Quantum Algorithms for Learning and Testing Juntas

Alp Atıcı[*]

Citadel Investment Group
Chicago, IL 60603

Rocco A. Servedio[†]

Department of Computer Science
Columbia University
New York, NY 10027
(Dated: July 20, 2007)

In this article we develop quantum algorithms for learning and testing *juntas*, i.e. Boolean functions which depend only on an unknown set of k out of n input variables. Our aim is to develop efficient algorithms:

• whose sample 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 ("black-box") queries. Instead, our algorithms use only classical examples generated uniformly at random and fixed quantum superpositions of such classical examples;

• which require only a few quantum examples but possibly many classical random examples (which are considered quite "cheap" relative to quantum examples).

Our quantum algorithms are based on a subroutine FS which enables sampling according to the Fourier spectrum of f; the FS subroutine was used in earlier work of Bshouty and Jackson on quantum learning. Our results are as follows:

• We give an algorithm for testing k-juntas to accuracy ϵ that uses $O(k/\epsilon)$ quantum examples. This improves on the number of examples used by the best known classical algorithm.

• We establish the following lower bound: any FS-based k-junta testing algorithm requires $\Omega(\sqrt{k})$ queries.

• We give an algorithm for learning k-juntas to accuracy ϵ that uses $O(\epsilon^{-1}k \log k)$ quantum examples and $O(2^k \log(1/\epsilon))$ random examples. We show that this learning algorithms is close to optimal by giving a related lower bound.

Keywords: juntas, quantum query algorithms, quantum property testing, computational learning theory, quantum computation, lower bounds

I. INTRODUCTION

A. Motivation

The field of *computational learning theory* deals with the abilities and limitations of algorithms that learn functions from data. Many models of how learning algorithms access data have been considered in the literature. Among these, two of the most prominent are via *membership queries* and via *random examples*. Membership queries are "black-box" queries; in a membership query, a learning algorithm submits an input x to an oracle and receives the value of $f(x)$. In models of learning from random examples, each time the learning algorithm queries the oracle it receives a labeled example $(x, f(x))$ where x is independently drawn from some fixed probability distribution over the space of all possible examples. (We give precise definitions of these, and all the learning models we consider, in Section II.)

In recent years a number of researchers have considered quantum variants of well-studied models in computational learning theory, see e.g. [1, 4, 8, 10, 15, 16, 28]. As we describe in Section II, models of learning from quantum membership queries and from fixed quantum superpositions of labeled examples (we refer to these as *quantum examples*)

have been considered; such oracles have been studied in the context of *quantum property testing* as well [6, 13, 22]. One common theme in the existing literature on quantum computational learning and testing is that these works study algorithms whose only access to the function is via some form of quantum oracle such as the quantum membership oracle or quantum example oracles mentioned above. For instance, [8] modifies the classical Harmonic Sieve algorithm of [17] so that it uses only uniform quantum examples to learn DNF formulas. [6] 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. [4] studies the information-theoretic requirements of exact learning using quantum membership queries and Probably Approximately Correct (PAC) learning using quantum examples. Many other articles such as [1, 15, 28] could 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 (as opposed to classical) oracle queries or examples required by quantum algorithms. Thus motivated, in this paper we are interested in designing testing and learning algorithms with access to both quantum and classical sources of information (with the goal of minimizing the quantum resources required).

B. Our results

All of our positive results are based on a quantum subroutine due to [8], which we will refer to as an FS (Fourier Sample) oracle call. As explained in Section II, a call to the FS oracle yields a subset of $\{1, \ldots, n\}$ (this set should be viewed as a subset of the input variables $x_1, \ldots, x_n$ of f) drawn according to the Fourier spectrum of the Boolean function f. As demonstrated by [8], such an oracle can be implemented using $O(1)$ uniform quantum examples from a uniform distribution quantum example oracle. In fact, all of our algorithms will be purely classical apart from their use of the FS oracle. Thus, all of our algorithms can be implemented within the (uniform distribution) quantum PAC model first proposed by [8]. This model is a natural quantum extension of the classical PAC model introduced by Valiant [29], as described in Section II. We emphasize that no membership queries, classical or quantum, are used in our algorithms, only uniform quantum superpositions of labeled examples, and we recall that such uniform quantum examples cannot efficiently simulate even classical membership queries in general (see [8]).

Our approach of focusing only on the FS oracle allows us to abstract away from the intricacies of quantum computation, and renders our results useful in any setting in which an FS oracle can be provided to the user. In fact, learning and testing with FS oracle queries may be regarded as a new distinct model (which may possibly be weaker than the uniform distribution quantum example model).

