{i+1},\mathbf{c})\neq f(\mathbf{a},\mathbf{b},\mathbf{c})]} \right)}_{=\text{The influence of the unique variable } b_{j(i)} \text{ that takes value } -1 \text{ in } u^{i+1} \odot u^i} \\
& < \epsilon/10. \quad [\text{Since every } b_j \in \mathcal{B} \text{ has influence} < \frac{\epsilon}{10k} \text{ and } |\mathcal{B}| \leq k]
\end{aligned}$$

□

For each $\mathbf{a} \in \mathbb{B}^{|\mathcal{A}|}$, consider a fixed setting of strings $\mathbf{b^a} \in \mathbb{B}^{|\mathcal{B}|}$, $\mathbf{c^a} \in \mathbb{B}^{|\mathcal{C}|}$. Let us call the list of all these assignments Γ, i.e. $\Gamma = \{\forall \mathbf{a} \in \mathbb{B}^{|\mathcal{A}|}, (\mathbf{a}, \mathbf{b^a}, \mathbf{c^a})\}$. For any such "list of assignments" Γ, we define the function $F_\Gamma \colon \{-1,1\}^n \to \{-1,1\}$ as follows: $F_\Gamma(\mathbf{a}, *, *) = f(\mathbf{a}, \mathbf{b^a}, \mathbf{c^a})$. The error incurred by approximating f by F_Γ is:

$$\mathbf{Pr}_{(\mathbf{a},\mathbf{b},\mathbf{c})}[F_\Gamma(\mathbf{a},\mathbf{b},\mathbf{c}) \neq f(\mathbf{a},\mathbf{b},\mathbf{c})] = \mathbf{Pr}_{(\mathbf{a},\mathbf{b},\mathbf{c})}[f(\mathbf{a},\mathbf{b^a},\mathbf{c^a}) \neq f(\mathbf{a},\mathbf{b},\mathbf{c})]$$

$$= \mathbf{Pr}_{(\mathbf{a},\mathbf{b},\mathbf{c})}[f(\mathbf{a},\mathbf{b^a},\mathbf{c}) \neq f(\mathbf{a},\mathbf{b},\mathbf{c})] \quad [\text{Since } f \text{ does not depend on the variables in } \mathcal{C}]$$

$$= \frac{1}{2^n} \sum_{\mathbf{a}\in\mathbb{B}^{|\mathcal{A}|}} \sum_{\mathbf{b}\in\mathbb{B}^{|\mathcal{B}|}} \sum_{\mathbf{c}\in\mathbb{B}^{|\mathcal{C}|}} \mathbf{1}_{[f(\mathbf{a},\mathbf{b^a},\mathbf{c}) \neq f(\mathbf{a},\mathbf{b},\mathbf{c})]} = \frac{1}{2^n} \sum_{\mathbf{a}\in\mathbb{B}^{|\mathcal{A}|}} \sum_{s\in\mathbb{B}^{|\mathcal{B}|}} \sum_{\mathbf{c}\in\mathbb{B}^{|\mathcal{C}|}} \mathbf{1}_{[f(\mathbf{a},\mathbf{b^a},\mathbf{c}) \neq f(\mathbf{a},\mathbf{b^a}\odot s,\mathbf{c})]} \tag{5.4.1}$$

Therefore if we consider the expected value of the incurred error $\mathbf{Pr}[F_\Gamma \neq f]$ over all "lists of assignments" Γ, equation (5.4.1) implies that:

$$\mathbf{E}_\Gamma[\mathbf{Pr}_{(\mathbf{a},\mathbf{b},\mathbf{c})}[F_\Gamma \neq f]] = \frac{1}{2^{|\mathcal{B}|}} \sum_{s\in\mathbb{B}^{|\mathcal{B}|}} \underbrace{\left(\frac{1}{2^n} \sum_{\mathbf{a}\in\mathbb{B}^{|\mathcal{A}|}} \sum_{\mathbf{b^a}\in\mathbb{B}^{|\mathcal{B}|}} \sum_{\mathbf{c}\in\mathbb{B}^{|\mathcal{C}|}} \mathbf{1}_{[f(\mathbf{a},\mathbf{b^a}\odot s,\mathbf{c}) \neq f(\mathbf{a},\mathbf{b^a},\mathbf{c})]} \right)}_{< \epsilon/10, \text{ due to Fact 5.4.6}}$$

$$< \epsilon/10.$$

Consequently, the expected error of approximating f by a uniformly chosen F_Γ is less than $\epsilon/10$. This also implies that for a uniformly chosen subset $\mathcal{S}$ of assignments to variables in $\mathcal{A}$ with size $(1-\epsilon/3)2^{|\mathcal{A}|}$, the expected error over $\mathcal{S}$ satisfies: $\mathbf{E}_\Gamma[\mathbf{Pr}_{\substack{(\mathbf{a},\mathbf{b},\mathbf{c}) \\ \mathbf{a}\in\mathcal{S}}}[F_\Gamma \neq f]] < \epsilon/10$. Therefore by Markov's Inequality, we obtain the following observation:

