EQUIVALENCES AND SEPARATIONS BETWEEN QUANTUM AND CLASSICAL LEARNABILITY*

ROCCO A. SERVEDIO† AND STEVEN J. GORTLER‡

Abstract. We consider quantum versions of two well-studied models of learning Boolean functions: Angluin's model of exact learning from membership queries and Valiant's Probably Approximately Correct (PAC) model of learning from random examples. For each of these two learning models we establish a polynomial relationship between the number of quantum versus classical queries required for learning. These results contrast known results which show that testing black-box functions for various properties, as opposed to learning, can require exponentially more classical queries than quantum queries. We also show that under a widely held computational hardness assumption (the intractability of factoring Blum integers) there is a class of Boolean functions which is polynomial-time learnable in the quantum version but not the classical version of each learning model. For the model of exact learning from membership queries, we establish a stronger separation by showing that if any one-way function exists, then there is a class of functions which is polynomial-time learnable in the quantum setting but not in the classical setting. Thus, while quantum and classical learning are equally powerful from an information theory perspective, the models are different when viewed from a computational complexity perspective.

Key words. computational learning theory, quantum computation, PAC learning, query complexity

AMS subject classifications. 68Q32, 68T05, 81P68

1. Introduction.

1.1. Motivation. In recent years many researchers have investigated the power of quantum computers which can query a black-box oracle for an unknown function [1, 5, 6, 9, 14, 10, 11, 15, 17, 20, 21, 24, 40, 46]. The broad goal of research in this area is to understand the relationship between the number of quantum versus classical oracle queries which are required to answer various questions about the function computed by the oracle. For example, a well-known result due to Deutsch and Jozsa [17] shows that exponentially fewer queries are required in the quantum model in order to determine with certainty whether a black-box oracle computes a constant Boolean function or a function which is balanced between outputs 0 and 1. More recently, several researchers have studied the number of quantum oracle queries which are required to determine whether the function computed by a black-box oracle is identically zero [5, 6, 9, 15, 24, 46].

A natural question which arises in this framework is the following: what is the relationship between the number of quantum versus classical oracle queries which are required in order to *exactly identify* the function computed by a black-box oracle? Here the goal is not to determine whether a black-box function satisfies some particular property such as ever taking a nonzero value, but rather to precisely identify an unknown black-box function from some restricted class of possible functions. The

*This paper combines results from conference papers [37] and [38].

†Department of Computer Science, Columbia University, 1214 Amsterdam Avenue, Mailcode 0401, New York, NY, 10027-7003 (rocco@cs.columbia.edu). This research was performed while at the Division of Engineering and Applied Sciences of Harvard University, and while supported by an NSF Graduate Research Fellowship, by NSF grant CCR-98-77049, and by the National Security Agency (NSA) and Advanced Research and Development Activity (ARDA) under Army Research Office (ARO) contract no. DAAD19-01-1-0506.

‡ Division of Engineering and Applied Sciences, Harvard University, 33 Oxford Street, Cambridge, MA, 02138 (sjg@cs.harvard.edu).

classical version of this problem has been well studied in the computational learning theory literature [2, 12, 22, 26, 27] and is known as the problem of *exact learning from membership queries*. The question stated above can thus be rephrased as follows: what is the relationship between the number of quantum versus classical membership queries which are required for exact learning? We answer this question in this paper.

In addition to the model of exact learning from membership queries, we also consider a quantum version of Valiant's widely studied PAC learning model which was introduced by Bshouty and Jackson [13]. While a learning algorithm in the classical PAC model has access to labeled examples drawn from some fixed probability distribution, a learning algorithm in the quantum PAC model has access to some fixed quantum superposition of labeled examples. Bshouty and Jackson gave a polynomial-time algorithm for a particular learning problem in the quantum PAC model, but did not address the general relationship between the number of quantum versus classical examples which are required for PAC learning. We answer this question as well.

1.2. Our results. We show that in an information-theoretic sense, quantum and classical learning are equivalent up to polynomial factors: for both the model of exact learning from membership queries and the PAC model, there is no learning problem which can be solved using significantly fewer quantum queries than classical queries. More precisely, our first main theorem is the following:

THEOREM 1.1. *Let $\mathcal{C}$ be any class of Boolean functions over $\{0,1\}^n$ and let D and Q be such that $\mathcal{C}$ is exact learnable from D classical membership queries or from Q quantum membership queries. Then $D = O(nQ^3)$.*

Our second main theorem is an analogous result for quantum versus classical PAC learnability:

THEOREM 1.2. *Let $\mathcal{C}$ be any class of Boolean functions over $\{0,1\}^n$ and let D and Q be such that $\mathcal{C}$ is PAC learnable from D classical examples or from Q quantum examples. Then $D = O(nQ)$.*

These results draw on lower bound techniques from both quantum computation and computational learning theory [2, 5, 6, 8, 12, 26]. A detailed description of the relationship between our results and previous work on quantum versus classical black-box query complexity is given in Section 3.4.