We are primarily interested in the information theoretic requirements (i.e. the number of 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 related to *k-juntas*; these are Boolean functions $f : \{-1, 1\}^n \to \{-1, 1\}$ that depend only on (an unknown subset of) at most k of the n input variables $x_1, \ldots, x_n$. Juntas have been the subject of intensive research in learning theory and property testing in recent years, see e.g. [2, 3, 5, 11, 12, 21, 24].

Our first result, in Section III, is a k-junta testing algorithm which uses $O(k\epsilon^{-1})$ FS oracle calls. Our algorithm uses fewer queries than the best known classical junta testing algorithm due to Fischer *et al.* [12], which uses $O((k \log k)^2 \epsilon^{-1})$ membership queries. However, since the best lower bound known for classical membership query based junta testing (due to Chockler and Gutfreund [11]) 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 k-junta testing algorithm that uses only a FS oracle requires $\Omega(\sqrt{k})$ calls to the FS oracle. This shows that our testing algorithm is not too far from optimal.

Finally, we consider algorithms that can both make FS queries and also access classical random examples. In Section IV we give an algorithm for learning k-juntas over $\{-1, 1\}^n$ that uses $O(\epsilon^{-1} k \log k)$ FS queries and $O(2^k \log(\epsilon^{-1}))$ 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 [8] can in fact be converted into ordinary classical random examples. We show that our algorithm is close to best possible by giving a nearly matching lower bound.

C.   Organization

In Section II we describe the models and problems we will consider and present some useful preliminaries from Fourier analysis and probability. Section III gives our results on testing juntas and Section IV gives our results on learning juntas.

II.   PRELIMINARIES

A.   The problems and the models

In keeping with standard terminology in learning theory, a *concept* f over $\{-1,1\}^n$ is a Boolean function $f : \{-1,1\}^n \to \{-1,1\}$, 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 paper 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 $f : \{-1,1\}^n \to \{-1,1\}$ is a k-junta if f depends only on k out of its n input variables.

1.   The problems

We are interested in the following computational problems:

PAC Learning under the uniform distribution: Given any *target concept* $f \in C$, an *ϵ-learning algorithm for concept class C* under the uniform distribution outputs a *hypothesis* function $h : \{-1,1\}^n \to \{-1,1\}$ which, with probability at least $2/3$, agrees with c on at least a $1 - \epsilon$ fraction of the inputs in $\{-1,1\}^n$. This is a widely studied framework in the learning theory literature both in classical (see for instance [17, 20]) and in quantum (see [8]) versions.

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

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

The notion of property testing was first developed by [14] and [27]. Quantum property testing was first studied by Buhrman *et al.* [6], who first gave an example of an exponential separation between the query complexity of classical and quantum testers for a particular concept class.

Note that a learning or testing algorithm for C "knows" the class C but does not know the identity of the concept f. While our primary concern is the number of oracle calls that our algorithms use, we are also interested in *time efficient* algorithms for testing and learning; for the concept class of k-juntas, these are algorithms running in $\mathrm{poly}(n, 2^k, \epsilon^{-1})$ time steps.

2.   Classical oracles

In order for learning and testing algorithms to gather information about the unknown concept f, they need 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 our algorithms will be allowed access to more than one type of oracle in our discussion.

In this paper we will consider the following types of oracles that provide classical information:

Membership oracle MQ: For f a Boolean function, a *membership oracle* $\mathsf{MQ}(f)$ is an oracle which, when queried with input x, outputs the label $f(x)$ assigned by f to x.

Uniform random example oracle EX: A query $\mathsf{EX}(f)$ of the random example oracle returns an ordered pair $(x, f(x))$ where x is drawn uniformly random from the set $\{-1,1\}^n$ of all possible inputs.

Clearly a 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 it is possible to obtain random labeled examples but impossible to obtained the label of a particular desired example (consider prediction problems dealing with phenomena such as weather or financial markets). We note that the set of concept classes that are known to be efficiently PAC learnable from uniform random examples only is rather limited, see e.g. [19, 23]. In contrast, there are known efficient algorithms that use membership queries to learning important function classes such as DNF (Disjunctive Normal Form) formulas [17].

3.   Quantum oracles:

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

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

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

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 $|1^n, 1\rangle$.

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

The model of PAC learning with a uniform quantum example oracle was introduced by Bshouty and Jackson in [8]. Several researchers have also studied learning from a more powerful $\mathsf{QMQ}(f)$ oracle, see e.g. [1, 4, 16, 28]. Turning to property testing, we are not aware of prior work on quantum testing using only the $\mathsf{QEX}(f)$ oracle; instead researchers have considered quantum testing algorithms that use the more powerful $\mathsf{QMQ}(f)$ oracle, see e.g. [6, 13, 22].