Observation 5.4.7. *For a uniformly chosen subset $\mathcal{S}$ and F_Γ as described above, F_Γ will agree with f on $(1-\epsilon/3)$ fraction of the coordinates $\{(\mathbf{a},\mathbf{b},\mathbf{c}), \mathbf{a} \in \mathcal{S}\}$ with probability at least $7/10$.*

Now if we go back and recall what the algorithm does in Stage 2, we will observe that the generation of the hypothesis in Stage 2 is equivalent to drawing a uniform F_Γ and $\mathcal{S}$ as described and resetting the values of F_Γ at those coordinates $\{(\mathbf{a},\mathbf{b},\mathbf{c}), \mathbf{a} \notin \mathcal{S}\}$ to TRUE. This is because the algorithm only draws classical random examples during Stage 2. Therefore due to Observation 5.4.7,

the hypothesis will disagree with f on at most

$$\underbrace{1-(1-\epsilon/3)^2}_{\text{The error incurred by } (\mathbf{a},\mathbf{b},\mathbf{c}),\mathbf{a}\in\mathcal{S}} + \underbrace{\epsilon/3}_{\text{The error incurred by } (\mathbf{a},\mathbf{b},\mathbf{c}),\mathbf{a}\notin\mathcal{S}} < \epsilon$$

fraction of the inputs with overall probability at least $2/3$. This gives the desired result. □

Note that this algorithm offers the following benefits:

- is valid in the quantum PAC learning model, which is weaker than learning with QMQ queries.
- has no dependence on n and thus goes beyond what is possible with classical MQ learning due to the lower bound of Lemma 5.4.1.
- has access to limited quantum information – just $O(\mathrm{poly}(k))$ quantum examples.
- has access to EX oracle as its only source of classical information – MQ queries are not allowed.

Moreover, one can compare this result to that of Theorem 5.4.2 which requires $\tilde{O}(n2^{6k}\epsilon^{-8})$ quantum examples to learn k-juntas. In contrast, our algorithm uses not only substantially fewer quantum examples but also fewer uniform random examples, which are considered quite cheap. Intuitively, this means that for the junta learning problem, almost all the quantum queries used by the algorithm of Bshouty and Jackson [BJ99] can in fact be converted into ordinary classical random examples.

Lower bounds

The algorithm of Theorem 5.4.5 is optimal in the following sense:

Observation 5.4.8. *Any $1/10$-learning quantum algorithm for k-juntas with $\frac{1}{120}2^k$ random examples (or even classical* MQ *queries) requires $\Omega(2^k)$* QMQ *queries.*

Proof. This statement easily follows from Fact 5.4.3 since a classical random example and a classical membership query can be simulated by a QMQ query. □

In contrast to Observation 5.4.8, (for a constant value of ϵ) Algorithm 12 uses the same order of random examples but about exponentially fewer quantum queries: $O(k \log k)$. In other words, for sufficiently small values of ϵ if the allowed number of classical random examples are dropped slightly from $O(2^k \log \epsilon^{-1})$ to $\frac{1}{120}2^k$ then every quantum learning algorithm additionally requires $\Omega(2^k)$ queries even if it is allowed QMQ oracle access. Note that Algorithm 12 of Theorem 5.4.5 queries the uniform quantum example oracle, a weaker source of information than the quantum membership oracle.

5.5 Learning polynomially sparse functions with a FS oracle and random examples

We use a similar terminology to that of [KM93] for the concept class.

Definition 5.5.1 (See [KM93])**.**

- A function $f : \{-1,1\}^n \to \mathbb{R}$ is said to be *t-sparse* if its Fourier spectrum contains only t non-zero frequencies. Note that sparse functions in this definition do not have to be Boolean valued.
- A function $f : \{-1,1\}^n \to \{-1,1\}$ is said to be *polynomially $t(\epsilon)$-sparse* if
  - for every $\epsilon > 0$ there exists a $t(\epsilon)$-sparse function $g : \{-1,1\}^n \to \mathbb{R}$ such that $\mathbf{E}_x[(f(x) - g(x))^2] \leq \epsilon$; and
  - $t(\epsilon)$ is a polynomial in ϵ^{-1} with coefficients possibly depending on n.

  When we refer to an polynomially $t(\epsilon)$-sparse function, it will be implicit that t is a polynomial in ϵ^{-1} and we will simply write t instead of $t(\epsilon)$.

5.5.1 Known bounds on learning polynomially sparse functions

As shown by [KM93], some important and natural classes of functions such as decision trees and functions with bounded L_1 norm are polynomially sparse. Thus polynomially sparse functions are of considerable interest from a learning theory perspective.

Algorithms that learn sparse functions using classical membership queries must make at least $\Omega(n)$ queries:

Lemma 5.5.2. *Any algorithm for learning t-sparse Boolean functions to accuracy $\epsilon = 1/10$ with classical membership queries must make $\Omega(n\sqrt{t} + t)$ queries.*

Proof. The $\Omega(t)$ lower bound follows from Observation 5.5.4 below.