Theorems 1.1 and 1.2 are information-theoretic rather than computational in nature; they show that for any learning problem, if there is a quantum learning algorithm which uses polynomially many examples then there must also exist a classical learning algorithm which uses polynomially many examples. However, Theorems 1.1 and 1.2 do not imply that every *polynomial time* quantum learning algorithm must have a polynomial time classical analogue. In fact, we show that a separation exists between efficient quantum learnability and efficient clasical learnability. Under a widely held computational hardness assumption for classical computation (the hardness of factoring Blum integers), we observe that for each of the two learning models considered in this paper there is a concept class which is polynomial-time learnable in the quantum version but not in the classical version of the model.

For the model of exact learning from membership queries we give an even stronger separation between efficient quantum and classical learnability. Our third main theorem is:

THEOREM 1.3. *If any one-way function exists, then there is a concept class C which is polynomial-time exact learnable from quantum membership queries but is not polynomial-time exact learnable from classical membership queries.*

This result establishes a robust separation between efficient quantum and classical learnability. Even if a polynomial-time classical factoring algorithm were to be discovered, the separation would hold as long as *any* one-way function exists (a universally held belief in public-key cryptography). As discussed in Section 9, our results prove the existence of a quantum oracle algorithm which defeats a general cryptographic construction secure in the classical setting. More precisely, given any one-way function we construct a family of pseudorandom functions which are classically secure but can be distinguished from truly random functions (and in fact exactly identified) by an algorithm which can make quantum oracle queries. To our knowledge, this is the first break of a generic cryptographic construction (not based on a specific assumption such as factoring) in a quantum setting.

The main cryptographic tool underlying Theorem 1.3 is a new construction of pseudorandom functions which are invariant under an XOR mask (see Section 7). As described in Section 8.1, each concept $c \in C$ combines these new pseudorandom functions with pseudorandom permutations in a particular way. Roughly speaking, the XOR mask invariance of the new pseudorandom functions ensures that a quantum algorithm due to Simon [40] can be used to extract some information about the structure of the target concept and thus make progress towards learning. On the other hand, the pseudorandomness ensures that no probabilistic polynomial-time learning algorithm can extract any useful information, and thus no such algorithm can learn successfully.

1.3. Organization. We set notation, define the exact learning model and the PAC learning model, and describe the quantum computation framework in Section 2. We prove the relationship between quantum and classical exact learning from membership queries (Theorem 1.1) in Section 3, and we prove the relationship between quantum and classical PAC learning (Theorem 1.2) in Section 4. In Section 5 we observe that if factoring Blum integers is classically hard, then in each of these two learning models there is a concept class which is quantum learnable in polynomial time but not classically learnable in polynomial time. We prove Theorem 1.3 in Sections 6 through 8.

2. Preliminaries. For $\alpha, \beta \in \{0,1\}$ we write $\alpha \oplus \beta$ to denote the exclusive-or $\alpha + \beta \pmod 2$. Similarly for $x, y \in \{0,1\}^n$ we write $x \oplus y$ to denote the n-bit string which is the bitwise XOR of x and y. We write $x \cdot y$ to denote the inner product $x_1 y_1 + \cdots + x_n y_n \pmod 2$, and we write $|x|$ to denote the length of string x.

We use script capital letters to denote probability distributions over sets of functions; in particular $\mathcal{F}_n$ denotes the uniform distribution over all 2^{n2^n} functions from $\{0,1\}^n$ to $\{0,1\}^n$. If S is a finite set we write $\Pr_{s \in S}$ to denote a uniform choice of s from S.

We write $M(s)$ to indicate that algorithm M is given string s as input and M^g to indicate that M has access to an oracle for the function g. If M is a probabilistic polynomial-time (henceforth abbreviated p.p.t.) algorithm which has access to an oracle $g : \{0,1\}^{\ell_1} \rightarrow \{0,1\}^{\ell_2}$, then the running time of M^g is bounded by $p(\ell_1 + \ell_2)$ for some polynomial p.

A *concept* c over $\{0,1\}^n$ is a Boolean function over the domain $\{0,1\}^n$, or equivalently a concept can be viewed as a subset $\{x \in \{0,1\}^n : c(x) = 1\}$ of $\{0,1\}^n$. A *concept class* $\mathcal{C} = \cup_{n \geq 1} C_n$ is a collection of concepts, where $C_n = \{c \in \mathcal{C} : \ c$ is a concept over $\{0,1\}^n\}$. For example, C_n might be the family of all Boolean formulae over n variables which are of size at most n^2. We say that a pair $\langle x, c(x) \rangle$ is a *labeled example* of the concept c.

While many different learning models have been proposed, most models follow the same basic paradigm: a learning algorithm for a concept class $\mathcal{C}$ typically has access to some kind of oracle which provides examples that are labeled according to a fixed but unknown target concept $c \in \mathcal{C}$, and the goal of the learning algorithm is to infer (in some sense) the target concept c. The two learning models which we discuss in this paper, the model of exact learning from membership queries and the PAC model, make this rough notion precise in different ways.