B.   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.

Parseval's Identity, which is an easy consequence of orthonormality of the basis functions, relates the values of the coefficients to the values of the function:

Lemma II.1 (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 will use the following simple and well-known fact:

Fact II.2 (See [20]) *For any* $f: \{-1,1\}^n \to \{-1,1\}$ *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. [18]) that $\mathrm{Inf}_i(f) = \sum_{S \ni x_i} |\hat{f}(S)|^2$.

C.   Additional tools

Fact II.3 (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 Chernoff bounds on tails of sums of independent random variables:

Fact II.4 (Additive Bound) *Let* $X_1, \ldots, X_m$ *be i.i.d. random variables with mean* μ *taking values in the range* $[a,b]$. *Then for all* $\lambda > 0$ *we have* $\mathbf{Pr}[|\frac{1}{m}\sum_{i=1}^m X_i - \mu| \geq \lambda] \leq 2\exp(\frac{-2\lambda^2 m}{(b-a)^2})$.

D.   The Fourier sampling oracle: FS

Definition II.5 *Let* $f : \{-1,1\}^n \to \{-1,1\}$ *be a Boolean function. The* Fourier sampling oracle $FS(f)$ *is the classical oracle which, at each invocation, returns each subset of variables* $S \subseteq \{1, \ldots, n\}$ *with probability* $|\hat{f}(S)|^2$, *where* $\hat{f}(S)$ *denotes the Fourier coefficient corresponding to* $\chi_S(x)$ *as defined in Section II B.*

This oracle will play an important role in our algorithms. Note that by Parseval's Identity we have $\sum_{S \subseteq [n]} |\hat{f}(S)|^2 = 1$ so the probability distribution over sets S indeed has total weight 1.

In [8] Bshouty and Jackson describe a simple constant-size quantum network QSAMP, which has its roots in an idea from [9]. QSAMP allows sampling from the Fourier spectrum of a Boolean function using $O(1)$ QEX oracle queries:

Fact II.6 (See [8]) *For any Boolean function* f, *it is possible to simulate a draw from the* $FS(f)$ *oracle with probability* $1 - \delta$ *using* $O(\log \delta^{-1})$ *queries to* QEX(f).

All the algorithms we describe are actually classical algorithms that make FS queries.

III.   TESTING JUNTAS

Fischer *et al.* [12] 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 with query complexity independent of n, the most efficient of which yields the following:

Theorem III.1 (See [12, Theorem 6]) *There is an algorithm that tests whether an unknown* $f : \{-1,1\}^n \to \{-1,1\}$ *is a* k-junta *using* $O((k \log k)^2 \epsilon^{-1})$ *membership queries.*

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 III.2 (See [11]) *Any algorithm that tests whether* f *is a* k-junta *or is* $1/3$-far *from every* k-junta *must use* $\Omega(k)$ *membership queries.*

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

A. 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 III.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 II, the algorithm can thus be implemented using $O(k/\epsilon)$ uniform quantum examples from 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 $\mathcal{S}$. 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 $g = \sum_{S \subseteq \mathcal{T}} \hat{f}(S)\chi_S, h = \operatorname{sgn}(g)$ and use Fact II.2, 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 $O((k \log k)^2/\epsilon)$ 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.

B. Lower bounds for testing with a FS oracle

1. 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 [11]. To prove Theorem III.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 $k + 2, \ldots, n$.
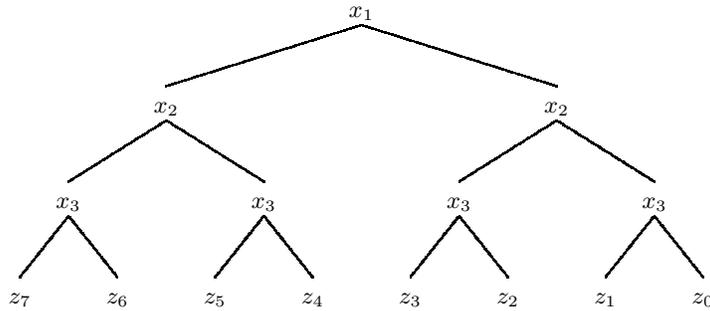
FIG. 1: A decision tree computing the addressing function in the case $r = 3$. The left edge out of each node corresponds to the variable at the node taking value -1 and the right edge to the variable taking value 1.