For the $\Omega(n\sqrt{t})$ lower bound, let $r = \frac{1}{2}\log t$ and let $R = n - r$. We consider the concept class C of "junta selected parities;" these are functions where each setting of the first r variables determines a parity function over the final R variables, whose value is the output of the function. Formally, we have that

$$C = \{f(x_1, \ldots, x_r, y_1, \ldots, y_R) : f(x_1, \ldots, x_r, y_1, \ldots, y_R) = \chi_{S_x}(y_1, \ldots, y_R)\}$$

so each $f \in C$ is determined by a sequence of 2^r parities $\{S_x\}_{x \in \{-1,1\}^r}$ each of which may be an arbitrary subset of $\{y_1, \ldots, y_R\}$.

Note that each $f \in C$ can be represented as a decision tree of depth $r = \frac{1}{2}\log t$ with a parity function at each leaf. Each root-to-leaf path in such a decision tree contributes at most $\sqrt{t}$ nonzero terms to the Fourier representation of f, and there are $\sqrt{t}$ leaves; thus every $f \in C$ is t-sparse.

Let us consider the problem of learning one of the parity functions at a given fixed leaf of the above described decision tree for f. Any two distinct parity functions agree on precisely half of all inputs; if fewer than $n - r$ membership queries have been made on inputs that reach that leaf, then there will be at least two consistent parity functions which could be labeling the leaf, and no hypothesis can have accuracy greater than $3/4$ with probability exceeding $1/2$. Hence in order to learn a given function in C to accuracy $1 - \epsilon$, the algorithm will require at least $n - r$ membership queries for at least (say) a $1 - 8\epsilon$ fraction of all $2^r = \sqrt{t}$ many leaves of the tree. This gives a query complexity of $\Omega((n - r)\sqrt{t})$ for any successful algorithm. □

Note that this lower bound has a linear dependence over n.

The following result due to Kushilevitz and Mansour gave the first polynomial time algorithm for learning decision trees given classical membership queries:

Theorem 5.5.3 (See [KM93]). *There is an efficient classical algorithm learning polynomially t-sparse functions using $O(\mathrm{poly}(n, t, \epsilon^{-1}, \log \delta^{-1}))$ membership queries.*

We have the following lower bound on learning sparse functions even when quantum membership queries are allowed:

Observation 5.5.4. *Any algorithm for learning t-sparse Boolean functions to accuracy $\epsilon = 1/10$ with quantum membership queries must use $\Omega(t)$ queries.*

This follows from Fact 5.4.3 and the easy observation that every $(\log t)$-junta is a t-sparse function.

The QEX based DNF learning algorithm of Bshouty and Jackson can also be used for learning sparse functions.

Theorem 5.5.5 (Implicit in [BJ99]). *There exists a ϵ-learning quantum algorithm for Boolean valued t-sparse functions using $\tilde{O}(n^7 t^6 \epsilon^{-8})$ quantum examples under the uniform distribution quantum PAC model.*

Proof. Observe that by Cauchy-Schwarz inequality, any t-sparse function has an L_1 norm of at most $\sqrt{t}$. Now we can invoke a result due to [BS92] stating that any Boolean function over n variables with L_1 norm ℓ has a PTF of weight at most $O(n\ell^2)$. Moreover by [BJ99], we know any PTF of weight W over n variables can be learned using $\tilde{O}(nW^6\epsilon^{-8})$ uniform quantum examples. This gives the desired result. □

5.5.2 A new learning algorithm

Our goal is to obtain a quantum algorithm which is not only more efficient but can also learn polynomially sparse functions.

The following lemma which is a slightly modified version of [KM93, Lemma 3.1] will play an important role in our derivations.

Lemma 5.5.6. *If $f : \{-1,1\}^n \to \{-1,1\}$ is a polynomially t-sparse function then for every $\epsilon > 0$ there exists a function $h_\epsilon : \{-1,1\}^n \to \mathbf{R}$ such that*

- *$h_\epsilon(x)$ is $t(\frac{\epsilon}{2})$-sparse,*

- *For every $S \subseteq [n]$ such that $\hat{h_\epsilon}(S) \neq 0$, $|\hat{h_\epsilon}(S)| > (\frac{2}{\epsilon} t(\frac{\epsilon}{2}))^{-1/2}$ and $\hat{h_\epsilon}(S) = \hat{f}(S)$.*
- $\mathbf{E}_x[(f(x) - h_\epsilon(x))^2] \leq \epsilon$.

Proof. Since f is a polynomially $t(\epsilon)$-sparse function, for every $\epsilon > 0$ there exists a $t(\frac{\epsilon}{2})$-sparse function g_ϵ satisfying $\mathbf{E}_x[(f(x) - g_\epsilon(x))^2] \leq \epsilon/2$. For any given ϵ, let's fix the function g_ϵ.

Let $\Gamma = \{S\colon \hat{g_\epsilon}(S) \neq 0\}$ and $\Gamma' = \Gamma \cap \{S\colon |\hat{f}(S)| > (\frac{2}{\epsilon} t(\frac{\epsilon}{2}))^{-1/2}\}$. Clearly $|\Gamma'| \leq |\Gamma| \leq t(\frac{\epsilon}{2})$.