2.1. Classical Exact Learning from Membership Queries. The model of *exact learning from membership queries* was introduced by Angluin [2] and has since been widely studied [2, 12, 22, 26, 27]. In this model the learning algorithm has access to a *membership oracle* MQ_c where $c \in C_n$ is the unknown target concept. When given an input string $x \in \{0,1\}^n$, in one time step the oracle MQ_c returns the bit $c(x)$; such an invocation is known as a *membership query* since the oracle's answer tells whether or not $x \in c$ (viewing c as a subset of $\{0,1\}^n$). The goal of the learning algorithm is to construct a hypothesis $h : \{0,1\}^n \rightarrow \{0,1\}$ which is logically equivalent to c, i.e. $h(x) = c(x)$ for all $x \in \{0,1\}^n$. Formally, we say that an algorithm A is an *exact learning algorithm for* $\mathcal{C}$ *using membership queries* if for all $n \geq 1$, for all $c \in C_n$, if A is given n and access to MQ_c, then with probability at least $2/3$ (over the internal randomness of A) algorithm A outputs a Boolean circuit h such that $h(x) = c(x)$ for all $x \in \{0,1\}^n$. The *sample complexity* $T(n)$ of a learning algorithm A for $\mathcal{C}$ is the maximum number of calls to MQ_c which A ever makes for any $c \in C_n$.

2.2. Classical PAC Learning. The PAC (Probably Approximately Correct) model of concept learning was introduced by Valiant in [41] and has since been extensively studied [4, 29]. In this model the learning algorithm has access to an *example oracle* $EX(c, \mathcal{D})$ where $c \in C_n$ is the unknown target concept and $\mathcal{D}$ is an unknown distribution over $\{0,1\}^n$. The oracle $EX(c, \mathcal{D})$ takes no inputs; when invoked, in one time step it returns a labeled example $\langle x, c(x) \rangle$ where $x \in \{0,1\}^n$ is randomly selected according to the distribution $\mathcal{D}$. The goal of the learning algorithm is to generate a hypothesis $h : \{0,1\}^n \rightarrow \{0,1\}$ which is an *ϵ-approximator for c under $\mathcal{D}$*, i.e. a hypothesis h such that $\Pr_{x \in \mathcal{D}}[h(x) \neq c(x)] \leq \epsilon$. An algorithm A is a *PAC learning algorithm for* $\mathcal{C}$ if the following condition holds: for all $n \geq 1$ and $0 < \epsilon, \delta < 1$, for all $c \in C_n$, for all distributions $\mathcal{D}$ over $\{0,1\}^n$, if A is given n, ϵ, δ and access to $EX(c, \mathcal{D})$, then with probability at least $1 - \delta$ algorithm A outputs a circuit h which is an ϵ-approximator for c under $\mathcal{D}$. The *sample complexity* $T(n, \epsilon, \delta)$ of a learning algorithm A for $\mathcal{C}$ is the maximum number of calls to $EX(c, \mathcal{D})$ which A ever makes for any concept $c \in C_n$ and any distribution $\mathcal{D}$ over $\{0,1\}^n$.

2.3. Quantum Computation. Detailed descriptions of the quantum computation model can be found in [7, 16, 31, 45]; here we outline only the basics using the terminology of *quantum networks* as presented in [5]. A quantum network $\mathcal{N}$ is a quantum circuit (over some standard basis augmented with one oracle gate) which acts on an m-bit quantum register; the computational basis states of this register are the 2^m binary strings of length m. A quantum network can be viewed as a sequence of unitary transformations

$$U_0, O_1, U_1, O_2, \ldots, U_{T-1}, O_T, U_T,$$

where each U_i is an arbitrary unitary transformation on m qubits and each O_i is a unitary transformation which corresponds to an oracle call.[1] Such a network is

[1] Since there is only one kind of oracle gate, each O_i is the same transformation.

said to have *query complexity* T. At every stage in the execution of the network, the current state of the register can be represented as a superposition $\sum_{z\in\{0,1\}^m} \alpha_z|z\rangle$ where the α_z are complex numbers which satisfy $\sum_{z\in\{0,1\}^m} \|\alpha_z\|^2 = 1$. If this state is measured, then with probability $\|\alpha_z\|^2$ the string $z \in \{0,1\}^m$ is observed and the state collapses down to $|z\rangle$. After the final transformation U_T takes place, a measurement is performed on some subset of the bits in the register and the observed value (a classical bit string) is the output of the computation.

Several points deserve mention here. First, since the information which our quantum network uses for its computation comes from the oracle calls, we may stipulate that the initial state of the quantum register is $|0^m\rangle$. Second, as described above each U_i can be an arbitrarily complicated unitary transformation (as long as it does not contain any oracle calls) which may require a large quantum circuit to implement. This is of small concern since we are chiefly interested in query complexity and not circuit size. Third, as defined above our quantum networks can make only one measurement at the very end of the computation; this is an inessential restriction since any algorithm which uses intermediate measurements can be modified to an algorithm which makes only one final measurement. Finally, we have not specified just how the oracle calls O_i work; we address this point separately in Sections 3.1 and 4.1 for each type of oracle.