- **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 III.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 [26], 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 [26] directly implies that for any fixed variable $x_i, i \in 1, \ldots, k+1$, we have:

$$\mathbf{Pr}_{f \leftarrow \mathcal{D}_{k,n}^{(0)}} \left[\sum_{S \ni x_i} \hat{f}(S)^2 > \frac{1}{3} \right] < \exp(-2^{k+1}/2592).$$

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.

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

Our main result in this section is the following theorem:

Theorem III.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 1 depicts a decision tree that computes the addressing function in the case $r = 3$. Formally, the Addressing function $\text{ADDRESSING} : \{-1,1\}^{r+R} \to \{-1,1\}$ is defined as follows:

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

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 $k = r + R/2$ and as we shall see is in fact far from every k-junta) 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 $\mathcal{D}_{\text{REJECT}}, \mathcal{D}_{\text{ACCEPT}}$ over functions on these variables.

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

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

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

   with the variables in T (choosing the variables from T in a uniformly random order). 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 1).

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

$$f_\tau(x_1, \ldots, x_r, y_0, \ldots, y_{n-r-1}) = \text{ADDRESSING}(x_1, x_2, \ldots, x_r, y_{\tau(0)}, y_{\tau(1)}, \ldots, y_{\tau(R-1)}); \tag{III.1}$$

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 III.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 $1/2$. ∎

Fix any function f_τ in C_{REJECT}. We now give an expression for the Fourier representation of f_τ. The expression is obtained by viewing f_τ as a sum of R subfunctions, one for each leaf of the decision tree, where each subfunction takes the appropriate nonzero value on inputs which reach the corresponding leaf and takes value 0 on all other inputs:

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

$$= \frac{1}{2^r} \sum_{\mathbf{i}=0}^{R-1} \sum_{X \subseteq \{x_1, \ldots, x_r\}} (-1)^{(\sum_{x_j \in X} i_j)} y_{\tau(\mathbf{i})} \chi_X. \tag{III.3}$$

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 (III.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 $\left(\frac{1+(-1)^{i_j} x_j}{2}\right) = 0$. Consequently this sum gives rise to exactly the Addressing function in Equation (III.1) which is defined as f_τ and consequently the equality in Equation (III.2) follows. Equation (III.3) follows easily from rearranging (III.2).

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 $0 \le \mathbf{i} < R/2$ 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 $\mathcal{D}_{\mathrm{ACCEPT}}$ 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: $\overline{\mathbf{i}} = \overline{i_1} \circ \overline{i_2} \circ \cdots \circ \overline{i_r} = R - 1 - \mathbf{i}$.

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

$$
\begin{aligned}
& g_{\tau,s}(x_1, \ldots, x_r, y_0, \ldots, y_{n-r-1}) = \\
& \mathrm{ADDRESSING}(x_1, \ldots, x_r, y_{\tau(0)}, \ldots, y_{\tau(R/2-1)}, s_{(R/2-1)} y_{\tau(R/2-1)}, \ldots, s_0 y_{\tau(0)});
\end{aligned}
\tag{III.4}
$$

a draw from $\mathcal{D}_{\mathrm{ACCEPT}}$ is a function chosen uniformly at random from the set $C_{\mathrm{ACCEPT}} = \{g_{\tau,s}\}$ 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_{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 $\mathbf{i}, \overline{\mathbf{i}}$ of the decision tree, we obtain the following expression for the Fourier expansion of each function in the support of $\mathcal{D}_{\mathrm{ACCEPT}}$:

$$
\begin{aligned}
g_{\tau,\mathbf{s}}(x_1, \ldots, x_r, y_0, \ldots, y_{n-r-1}) = & \sum_{\mathbf{i}=i_1 i_2 \ldots i_r = 0}^{R/2-1} y_{\tau(\mathbf{i})} \left(\frac{1 + (-1)^{i_1} x_1}{2} \right) \left(\frac{1 + (-1)^{i_2} x_2}{2} \right) \ldots \left(\frac{1 + (-1)^{i_r} x_r}{2} \right) \\
& + \sum_{\mathbf{i}=0}^{R/2-1} s_{\mathbf{i}} y_{\tau(\mathbf{i})} \left(\frac{1 + (-1)^{\overline{i_1}} x_1}{2} \right) \left(\frac{1 + (-1)^{\overline{i_2}} x_2}{2} \right) \ldots \left(\frac{1 + (-1)^{\overline{i_r}} x_r}{2} \right)
\end{aligned}
\tag{III.5}
$$