Define h_ϵ as follows:

$$h_\epsilon(x) = \sum_{S \in \Gamma'} \hat{f}(S) \chi_S(x).$$

$h_\epsilon(x)$ satisfies the first two properties by construction. Moreover by Parseval's identity we have:

$$\mathbf{E}_x[(f(x) - h_\epsilon(x))^2] = \sum_{S \notin \Gamma} |\hat{f}(S)|^2 + \sum_{S \in \Gamma \setminus \Gamma'} |\hat{f}(S)|^2 \leq \epsilon/2 + \underbrace{|\Gamma \setminus \Gamma'|}_{\leq t(\frac{\epsilon}{2})} \cdot \underbrace{\max_{S \in \Gamma \setminus \Gamma'} |\hat{f}(S)|^2}_{\leq (\frac{2}{\epsilon} t(\frac{\epsilon}{2}))^{-1}} \leq \epsilon.$$

The first term is at most $\epsilon/2$ because the sum is over all the terms S in the spectrum which are not in the support of g_ϵ and g_ϵ satisfies $\mathbf{E}_x[(f(x) - g_\epsilon(x))^2] \leq \epsilon/2$. Observe that since for every $S \in \Gamma \setminus \Gamma'$, $|\hat{f}(S)| \leq (\frac{2}{\epsilon} t(\frac{\epsilon}{2}))^{-1/2}$ and $|\Gamma \setminus \Gamma'| \leq t(\frac{\epsilon}{2})$, the second term is also at most $\epsilon/2$. This gives the desired result. □

Our learning algorithm is similar to the junta learning algorithm of Section 5.4 in the sense that it works by first locating the heavy Fourier frequencies using FS oracle queries followed by random examples and classical computation. However, in contrast to the junta learning algorithm the algorithm uses classical random examples to approximate the heavy Fourier coefficients.

Our algorithm is based on the following lemma.

Lemma 5.5.7. *Let f be a polynomially t-sparse function, and let $h_\epsilon(x)$ be the function defined in Lemma 5.5.6. Then there is an algorithm that queries the FS oracle $O(\frac{t}{\epsilon} \log \frac{t}{\epsilon})$ many times and outputs a set $\mathcal{S}$ with $|\mathcal{S}| \leq O(\frac{t}{\epsilon} \log \frac{t}{\epsilon})$ containing all nonzero Fourier frequencies of h_ϵ with high probability.*

Proof. Consider the algorithm that, given $\epsilon > 0$ and access to $\mathsf{FS}(f)$, queries the $\mathsf{FS}(f)$ oracle N times and outputs a list $\mathcal{S}$ of all FS oracle call results that are obtained.

Due to the second property of construction in Lemma 5.5.6 for every $S \subseteq [n]$ such that $\widehat{h_\epsilon}(S) \neq 0$, we have $|\widehat{h_\epsilon}(S)| > (\frac{2}{\epsilon}t(\frac{\epsilon}{2}))^{-1/2}$ and $\widehat{h_\epsilon}(S) = \hat{f}(S)$. Thus for any such fixed coefficient S, the probability of not drawing S after N calls to the FS oracle is at most $(1 - \frac{\epsilon}{2t(\frac{\epsilon}{2})})^N$. This probability can be made smaller than $(10t(\frac{\epsilon}{2}))^{-1}$ for a sufficiently large choice of the constant c where $N = c\frac{t(\epsilon/2)}{\epsilon}\log\frac{t(\epsilon/2)}{\epsilon}$. Since $h_\epsilon(x)$ is $t(\frac{\epsilon}{2})$-sparse, the union bound gives the result. □

Finally the following theorem gives the polynomially sparse function learning algorithm. As earlier, our algorithm uses FS oracle queries and the rest of the computation is classical.

Theorem 5.5.8. *There is an efficient algorithm that ϵ-learns any polynomially t-sparse function using $O(\frac{t}{\epsilon}\log\frac{t}{\epsilon})$ calls to the* FS *oracle and $O(\frac{t}{\epsilon}\log\frac{t}{\epsilon})$ random examples.*

Proof. We claim Algorithm 13 satisfies these requirements.

Algorithm 13 The polynomially sparse function learning algorithm.

1: **Input:** $\epsilon > 0, \mathsf{FS}(f), \mathsf{EX}(f)$.
2: **Stage 1:**
3: Construct a set containing all nonzero Fourier frequencies of $h_{\epsilon/2}$ (as defined in Lemma 5.5.6) using the algorithm in Lemma 5.5.7. Let $\mathcal{S}$ be the final result.
4: **Stage 2:**
5: **for** $i = 0$ to $N = O(\frac{t}{\epsilon}\log\frac{t}{\epsilon})$ **do**
6: $\quad\langle x_i, f(x_i)\rangle \leftarrow$ Draw from $\mathsf{EX}(f)$.
7: **end for**
8: For every $S \in \mathcal{S}$, $c_S \leftarrow \frac{1}{N}\sum_{j=1}^{N} f(x_j)\chi_S(x_j)$ (Each c_S approximates $\hat{f}(S) = \widehat{h_{\epsilon/2}}(S)$).
9: Output the hypothesis $H(x) = \mathrm{sgn}(\sum_{S\in\mathcal{S}} c_S\chi_S)$.