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* betweeen $|\phi\rangle$ 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 $\sum_x |\mathcal{D}_1(x) - \mathcal{D}_2(x)|$. The following fact (Lemma 3.2.6 of [7]), which relates the Euclidean distance between two superpositions and the total variation distance between the distributions induced by measuring the two superpositions, will be useful:

FACT 1. *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.*

3. Exact Learning from Quantum Membership Queries.

3.1. Quantum Membership Queries. A *quantum membership oracle* QMQ_c is the natural quantum generalization of a classical membership oracle MQ_c: on input a superposition of query strings, the oracle QMQ_c generates the corresponding superposition of example labels. More formally, a QMQ_c gate maps the basis state $|x, b\rangle$ (where $x \in \{0,1\}^n$ and $b \in \{0,1\}$) to the state $|x, b\oplus c(x)\rangle$. If $\mathcal{N}$ is a quantum network which has QMQ_c gates as its oracle gates, then each O_i is the unitary transformation which maps $|x, b, y\rangle$ (where $x \in \{0,1\}^n$, $b \in \{0,1\}$ and $y \in \{0,1\}^{m-n-1}$) to $|x, b\oplus c(x), y\rangle$.[2] Our QMQ_c oracle is identical to the well-studied notion of a quantum black-box oracle for c [5, 6, 7, 9, 10, 11, 15, 17, 24, 46].

A *quantum exact learning algorithm for* $\mathcal{C}$ is a family $\mathcal{N}_1, \mathcal{N}_2, \ldots,$ of quantum networks where each network $\mathcal{N}_n$ has a fixed architecture independent of the choice of $c \in C_n$, with the following property: for all $n \geq 1$, for all $c \in C_n$, if $\mathcal{N}_n$'s oracle gates are instantiated as QMQ_c gates, then with probability at least $2/3$ the network $\mathcal{N}_n$ outputs 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 sample complexity* of a quantum exact learning algorithm for $\mathcal{C}$ is $T(n)$, where $T(n)$ is the query complexity of $\mathcal{N}_n$.

[2] Note that each O_i only affects the first $n+1$ bits of a basis state. This is without loss of generality since the transformations U_j can "permute bits" of the network.

3.2. Lower Bounds on Classical and Quantum Exact Learning. Two different lower bounds are known for the number of classical membership queries which are required to exact learn any concept class. In this section we prove two analogous lower bounds on the number of quantum membership queries required to exact learn any concept class. Throughout this section for ease of notation we omit the subscript n and write C for C_n.

A Lower Bound Based on Similarity of Concepts. Consider a set of concepts which are all "similar" in the sense that for every input almost all concepts in the set agree. Known results in learning theory state that such a concept class must require a large number of membership queries for exact learning. More formally, let $C' \subseteq C$ be any subset of C. For $a \in \{0,1\}^n$ and $b \in \{0,1\}$ let $C'_{\langle a,b\rangle}$ denote the set of those concepts in C' which assign label b to example a, i.e. $C'_{\langle a,b\rangle} = \{c \in C' : c(a) = b\}$. Let $\gamma^{C'}_{\langle a,b\rangle} = |C'_{\langle a,b\rangle}|/|C'|$ be the fraction of such concepts in C', and let $\gamma^{C'}_a = \min\{\gamma^{C'}_{\langle a,0\rangle}, \gamma^{C'}_{\langle a,1\rangle}\}$; thus $\gamma^{C'}_a$ is the minimum fraction of concepts in C' which can be eliminated by querying MQ_c on the string a. Let $\gamma^{C'} = \max\{\gamma^{C'}_a : a \in \{0,1\}^n\}$. Finally, let $\hat{\gamma}^C$ be the minimum of $\gamma^{C'}$ across all $C' \subseteq C$ such that $|C'| \geq 2$. Thus

$$\hat{\gamma}^C = \min_{C' \subseteq C, |C'| \geq 2} \; \max_{a \in \{0,1\}^n} \; \min_{b \in \{0,1\}} \frac{|{C'}_{\langle a,b\rangle}|}{|C'|}.$$

Intuitively, the inner min corresponds to the fact that the oracle may provide a worst-case response to any query; the max corresponds to the fact that the learning algorithm gets to choose the "best" query point a; and the outer min corresponds to the fact that the learner must succeed no matter what subset C' of C the target concept is drawn from. Thus $\hat{\gamma}^C$ is small if there is a large set C' of concepts which are all very similar in that any query eliminates only a few concepts from C'. If this is the case then many membership queries should be required to learn C; formally, we have the following lemma which is a variant of Fact 2 from [12]:

LEMMA 3.1. *Any (classical) exact learning algorithm for C must have sample complexity $\Omega(\frac{1}{\hat{\gamma}^C})$.*

Proof. Let $C' \subseteq C$, $|C'| \geq 2$ be such that $\gamma^{C'} = \hat{\gamma}^C$. Consider the following adversarial strategy for answering queries: given the query string a, answer the bit b which maximizes $\gamma^{C'}_{\langle a,b\rangle}$. This strategy ensures that each response eliminates at most a $\gamma^{C'}_a \leq \gamma^{C'} = \hat{\gamma}^C$ fraction of the concepts in C'. After $\frac{1}{2\hat{\gamma}^C} - 1$ membership queries, fewer than half of the concepts in C' have been eliminated, so at least two concepts have not yet been eliminated. Consequently, it is impossible for A to output a hypothesis which is equivalent to the correct concept with probability greater than 1/2. ☐