$$
[\text{Since } (-1)^{\overline{i_j}} = -(-1)^{i_j}] \quad = \frac{1}{2^{r-1}} \sum_{\mathbf{i}=0}^{R/2-1}
\begin{cases}
\sum_{X \subseteq \{x_1, \ldots, x_r\}, |X| \text{ even}} (-1)^{(\sum_{x_j \in X} i_j)} y_{\tau(\mathbf{i})} \chi_X & \text{if } s_{\mathbf{i}} = 1; \\
\sum_{X \subseteq \{x_1, \ldots, x_r\}, |X| \text{ odd}} (-1)^{(\sum_{x_j \in X} i_j)} y_{\tau(\mathbf{i})} \chi_X & \text{if } s_{\mathbf{i}} = -1.
\end{cases}
\tag{III.6}
$$

Just as in the Equation (III.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 (III.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 $\mathbf{i} \ge R/2$. Therefore this sum gives rise to exactly the Addressing function in Equation (III.4) which is defined as $g_{\tau,s}$ and consequently the equality in Equation (III.5) follows.

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

Now we are ready to prove Theorem III.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. Considering the equations (III.3) and (III.6), for any f in C_{ACCEPT} or C_{REJECT} its FS oracle will return a pair of the form $(y_{j=\tau(\mathbf{i})}, X)$, $X \subseteq \{x_1, \ldots, x_r\}$.

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 $f_\tau \leftarrow \mathcal{D}_{\mathrm{REJECT}}$, 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 (III.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 $g_{\tau,\mathbf{s}} \leftarrow \mathcal{D}_{\mathrm{ACCEPT}}$, 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 (III.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 II.3 implies that no algorithm making only N oracle calls can distinguish between these two scenarios with high probability. This gives us the result, and concludes the proof of Theorem III.5. ∎

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 $\Theta(\sqrt{R})$ times it will start to see some y_i's more than once with constant probability (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 with constant probability between these two scenarios.

IV. LEARNING JUNTAS

A. Known results

The problem of learning an unknown k-junta has been well studied in the computational learning theory literature, see e.g. [2, 5, 24]. The following classical lower bound will be a yardstick against which we will measure our results.

Lemma IV.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 function $f(x)$ which is simply a single Boolean variable from $\{x_1, \ldots, x_n\}$. 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 requires $\Omega(\log n)$ queries for any exact learning algorithm that uses membership queries only, see e.g. [7]. This gives the $\Omega(\log n)$ lower bound.

For the $\Omega(2^k)$ lower bound, we may suppose that 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 [24].

Bshouty and Jackson [8] 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 IV.2 (See [8]) *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 [8] 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 [4, Observation 6.3] for learning with quantum membership queries for an arbitrary concept class can be rephrased for the purpose of learning k-juntas as follows.

Fact IV.3 (See [4]) *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 [4, 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.

B. 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 IV.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 IV.5 *There is an efficient algorithm ϵ-learning k-juntas with $O(\epsilon^{-1}k \log k)$ queries of the FS oracle and $O(2^k \log(\epsilon^{-1}))$ random examples.*

Algorithm 1 The junta learning algorithm.

1: **Input:** $\epsilon > 0, \mathrm{FS}(f), \mathsf{EX}(f)$.
2: **Stage 1:**
3: Construct a set containing all variables of f with an influence at least $(\epsilon/10k)$ using the algorithm in Lemma IV.4. Let $\mathcal{A}$ be the final result.
4: $\forall \mathbf{a} \in \{-1, 1\}^{|\mathcal{A}|}, encountered(\mathbf{a}) \leftarrow \text{FALSE}$.
5: **Stage 2:**
6: **repeat**
7: $\langle x, f(x) \rangle \leftarrow$ Draw from $\mathsf{EX}(f)$. Let $x|_{\mathcal{A}}$ denote the projection of x onto the variables in $\mathcal{A}$.
8: **if** $encountered(x|_{\mathcal{A}}) = \text{FALSE}$ **then**
9: $value(x|_{\mathcal{A}}) \leftarrow f(x), encountered(x|_{\mathcal{A}}) \leftarrow \text{TRUE}$.
10: **end if**
11: **until** For at least $(1 - \epsilon/3)$ fraction of all $\mathbf{a} \in \{-1, 1\}^{|\mathcal{A}|}, encountered(\mathbf{a}) = \text{TRUE}$.
12: Output the hypothesis:
$$H(x) = \begin{cases} value(x|_{\mathcal{A}}) & \text{if } encountered(x|_{\mathcal{A}}) = \text{TRUE} \\ \text{TRUE} & \text{otherwise.} \end{cases}$$

Proof: We claim Algorithm 1 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 IV.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 IV.4 was successful, we group the variables as follows:

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 $n - |\mathcal{I}|$ variables the function does not depend on.