Recall $t(\epsilon/4) = O(t(\epsilon))$ by the definition of a polynomially t-sparse function. In the first stage the algorithm calculates with high probability all nonzero frequencies $\mathcal{S}$ of $h_{\epsilon/4}$ using the algorithm in Lemma 5.5.7. Thus $O(\frac{t}{\epsilon}\log\frac{t}{\epsilon})$ queries of the FS oracle is made.

In the second stage for each $S \in \mathcal{S}$, the algorithm attempts to approximate each $\widehat{h_{\epsilon/2}}(S) = \hat{f}(S)$ to within $(\frac{4}{\epsilon}t(\frac{\epsilon}{4}))^{-1/2}$. Thus the additive Chernoff bound shows using $N = O(\frac{t}{\epsilon}\log\frac{t}{\epsilon})$ random examples it is possible to approximate every $\widehat{h_{\epsilon/2}}(S)$ to the desired accuracy with high probability.

Finally by virtue of the last property of the construction in Lemma 5.5.6,

$$\mathbf{E}_x[(f(x) - h_{\epsilon/2}(x))^2] \leq \epsilon/2 \Rightarrow [\text{By Parseval's identity}] \sum_{S \notin \mathcal{S}} |\hat{f}(S)|^2 \leq \epsilon/2, \text{ therefore}$$

$$\mathbf{E}_x[(f(x) - (\sum_{S \in \mathcal{S}} c_S \chi_S(x)))^2] = [\text{By Parseval's identity}] \underbrace{\sum_{S \notin \mathcal{S}} |\hat{f}(S)|^2}_{\leq \epsilon/2} + \underbrace{\sum_{S \in \mathcal{S}} |c_S - \hat{f}(S)|^2}_{\leq t(\frac{\epsilon}{4}) \cdot (\frac{4}{\epsilon} t(\frac{\epsilon}{4}))^{-1}} \leq \epsilon$$

$$\Rightarrow [\text{By Fact 5.2.3}]\ \mathbf{Pr}_x[f(x) \neq (H(x) = \text{sgn}(\sum_{S \in \mathcal{S}} c_S \chi_S))] \leq \epsilon.$$

This gives the desired result. □

Note that this algorithm offers the following benefits:

- is valid in the quantum PAC learning model, which is weaker than learning with QMQ queries.
- has no dependence on n and thus goes beyond what is possible with classical MQ learning due to the lower bound of Lemma 5.5.2.
- is close to optimal considering the lower bound given by Observation 5.5.4.
- has access to EX oracle as its only source of classical information – MQ queries are not allowed.

In contrast to the algorithm of Theorem 5.5.5, our algorithm uses substantially fewer quantum examples to learn Boolean valued sparse functions and can also learn polynomially sparse functions.

Chapter 6

Conclusions

We studied the capabilities and limitations of quantum algorithms for a wide variety of problems under several different models. In particular, in Chapter 3 we considered the information theoretic requirements of exact learning, partition learning and PAC learning of arbitrary concept classes; in Chapter 4 we considered the problem of learning $\omega(\log n)$ unions of high dimensional rectangles efficiently in the case where both n and $\log b$ are viewed as "large"; and finally in Chapter 5 we considered the problems of learning and testing juntas as well as learning polynomially sparse functions.

We believe the following natural questions arising from our results offer promising directions for future study (grouped with respect to the chapter of relevance):

Chapter 3: For the quantum exact learning model, is it possible to get rid of the $\log\log|C|$ factor in our algorithm's upper bound and thus prove the conjecture of Hunziker *et al.* [HMP$^+$03] exactly? For the partitions problem, can we extend the range of partition sizes (as a function of $|C|$) for which there can be a superpolynomial separation between the quantum and classical query complexity of learning the partition? Finally, for the PAC learning model, a natural goal is to strengthen our $\Omega(\frac{\sqrt{d}}{\epsilon})$ lower bound on sample complexity to $\Omega(\frac{d}{\epsilon})$ and thus match the lower bound of [EHKV89] for classical PAC learning.

Chapter 4: Besides the obvious goal of improving our positive results, it would be interesting to explore the limitations of current techniques for learning unions of rectangles over $[b]^n$. At this point we can't rule out the possibility that the GHS algorithm is in fact a poly$(n, s, \log b)$

time algorithm for learning unions of s arbitrary rectangles over $[b]^n$. Can evidence for or against this possibility be given? For example, can one show that the representational power of the hypotheses which the GHS algorithm produces (when run for $\mathrm{poly}(n, s, \log b)$ many stages) is – or is not – sufficient to express high-accuracy approximators to arbitrary unions of s rectangles over $[b]^n$?