We now develop some tools which will enable us to prove a quantum version of Lemma 3.1. Let $C' \subseteq C, |C'| \geq 2$ be such that $\gamma^{C'} = \hat{\gamma}^C$ and let $c_1, \ldots, c_{|C'|}$ be a listing of the concepts in C'. Let the *typical concept for C'* be the function $\hat{c} : \{0,1\}^n \rightarrow \{0,1\}$ defined as follows: for all $a \in \{0,1\}^n$, $\hat{c}(a)$ is the bit b such that $|C'_{\langle a,b\rangle}| \geq |C'|/2$ (ties are broken arbitrarily; note that a tie occurs only if $\hat{\gamma}^C = 1/2$). The typical concept $\hat{c}$ need not belong to C' or even to C. The *difference matrix D* is the $|C'| \times 2^n$ zero/one matrix where rows are indexed by concepts in C', columns are indexed by strings in $\{0,1\}^n$, and $D_{i,x} = 1$ iff $c_i(x) \neq \hat{c}(x)$. By our choice of C' and the definition of $\hat{\gamma}^C$, each column of D has at most $|C'| \cdot \hat{\gamma}^C$ ones, so the L_1 matrix norm of D is $\|D\|_1 \leq |C'| \cdot \hat{\gamma}^C$.

Our quantum lower bound proof uses ideas which were first introduced by Bennett *et al.* [6]. Let $\mathcal{N}$ be a fixed quantum network architecture and let $U_0, O_1, \ldots, U_{T-1}$, O_T, U_T be the corresponding sequence of transformations. For $1 \leq t \leq T$ let $|\phi_t^c\rangle$ be the state of the quantum register after the transformations up through U_{t-1} have been performed (we refer to this stage of the computation as time t) if the oracle gate is QMQ_c. As in [6], for $x \in \{0,1\}^n$ let $q_x(|\phi_t^c\rangle)$, the *query magnitude of string x at time t with respect to c,* be the sum of the squared magnitudes in $|\phi_t^c\rangle$ of the basis states which are querying QMQ_c on string x at time t; so if $|\phi_t^c\rangle = \sum_{z\in\{0,1\}^m} \alpha_z|z\rangle$, then

$$q_x(|\phi_t^c\rangle) = \sum_{w\in\{0,1\}^{m-n}} \|\alpha_{xw}\|^2.$$

The quantity $q_x(|\phi_t^c\rangle)$ can be viewed as the amount of amplitude which the network $\mathcal{N}$ invests in the query string x to QMQ_c at time t. Intuitively, the final outcome of $\mathcal{N}$'s computation cannot depend very much on the oracle's responses to queries which have little amplitude invested in them. Bennett *et al.* formalized this intuition in the following theorem ([6], Theorem 3.3):

THEOREM 3.2. *Let $|\phi_t^c\rangle$ be defined as above. Let $F \subseteq \{0,\ldots,T-1\}\times\{0,1\}^n$ be a set of time-string pairs such that $\sum_{(t,x)\in F} q_x(|\phi_t^c\rangle) \leq \frac{\epsilon^2}{T}$. Now suppose the answer to each query instance $(t,x) \in F$ is modified to some arbitrary fixed bit $a_{t,x}$ (these answers need not be consistent with any oracle). Let $|\tilde{\phi}_t^c\rangle$ be the state of the quantum register at time t if the oracle responses are modified as stated above. Then $||\phi_T^c\rangle - |\tilde{\phi}_T^c\rangle| \leq \epsilon$.*

The following lemma, which is an extension of Corollary 3.4 from [6], shows that no quantum learning algorithm which makes few QMQ queries can effectively distinguish many concepts in C' from the typical concept $\hat{c}$.

LEMMA 3.3. *Fix any quantum network architecture $\mathcal{N}$ which has query complexity T. For all $\epsilon > 0$ there is a set $S \subseteq C'$ of cardinality at most $T^2|C'|\hat{\gamma}^C/\epsilon^2$ such that for all $c \in C' \setminus S$, we have $||\phi_T^{\hat{c}}\rangle - |\phi_T^c\rangle| \leq \epsilon$.*