Note that $|\mathcal{A}| + |\mathcal{B}| \leq k$ by Lemma IV.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 $x = (\mathbf{a}, \mathbf{b^a}, \mathbf{c^a})$. At the end of the algorithm, the following hypothesis is produced as the output:

$$H(\mathbf{a}, *, *) = \begin{cases} f(\mathbf{a}, \mathbf{b^a}, \mathbf{c^a}) & \text{if } \mathbf{a} \in \mathcal{S} \\ \text{TRUE} & \text{otherwise.} \end{cases}$$

In other words, the value of the hypothesis only depends on the setting of the variables in $\mathcal{A}$. Observe 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 IV.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,

$$\text{For any } s \in \mathbb{B}^{|\mathcal{B}|}, \quad \frac{1}{2^n} \sum_{\mathbf{a} \in \mathbb{B}^{|\mathcal{A}|}} \sum_{\mathbf{b} \in \mathbb{B}^{|\mathcal{B}|}} \sum_{\mathbf{c} \in \mathbb{B}^{|\mathcal{C}|}} \mathbf{1}_{[f(\mathbf{a}, \mathbf{b} \odot s, \mathbf{c}) \neq f(\mathbf{a}, \mathbf{b}, \mathbf{c})]}$$

$$\leq \frac{1}{2^n} \sum_{\mathbf{a} \in \mathbb{B}^{|\mathcal{A}|}} \sum_{\mathbf{b} \in \mathbb{B}^{|\mathcal{B}|}} \sum_{\mathbf{c} \in \mathbb{B}^{|\mathcal{C}|}} \sum_{i=1}^{|s|} \mathbf{1}_{[f(\mathbf{a}, \mathbf{b} \odot u^{i+1}, \mathbf{c}) \neq f(\mathbf{a}, \mathbf{b} \odot u^i, \mathbf{c})]}$$

$$= \sum_{i=1}^{|s|} \underbrace{\left(\frac{1}{2^n} \sum_{\mathbf{a} \in \mathbb{B}^{|\mathcal{A}|}} \sum_{\mathbf{b} \in \mathbb{B}^{|\mathcal{B}|}} \sum_{\mathbf{c} \in \mathbb{B}^{|\mathcal{C}|}} \mathbf{1}_{[f(\mathbf{a}, \mathbf{b} \odot u^i \odot u^{i+1}, \mathbf{c}) \neq f(\mathbf{a}, \mathbf{b}, \mathbf{c})]} \right)}_{= \text{The influence of the unique variable } b_{j(i)} \text{ that takes value } -1 \text{ in } u^{i+1} \odot u^i}$$

$$< \epsilon/10. \quad \left[\text{Since every } b_j \in \mathcal{B} \text{ has influence} < \frac{\epsilon}{10k} \text{ and } |\mathcal{B}| \leq k \right]$$

∎

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 $F_\Gamma : \{-1, 1\}^n \to \{-1, 1\}$ 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:

$$\mathbf{Pr}_{(\mathbf{a}, \mathbf{b}, \mathbf{c})}[F_\Gamma(\mathbf{a}, \mathbf{b}, \mathbf{c}) \neq f(\mathbf{a}, \mathbf{b}, \mathbf{c})] = \mathbf{Pr}_{(\mathbf{a}, \mathbf{b}, \mathbf{c})}[f(\mathbf{a}, \mathbf{b^a}, \mathbf{c^a}) \neq f(\mathbf{a}, \mathbf{b}, \mathbf{c})]$$

$$= \mathbf{Pr}_{(\mathbf{a}, \mathbf{b}, \mathbf{c})}[f(\mathbf{a}, \mathbf{b^a}, \mathbf{c}) \neq f(\mathbf{a}, \mathbf{b}, \mathbf{c})] \quad [\text{Since } f \text{ does not depend on the variables in } \mathcal{C}]$$

$$= \frac{1}{2^n} \sum_{\mathbf{a} \in \mathbb{B}^{|\mathcal{A}|}} \sum_{\mathbf{b} \in \mathbb{B}^{|\mathcal{B}|}} \sum_{\mathbf{c} \in \mathbb{B}^{|\mathcal{C}|}} \mathbf{1}_{[f(\mathbf{a}, \mathbf{b^a}, \mathbf{c}) \neq f(\mathbf{a}, \mathbf{b}, \mathbf{c})]} = \frac{1}{2^n} \sum_{\mathbf{a} \in \mathbb{B}^{|\mathcal{A}|}} \sum_{s \in \mathbb{B}^{|\mathcal{B}|}} \sum_{\mathbf{c} \in \mathbb{B}^{|\mathcal{C}|}} \mathbf{1}_{[f(\mathbf{a}, \mathbf{b^a}, \mathbf{c}) \neq f(\mathbf{a}, \mathbf{b^a} \odot s, \mathbf{c})]} \quad \text{(IV.1)}$$