Chapter 5:

- Does there exist any $o(k)$ junta testing algorithm using FS oracle queries?
- Is it possible to obtain a $\omega(\mathrm{poly}(\log k))$ lower bound for quantum algorithms testing the property of being a k-junta using quantum examples (or quantum membership queries)?
- Can we reduce the quantum query complexity of testing any further by allowing access to classical random examples ?
- What are the implications of "having access to a FS oracle" for testing the property of being a sparse function? More generally, what is the quantum complexity of testing this property using QEX oracle or QMQ oracle?

Bibliography

[AB97] Martin Anthony and Norman Biggs, *Computational learning theory*, Cambridge Tracts in Theoretical Computer Science, vol. 30, Cambridge University Press, Cambridge, UK, 1997, An introduction, Corrected reprint of the 1992 original.

[ABK+98] Howard Aizenstein, Avrim Blum, Roni Khardon, Eyal Kushilevitz, Leonard Pitt, and Dan Roth, *On learning read-k-satisfy-j DNF*, SIAM J. Comput. **27** (1998), no. 6, 1515–1530.

[AGS03] Adi Akavia, Shafi Goldwasser, and Samuel Safra, *Proving hard-core predicates using list decoding*, FOCS '03: Proc. of the 44th annual IEEE symposium on foundations of computer science, IEEE Comput. Soc. Press, 2003, pp. 146–156.

[AIK+04a] Andris Ambainis, Kazuo Iwama, Akinori Kawachi, Hiroyuki Masuda, Raymond H. Putra, and Shigeru Yamashita, *Quantum identification of Boolean oracles*, Proc. of STACS 2004, Lecture Notes in Comput. Sci., vol. 2996, Springer, Berlin, 2004, pp. 105–116.

[AIK+04b] Andris Ambainis, Kazuo Iwama, Akinori Kawachi, Rudy Raymond, and Shigeru Yamashita, *Robust quantum algorithms for oracle identification*, arXiv e-print: quant-ph/0411204, 2004.

[Ang88] Dana Angluin, *Queries and concept learning*, Mach. Learn. **2** (1988), no. 4, 319–342.

[AR03] Jan Arpe and Rüdiger Reischuk, *Robust inference of relevant attributes*, Algorithmic learning theory, Lecture Notes in Comput. Sci., vol. 2842, Springer, Berlin, 2003, pp. 99–113.

[AS05] Alp Atıcı and Rocco A. Servedio, *Improved bounds on quantum learning algorithms*, Quantum Inf. Process. **4** (2005), no. 5, 355–386.

[AS06] Alp Atıcı and Rocco A. Servedio, *Learning unions of $\omega(1)$-dimensional rectangles*, Theoretical Computer Science special issue ALT 2006, to appear (2006), ALT '06: Proc. of the 17th international conference on algorithmic learning theory.

[BBBV97] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh V. Vazirani, *Strengths and weaknesses of quantum computing*, SIAM J. Comput. **26** (1997), no. 5, 1510–1523.

[BBC+01] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf, *Quantum lower bounds by polynomials*, J. Assoc. Comput. Mach. **48** (2001), no. 4, 778–797.

[BBHT98] Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp, *Tight bounds on quantum searching*, Fortschritte der Physik **46** (1998), 493–505.

[BCG+96] Nader H. Bshouty, Richard Cleve, Ricard Gavaldà, Sampath Kannan, and Christino Tamon, *Oracles and queries that are sufficient for exact learning*, J. Comput. System Sci. **52** (1996), no. 3, 421–433, COLT '94: Proc. of the 7th annual conference on computational learning theory.

[BEHW89] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth, *Learnability and the Vapnik-Chervonenkis dimension*, J. Assoc. Comput. Mach. **36** (1989), no. 4, 929–965.

[BFNR03] Harry Buhrman, Lance Fortnow, Ilan Newman, and Hein Röhrig, *Quantum property testing*, Proc. of the 14th annual ACM-SIAM symposium on discrete algorithms, ACM Press, 2003, pp. 480–488.

[BJ99] Nader H. Bshouty and Jeffrey C. Jackson, *Learning DNF over the uniform distribution using a quantum example oracle*, SIAM J. Comput. **28** (1999), no. 3, 1136–1153.

[BK98] Amos Beimel and Eyal Kushilevitz, *Learning boxes in high dimension*, Algorithmica **22** (1998), no. 1-2, 76–90.

[Blu03] Avrim Blum, *Learning a function of r relevant variables*, COLT '03: Proc. of the 16th annual conference on computational learning theory, Springer Berlin / Heidelberg, 2003, pp. 731–733.

[Bru90] Jehoshua Bruck, *Harmonic analysis of polynomial threshold functions*, SIAM J. Discrete Math. **3** (1990), no. 2, 168–177.

[BS92] Jehoshua Bruck and Roman Smolensky, *Polynomial threshold functions,* AC^0 *functions, and spectral norms*, SIAM J. Comput. **21** (1992), no. 1, 33–42.

[BV97] Ethan Bernstein and Umesh V. Vazirani, *Quantum complexity theory*, SIAM J. Comput. **26** (1997), no. 5, 1411–1473.