Proof. Since $||\phi_t^{\hat{c}}\rangle| = 1$ for $t = 0,1,\ldots,T-1$, we have $\sum_{t=0}^{T-1}\sum_{x\in\{0,1\}^n} q_x(|\phi_t^{\hat{c}}\rangle) = T$. Let $q(|\phi_t^{\hat{c}}\rangle) \in \Re^{2^n}$ be the 2^n-dimensional vector which has entries indexed by strings $x \in \{0,1\}^n$ and which has $q_x(|\phi_t^{\hat{c}}\rangle)$ as its x-th entry. Note that the L_1 norm $\|q(|\phi_t^{\hat{c}}\rangle)\|_1$ is 1 for all $t = 0,\ldots,T-1$. For any $c_i \in C'$ let $q_{c_i}(|\phi_t^{\hat{c}}\rangle)$ be defined as $\sum_{x:c_i(x)\neq\hat{c}(x)} q_x(|\phi_t^{\hat{c}}\rangle)$. The quantity $q_{c_i}(|\phi_t^{\hat{c}}\rangle)$ can be viewed as the total query magnitude with respect to $\hat{c}$ at time t of those strings which distinguish c_i from $\hat{c}$. Note that $Dq(|\phi_t^{\hat{c}}\rangle) \in \Re^{|C'|}$ is an $|C'|$-dimensional vector whose i-th element is precisely $\sum_{x:c_i(x)\neq\hat{c}(x)} q_x(|\phi_t^{\hat{c}}\rangle) = q_{c_i}(|\phi_t^{\hat{c}}\rangle)$. Since $\|D\|_1 \leq |C'|\cdot\hat{\gamma}^C$ and $\|q(|\phi_t^{\hat{c}}\rangle)\|_1 = 1$, by the basic property of matrix norms we have that $\|Dq(|\phi_t^{\hat{c}}\rangle)\|_1 \leq |C'|\cdot\hat{\gamma}^C$, i.e. $\sum_{c_i\in C'} q_{c_i}(|\phi_t^{\hat{c}}\rangle) \leq |C'|\cdot\hat{\gamma}^C$. Hence

$$\sum_{t=0}^{T-1}\sum_{c_i\in C'} q_{c_i}(|\phi_t^{\hat{c}}\rangle) \leq T|C'|\cdot\hat{\gamma}^C.$$

If we let $S = \{c_i \in C' : \sum_{t=0}^{T-1} q_{c_i}(|\phi_t^{\hat{c}}\rangle) \geq \frac{\epsilon^2}{T}\}$, by Markov's inequality we have $|S| \leq T^2|C'|\hat{\gamma}^C/\epsilon^2$. Finally, if $c \notin S$ then $\sum_{t=0}^{T-1} q_c(|\phi_t^{\hat{c}}\rangle) \leq \frac{\epsilon^2}{T}$. Theorem 3.2 then implies that $||\phi_T^{\hat{c}}\rangle - |\phi_T^c\rangle| \leq \epsilon$. □

Now we can prove our quantum version of Lemma 3.1.

THEOREM 3.4. *Any quantum exact learning algorithm for C must have sample complexity* $\Omega\left(\left(\frac{1}{\hat{\gamma}^C}\right)^{1/2}\right)$.

Proof. Suppose that $\mathcal{N}$ is a quantum exact learning algorithm for $\mathcal{C}$ which makes at most $T = \frac{1}{64} \cdot \left(\frac{1}{\hat{\gamma}^C}\right)^{1/2}$ quantum membership queries. If we take $\epsilon = \frac{1}{32}$, then Lemma 3.3 implies that there is a set $S \subset C'$ of cardinality at most $\frac{|C'|}{4}$ such that for all $c \in C' \setminus S$ we have $\left| |\phi_T^c\rangle - |\phi_T^{\hat{c}}\rangle \right| \leq \frac{1}{32}$. Let c_1, c_2 be any two concepts in $C' \setminus S$. By Fact 1, the probability that $\mathcal{N}$ outputs a circuit equivalent to c_1 can differ by at most $\frac{1}{8}$ if $\mathcal{N}$'s oracle gates are $QMQ_{\hat{c}}$ as opposed to QMQ_{c_1}, and likewise for $QMQ_{\hat{c}}$ versus QMQ_{c_2}. It follows that the probability that $\mathcal{N}$ outputs a circuit equivalent to c_1 can differ by at most $\frac{1}{4}$ if $\mathcal{N}$'s oracle gates are QMQ_{c_1} as opposed to QMQ_{c_2}, but this contradicts the assumption that $\mathcal{N}$ is a quantum exact learning algorithm for C. $\square$

Well known results [9, 24] show that $O(\sqrt{N})$ queries are sufficient to search a quantum database of N unordered items for a desired item. These upper bounds can easily be used to show that Theorem 3.4 is tight up to constant factors.

A Lower Bound Based on Concept Class Size. A second reason why a concept class can require many membership queries is its size. Angluin [2] has given the following simple bound, incomparable to the bound of Lemma 3.1, on the number of classical membership queries required for exact learning:

LEMMA 3.5. *Any classical exact learning algorithm for C must have sample complexity* $\Omega(\log |C|)$.

Proof. Consider the following adversarial strategy for answering queries: if $C' \subseteq C$ is the set of concepts which have not yet been eliminated by previous responses to queries, then given the query string a, answer the bit b such that $\gamma^{C'}_{\langle a,b\rangle} \geq \frac{1}{2}$. Under this strategy, after $\log |C| - 1$ membership queries at least two possible target concepts will remain. $\square$

In this section we prove a variant of this lemma for the quantum model. Our proof uses ideas from [5] so we introduce some of their notation. Let $N = 2^n$. For each concept $c \in C$, let $X^c = (X_0^c, \ldots, X_{N-1}^c) \in \{0,1\}^N$ be a vector which represents c as an N-tuple, i.e. $X_i^c = c(x^i)$ where $x^i \in \{0,1\}^n$ is the binary representation of i. From this perspective we may identify C with a subset of $\{0,1\}^N$, and we may view a QMQ_c gate as a black-box oracle for X^c which maps basis state $|x^i, b, y\rangle$ to $|x^i, b \oplus X_i^c, y\rangle$.