Therefore if we consider the expected value of the incurred error $\mathbf{Pr}[F_\Gamma \neq f]$ over all "lists of assignments" Γ, equation (IV.1) implies that:

$$\mathbf{E}_\Gamma[\mathbf{Pr}_{(\mathbf{a}, \mathbf{b}, \mathbf{c})}[F_\Gamma \neq f]] = \frac{1}{2^{|\mathcal{B}|}} \sum_{s \in \mathbb{B}^{|\mathcal{B}|}} \underbrace{\left(\frac{1}{2^n} \sum_{\mathbf{a} \in \mathbb{B}^{|\mathcal{A}|}} \sum_{\mathbf{b^a} \in \mathbb{B}^{|\mathcal{B}|}} \sum_{\mathbf{c} \in \mathbb{B}^{|\mathcal{C}|}} \mathbf{1}_{[f(\mathbf{a}, \mathbf{b^a} \odot s, \mathbf{c}) \neq f(\mathbf{a}, \mathbf{b^a}, \mathbf{c})]} \right)}_{< \epsilon/10, \text{ due to Fact IV.6}}$$

$$< \epsilon/10.$$

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 IV.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 IV.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

- uses only a moderate number of quantum examples;

- has overall query complexity with no dependence on n, in contrast with known lower bounds (Lemma IV.1) for learning from classical membership queries;

- uses the EX oracle as its only source of classical information (MQ queries are not used); and

- is computationally efficient.

One can compare this result to that of Theorem IV.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 [8] can in fact be converted into ordinary classical random examples.

1. Lower bounds

The algorithm of Theorem IV.5 is optimal in the following sense:

Observation IV.8 *Any $1/10$-learning quantum membership query algorithm for k-juntas that uses only $\frac{1}{101}2^k$ classical MQ queries must additionally use $\Omega(2^k)$ QMQ queries.*

Proof: This statement easily follows from Fact IV.3 since a classical membership query can be simulated by a QMQ query. ∎

Contrasting our junta learning algorithm with Observation IV.8, we see that if the allowed number of classical examples or queries is decreased even slightly from the $O(2^k \log \epsilon^{-1})$ used by our algorithm to $\frac{1}{101}2^k$, then an additional $\Omega(2^k)$ quantum queries are required, even if QMQ queries are allowed.

V. CONCLUSION

We have given some results on learning and testing k-juntas using both quantum examples and classical random examples. It would be interesting to develop other testing and learning algorithms that combine these two sorts of oracles, with the goal of minimizing the number of quantum oracle calls required.

Another interesting goal for future work is to further explore the power of the FS oracle. Can the gap between our $O(k/\epsilon)$-query upper bound and our $\Omega(\sqrt{k})$-query lower bound for the FS oracle be closed?

---

[1] A. Ambainis, K. Iwama, A. Kawachi, H. Masuda, R. H. Putra, S. Yamashita, *Quantum Identification of Boolean Oracles*, Proceedings of STACS 2004, pp. 93-104.

[2] J. Arpe and R. Reischuk, *Robust Inference of Relevant Attributes*, Proceedings of the 14th International Conference on Algorithmic Learning Theory, pp. 99–113 (2003).

[3] J. Arpe and R. Reischuk, *Learning Juntas in the Presence of Noise*, Proceedings of the 3rd International Conference on Theory and Applications of Models of Computation, pp. 387–398 (2006).

[4] A. Atıcı, R. A. Servedio, *Improved Bounds on Quantum Learning Algorithms*, Quantum Information Processing, Vol. **4**, No. 5, pp. 355–386 (2005).

[5] A. Blum, *Learning a Function of r Relevant Variables (Open Problem)*, Proceedings of the 16th Annual Conference on Learning Theory and 7th Kernel Workshop, pp. 731–733 (2003).

[6] H. Buhrman, L. Fortnow, I. Newman, H. Röhrig, *Quantum Property Testing*, Proceedings of 14th SODA, pp. 480–488 (2003).