[CG04] Hana Chockler and Dan Gutfreund, *A lower bound for testing juntas*, Inform. Process. Lett. **90** (2004), no. 6, 301–305.

[CH96] Zhixiang Chen and Steven Homer, *The bounded injury priority method and the learnability of unions of rectangles*, Ann. Pure Appl. Logic **77** (1996), no. 2, 143–168.

[CM94] Zhixiang Chen and Wolfgang Maass, *On-line learning of rectangles and unions of rectangles*, Mach. Learn. **17** (1994), no. 2-3, 201–223.

[Dam01] Ivan Damgård, *QIP note: on the quantum Fourier transform and applications*, 2001, http://www.daimi.au.dk/˜ivan/fourier.pdf.

[Deu85] David Deutsch, *Quantum theory, the Church-Turing principle and the universal quantum computer*, Proc. Roy. Soc. London Ser. A **400** (1985), no. 1818, 97–117.

[Deu89] David Deutsch, *Quantum computational networks*, Proc. Roy. Soc. London Ser. A **425** (1989), no. 1868, 73–90.

[DJ92] David Deutsch and Richard Jozsa, *Rapid solution of problems by quantum computation*, Proc. Roy. Soc. London Ser. A **439** (1992), no. 1907, 553–558.

[DN05] Christopher M. Dawson and Michael A. Nielsen, *The Solovay-Kitaev algorithm*, 2005, arXiv e-print: quant-ph/0505030.

[EHKV89] Andrzej Ehrenfeucht, David Haussler, Michael Kearns, and Leslie Valiant, *A general lower bound on the number of examples needed for learning*, Inform. and Comput. **82** (1989), no. 3, 247–261.

[Fey82] Richard P. Feynman, *Simulating physics with computers*, Internat. J. Theoret. Phys. **21** (1981/82), no. 6-7, 467–488, Physics of computation, Part II (Dedham, Mass., 1981).

[FGGS98] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser, *Limit on the speed of quantum computation in determining parity*, Phys. Rev. Lett. **81** (1998), no. 24, 5442–5444.

[FKR$^+$04] Eldar Fischer, Guy Kindler, Dana Ron, Shmuel Safra, and Alex Samorodnitsky, *Testing juntas*, J. Comput. System Sci. **68** (2004), no. 4, 753–787.

[Fre95] Yoav Freund, *Boosting a weak learning algorithm by majority*, Inform. and Comput. **121** (1995), no. 2, 256–285.

[FS99] Yoav Freund and Robert E. Schapire, *A short introduction to boosting*, J. Japan. Soc. for Artif. Intel. **14** (1999), no. 5, 771–780.

[Gav94] Ricard Gavaldà, *The complexity of learning with queries*, Structure in Complexity Theory Conference, 1994, pp. 324–337.

[GGM94] Paul W. Goldberg, Sally A. Goldman, and H. David Mathias, *Learning unions of boxes with membership and equivalence queries*, COLT '94: Proc. of the 7th annual conference on computational learning theory, ACM Press, 1994, pp. 198–207.

[GGR98] Oded Goldreich, Shafi Goldwasser, and Dana Ron, *Property testing and its connection to learning and approximation*, J. Assoc. Comput. Mach. **45** (1998), no. 4, 653–750.

[Gol99] Sally A. Goldman, *Computational learning theory*, Algorithms and theory of computation handbook, CRC Press, Boca Raton, FL, 1999.

[Gro96] Lov K. Grover, *A fast quantum mechanical algorithm for database search*, STOC '96: Proc. of the 28th annual ACM symposium on the theory of computing, ACM Press, 1996, pp. 212–219.

[Heg95] Tibor Hegedűs, *Generalized teaching dimensions and the query complexity of learning*, COLT '95: Proc. of the 8th annual conference on computational learning theory, ACM Press, 1995, pp. 108–117.

[HMP$^+$93] András Hajnal, Wolfgang Maass, Pavel Pudlák, Márió Szegedy, and György Turán, *Threshold circuits of bounded depth*, J. Comput. System Sci. **46** (1993), no. 2, 129–154.

[HMP$^+$03] Markus Hunziker, David A. Meyer, Jihun Park, James Pommersheim, and Mitch Rothstein, *The geometry of quantum learning*, arXiv e-print: quant-ph/0309059, 2003.

[HPRW96] Lisa Hellerstein, Krishnan Pillaipakkamnatt, Vijay Raghavan, and Dawn Wilkins, *How many queries are needed to learn?*, J. Assoc. Comput. Mach. **43** (1996), no. 5, 840–862.

[Jac97] Jeffrey C. Jackson, *An efficient membership-query algorithm for learning DNF with respect to the uniform distribution*, J. Comput. System Sci. **55** (1997), no. 3, 414–440, FOCS '94: Proc. of the 35th annual IEEE symposium on foundations of computer science.

[JKS02] Jeffrey C. Jackson, Adam R. Klivans, and Rocco A. Servedio, *Learnability beyond* AC^0, STOC '02: Proc. of the 34th annual ACM symposium on theory of computing, ACM Press, 2002, pp. 776–784.