Using ideas from [20, 21], Beals *et al.* have proved the following useful lemma, which relates the query complexity of a quantum network to the degree of a certain polynomial ([5], Lemma 4.2):

LEMMA 3.6. *Let $\mathcal{N}$ be a quantum network that makes T queries to a black-box X, and let $B \subseteq \{0,1\}^m$ be a set of basis states. Then there exists a real-valued multilinear polynomial $P_B(X)$ of degree at most $2T$ which equals the probability that observing the final state of the network with black-box X yields a state from B.*

We use Lemma 3.6 to prove the following quantum lower bound based on concept class size. (Ronald de Wolf has observed that this lower bound can also be obtained from the results of [19].)

THEOREM 3.7. *Any exact quantum learning algorithm for C must have sample complexity* $\Omega\left(\frac{\log |C|}{n}\right)$.

Proof. Let $\mathcal{N}$ be a quantum network which learns C and has query complexity

T. For all $c \in C$ we have the following: if $\mathcal{N}$'s oracle gates are QMQ_c gates, then with probability at least 2/3 the output of $\mathcal{N}$ is a representation of a Boolean circuit h which computes c. Let $c_1, \ldots, c_{|C|}$ be all of the concepts in C, and let $X^1, \ldots, X^{|C|}$ be the corresponding vectors in $\{0,1\}^N$. For all $i = 1, \ldots, |C|$ let $B_i \subseteq \{0,1\}^m$ be the collection of those basis states which are such that if the final observation performed by $\mathcal{N}$ yields a state from B_i, then the output of $\mathcal{N}$ is a representation of a Boolean circuit which computes c_i. Clearly for $i \neq j$ the sets B_i and B_j are disjoint. By Lemma 3.6, for each $i = 1, \ldots, |C|$ there is a real-valued multilinear polynomial P_i of degree at most $2T$ such that for all $j = 1, \ldots, |C|$, the value of $P_i(X^j)$ is precisely the probability that the final observation on $\mathcal{N}$ yields a representation of a circuit which computes c_i, provided that the oracle gates are QMQ_{c_j} gates. The polynomials P_i thus have the following properties:

1. $P_i(X^i) \geq 2/3$ for all $i = 1, \ldots, |C|$;
2. For any $j = 1, \ldots, |C|$, we have $\sum_{i \neq j} P_i(X^j) \leq 1/3$ (since the total probability across all possible observations is 1).

Let $N_0 = \sum_{i=0}^{2T} \binom{N}{i}$. For any $X = (X_0, \ldots, X_{N-1}) \in \{0,1\}^N$ let $\tilde{X} \in \{0,1\}^{N_0}$ be the column vector which has a coordinate for each monic multilinear monomial over $X_0, \ldots, X_{N-1}$ of degree at most $2T$. Thus, for example, if $N = 4$ and $2T = 2$ we have $X = (X_0, X_1, X_2, X_3)$ and

$$\tilde{X}^t = (1, X_0, X_1, X_2, X_3, X_0X_1, X_0X_2, \\ X_0X_3, X_1X_2, X_1X_3, X_2X_3).$$

If V is a column vector in $\Re^{N_0}$, then $V^t\tilde{X}$ corresponds to the degree-$2T$ polynomial whose coefficients are given by the entries of V. For $i = 1, \ldots, |C|$ let $V_i \in \Re^{N_0}$ be the column vector which corresponds to the coefficients of the polynomial P_i. Let M be the $|C| \times N_0$ matrix whose i-th row is V_i^t; note that multiplication by M defines a linear transformation from $\Re^{N_0}$ to $\Re^{|C|}$. Since $V_i^t\tilde{X}^j$ is precisely $P_i(X^j)$, the product $M\tilde{X}^j$ is a column vector in $\Re^{|C|}$ which has $P_i(X^j)$ as its i-th coordinate.

Now let L be the $|C| \times |C|$ matrix whose j-th column is the vector $M\tilde{X}^j$. A square matrix A is said to be *diagonally dominant* if $|a_{ii}| > \sum_{j \neq i} |a_{ij}|$ for all i. Properties (1) and (2) above imply that the transpose of L is diagonally dominant. It is well known that any diagonally dominant matrix must be of full rank (see e.g. [32]). Since L is full rank and each column of L is in the image of M, it follows that the image under M of $\Re^{N_0}$ is all of $\Re^{|C|}$, and hence $N_0 \geq |C|$. Finally, since $N_0 = \sum_{i=0}^{2T} \binom{N}{i} \leq N^{2T}$, we have $T \geq \frac{\log |C|}{2 \log N} = \frac{\log |C|}{2n}$, which proves the theorem. □

The lower bound of Theorem 3.7 is nearly tight as witnessed by the following example: let C be the collection of all 2^n parity functions over $\{0,1\}^n$, so each function in C is defined by a string $a \in \{0,1\}^n$ and $c_a(x) = a \cdot x$. The quantum algorithm which solves the well-known Deutsch-Jozsa problem [17] can be used to exactly identify a and thus learn the target concept with probability 1 from a single query. It follows that the factor of n in the denominator of Theorem 3.7 cannot be replaced by any function $g(n) = o(n)$.