[7] N. Bshouty, R. Cleve, R. Gavaldà, S. Kannan and C. Tamon. *Oracles and queries that are sufficient for exact learning*, J. Comput. Syst. Sci., Vol **52**, No. 3 , pp. 421-433 (1996).

[8] N. H. Bshouty, J. C. Jackson, *Learning DNF over the Uniform Distribution Using a Quantum Example Oracle*, SIAM J. Comput. Vol. **28**, No. 3, pp. 1136–1153 (1999).

[9] E. Bernstein, U. Vazirani, *Quantum Complexity Theory*, SIAM Journal of Computing, **26**(5): pp. 1411–1473 (1997).

[10] J. Castro, *How many query superpositions are needed to learn?* Proceedings of 17th ALT, pp. 78-92 (2006).

[11] H. Chockler, D. Gutfreund, *A Lower Bound for Testing Juntas*, Information Processing Letters **90**(6): pp. 301–305 (2004).

[12] E. Fischer, G. Kindler, D. Ron, S. Safra, A. Samorodnitsky, *Testing Juntas*, Proceedings of the 43rd IEEE Symposium on Foundations of Computer Science, pp. 103–112 (2002).

[13] K. Friedl, F. Magniez, M. Santha, P. Sen. *Quantum Testers for Hidden Group Properties*, Proceedings of the 28th International Symposium on Mathematical Foundations of Computer Science, pp. 419–428.

[14] O. Goldreich, S. Goldwasser, D. Ron, *Property Testing and Its Connection to Learning and Approximation*, Journal of the ACM, **45**(4): pp. 653–750 (1998).

[15] M. Hunziker, D. A. Meyer, J. Park, J. Pommersheim and M. Rothstein, *The Geometry of Quantum Learning*, arXiv:quant-ph/0309059; to appear in Quantum Information Processing.

[16] K. Iwama, A. Kawachi, R. Raymond and S. Yamashita, *Robust Quantum Algorithms for Oracle Identification*, arXiv:quant-ph/0411204 (2005).

[17] J. C. Jackson, *An Efficient Membership-Query Algorithm for Learning* DNF *with Respect to the Uniform Distribution*, Journal of Computer and System Sciences **55**(3): 414–440 (1997).

[18] J. Kahn, G. Kalai, N. Linial, *The influence of variables on boolean functions*, Proceedings of the 29th IEEE Symposium on Foundations of Computer Science, pp. 68–80 (1988).

[19] J. Köbler, W. Lindner, *Learning Boolean Functions under the uniform distribution via the Fourier Transform*, Bulletin of the EATCS 89 (2006).

[20] E. Kushilevitz, Y. Mansour, *Learning Decision Trees using the Fourier Spectrum*, SIAM Journal on Computing **22**(6): 1331-1348 (1993).

[21] R. Lipton, E. Markakis, A. Mehta, N. Vishnoi, *On the Fourier Spectrum of Symmetric Boolean Functions with Applications to Learning Symmetric Juntas*, Proceedings of the 20th Annual IEEE Conference on Computational Complexity, pp. 112–119 (2005).

[22] F. Magniez, A. Nayak. *Quantum Complexity of Testing Group Commutativity*, Proceedings of the 32nd International Colloquium on Automata, Languages and Programming, pp. 1312–1324 (2005).

[23] Y. Mansour. *Learning Boolean functions via the Fourier transform*, in "Theoretical Advances in Neural Computation and Learning," Kluwer Academic Publishers, pp. 391-424 (1994).

[24] E. Mossel, R. O'Donnell and R. A. Servedio, *Learning Functions of k Variables*, Journal of Computer and System Sciences, Vol. **69**, No. 3, pp. 421–434 (2004).

[25] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press (2000).

[26] R. O'Donnell, R. A. Servedio, *Extremal Properties of Polynomial Threshold Functions*, Journal of Computer & System Sciences, to appear. Available at http://www.cs.columbia.edu/~rocco/papers/ccc03.html. Preliminary version appeared in Eighteenth Annual IEEE Conference on Computational Complexity, pp. 3–12 (2003).

[27] R. Rubinfeld and M. Sudan, *Robust Characterizations of Polynomials with Applications to Program Testing*, SIAM Journal on Computing, **25**(2): pp. 252–271 (1996).

[28] R. A. Servedio, S. J. Gortler, *Equivalences and Separations between Quantum and Classical Learnability*, SIAM J. Comput. Vol. **33**, No. 5, pp. 1067–1092 (2004).

[29] L. G. Valiant, *A Theory of the Learnable*, Communications of the Association for Computing Machinery 27:11, pp. 1134–1142 (1984).