[Kha94] Roni Khardon, *On using the Fourier transform to learn disjoint DNF*, Inf. Process. Lett. **49** (1994), no. 5, 219–222.

[Kit97] Alexei Yu. Kitaev, *Quantum computations: algorithms and error correction*, Russ. Math. Surv. **52** (1997), no. 6, 1191–1249.

[KKL88] Jeff Kahn, Gil Kalai, and Nathan Linial, *The influence of variables on Boolean functions*, FOCS '88: Proc. of the 29th annual IEEE symposium on foundations of computer science, IEEE Comput. Soc. Press, 1988, pp. 68–80.

[KM93] Eyal Kushilevitz and Yishay Mansour, *Learning decision trees using the Fourier spectrum*, SIAM J. Comput. **22** (1993), no. 6, 1331–1348.

[KP98] Matthias Krause and Pavel Pudlák, *Computing Boolean functions by polynomials and threshold circuits*, Comput. Complexity **7** (1998), no. 4, 346–370.

[KS04] Adam R. Klivans and Rocco A. Servedio, *Learning DNF in time* $2^{\tilde{O}(n^{1/3})}$, J. Comput. System Sci. **68** (2004), no. 2, 303–318.

[KSV02] Alexei Yu. Kitaev, Alexander H. Shen, and Mikhail N. Vyalyi, *Classical and quantum computation*, Graduate Studies in Mathematics, vol. 47, American Mathematical Society, Providence, RI, 2002, Translated from the 1999 Russian original by Lester J. Senechal.

[KV94] Michael J. Kearns and Umesh V. Vazirani, *An introduction to computational learning theory*, MIT Press, Cambridge, MA, 1994.

[MOS04] Elchanan Mossel, Ryan O'Donnell, and Rocco A. Servedio, *Learning functions of* k *relevant variables*, J. Comput. System Sci. **69** (2004), no. 3, 421–434.

[MW98] Wolfgang Maass and Manfred K. Warmuth, *Efficient learning with virtual threshold gates*, Inform. and Comput. **141** (1998), no. 1, 66–83.

[MZ03] Michele Mosca and Christof Zalka, *Exact quantum Fourier transforms and discrete logarithm algorithms*, arXiv e-print: quant-ph/0301093, 2003.

[NC00] Michael A. Nielsen and Isaac L. Chuang, *Quantum computation and quantum information*, Cambridge University Press, Cambridge, UK, 2000.

[OS03] Ryan O'Donnell and Rocco A. Servedio, *Extremal properties of polynomial threshold functions*, CCC '03: Eighteenth annual IEEE conference on computational complexity, IEEE Computer Society, 2003, pp. 3–12.

[Pis81] Gilles Pisier, *Remarques sur un résultat non publié de B. Maurey*, Séminaire d'Analyse Fonctionnelle, vol. 1, École Polytechnique, Palaiseau, 1981, pp. 1980–1981.

[RS96] Ronitt Rubinfeld and Madhu Sudan, *Robust characterizations of polynomials with applications to program testing*, SIAM J. Comput. **25** (1996), no. 2, 252–271.

[Sch90] Robert E. Schapire, *The strength of weak learnability*, Mach. Learn. **5** (1990), no. 2, 197–227.

[SG04] Rocco A. Servedio and Steven J. Gortler, *Equivalences and separations between quantum and classical learnability*, SIAM J. Comput. **33** (2004), no. 5, 1067–1092.

[Shi00] Yaoyun Shi, *Lower bounds of quantum black-box complexity and degree of approximating polynomials by influence of Boolean variables*, Inform. Process. Lett. **75** (2000), no. 1-2, 79–83.

[Sho94] Peter W. Shor, *Algorithms for quantum computation: discrete logarithms and factoring*, FOCS '94: Proc. of the 35th annual IEEE symposium on foundations of computer science, IEEE Comput. Soc. Press, 1994, pp. 124–134.

[Sho97] Peter W. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM J. Comput. **26** (1997), no. 5, 1484–1509.

[Sim97] Daniel R. Simon, *On the power of quantum computation*, SIAM J. Comput. **26** (1997), no. 5, 1474–1483.

[Val84] Leslie G. Valiant, *A theory of the learnable*, Commun. Assoc. Comput. Mach. **27** (1984), no. 11, 1134–1142.

[Vap98] Vladimir N. Vapnik, *Statistical learning theory*, Adaptive and Learning Systems for Signal Processing, Communications, and Control, John Wiley & Sons Inc., New York, 1998, A Wiley-Interscience Publication.

[vD98] Wim van Dam, *Quantum oracle interrogation: getting all information for almost half the price*, FOCS '98: Proc. of the 39th annual IEEE symposium on foundations of computer science, IEEE Comput. Soc. Press, 1998, pp. 362–367.

[Yao93] Andrew Chi-Chih Yao, *Quantum circuit complexity*, FOCS '93: Proc. of the 34th annual IEEE symposium on foundations of computer science, IEEE Comput. Soc. Press, 1993, pp. 352–361.