3.3. Quantum and Classical Exact Learning are Equivalent. We have seen two different reasons why exact learning a concept class can require a large number of classical membership queries: the class may contain many similar concepts (i.e. $\hat{\gamma}^C$ is small), or the class may contain very many concepts (i.e. $\log |C|$ is large). The following lemma, which is a variant of Theorem 3.1 from [26], shows that these are the only reasons why many membership queries may be required:

LEMMA 3.8. *There is an exact learning algorithm for C which has sample complexity $O((\log|C|)/\hat{\gamma}^C)$.*

Proof. Consider the following learning algorithm A: at each stage in its execution, if C' is the set of concepts in C which have not yet been eliminated by previous responses to queries, algorithm A's next query string is the string $a \in \{0,1\}^n$ which maximizes $\gamma_a^{C'}$. By following this strategy, each query response received from the oracle must eliminate at least a $\gamma^{C'}$ fraction of the set C', so with each query the size of the set of possible target concepts is multiplied by a factor which is at most $1 - \gamma^{C'} \leq 1 - \hat{\gamma}^C$. Consequently, after $O((\log|C|)/\hat{\gamma}^C)$ queries, only a single concept will not have been eliminated; this concept must be the target concept, so A can output a hypothesis h which is equivalent to c. □

Combining Theorem 3.4, Theorem 3.7 and Lemma 3.8 we obtain the following relationship between the quantum and classical sample complexity of exact learning:

Theorem 1.1 *Let $\mathcal{C}$ be any concept class over $\{0,1\}^n$ and let D and Q be such that $\mathcal{C}$ is exact learnable from D classical membership queries or from Q quantum membership queries. Then $D = O(nQ^3)$.*

We note that a QMQ_c oracle can clearly be used to simulate an MQ_c oracle, so $Q \leq D$ as well.

3.4. Discussion. Theorem 1.1 provides an interesting contrast to several known results for black-box quantum computation. Let F denote the set of all 2^{2^n} functions from $\{0,1\}^n$ to $\{0,1\}$. Beals *et al.* [5] have shown that if $f : F \rightarrow \{0,1\}$ is any total function (i.e. $f(c)$ is defined for every possible concept c over $\{0,1\}^n$), then the query complexity of any quantum network which computes f is polynomially related to the number of classical black-box queries required to compute f. Their result is interesting because it is well known [7, 11, 17, 40] that for certain concept classes $C \subset F$ and partial functions $f : C \rightarrow \{0,1\}$, the quantum black-box query complexity of f can be exponentially smaller than the classical black-box query complexity.

Our Theorem 1.1 provides a sort of dual to the results of Beals *et al.*: their bound on query complexity holds only for the fixed concept class F but for any function $f : F \rightarrow \{0,1\}$, while our bound holds for any concept class $C \subseteq F$ but only for the fixed problem of exact learning. In general, the problem of computing a function $f : C \rightarrow \{0,1\}$ from black-box queries can be viewed as an easier version of the corresponding exact learning problem: instead of having to figure out only one bit of information about the unknown concept c (the value of f), for the learning problem the algorithm must identify c exactly. Theorem 1.1 shows that for this more demanding problem, unlike the results in [7, 11, 17, 40] there is no way of restricting the concept class C so that learning becomes substantially easier in the quantum setting than in the classical setting.

4. PAC Learning from a Quantum Example Oracle.

4.1. The Quantum Example Oracle. Bshouty and Jackson [13] have introduced a natural quantum generalization of the standard PAC-model example oracle. While a standard PAC example oracle $EX(c, \mathcal{D})$ generates each example $\langle x, c(x)\rangle$ with probability $\mathcal{D}(x)$, where $\mathcal{D}$ is a distribution over $\{0,1\}^n$, a *quantum PAC example oracle* $QEX(c, \mathcal{D})$ generates a superposition of all labeled examples, where each labeled example $\langle x, c(x)\rangle$ appears in the superposition with amplitude proportional to the square root of $\mathcal{D}(x)$. More formally, a $QEX(c, \mathcal{D})$ gate maps the initial basis state $|0^n, 0\rangle$ to the state $\sum_{x\in\{0,1\}^n} \sqrt{\mathcal{D}(x)}|x, c(x)\rangle$. (We leave the action of a $QEX(c, \mathcal{D})$

gate undefined on other basis states, and stipulate that any quantum network which includes T $QEX(c,\mathcal{D})$ gates must have all T gates at the "bottom of the circuit," i.e. no gate may occur on any wire between the inputs and any $QEX(c,\mathcal{D})$ gate.) A quantum network with T $QEX(c,\mathcal{D})$ gates is said to be a QEX network with *query complexity* T.

A *quantum PAC learning algorithm for* $\mathcal{C}$ is a family $\{\mathcal{N}_{(n,\epsilon,\delta)} : n \geq 1,\ 0 < \epsilon, \delta < 1\}$ of QEX networks with the following property: for all $n \geq 1$ and $0 < \epsilon, \delta < 1$, for all $c \in C_n$, for all distributions $\mathcal{D}$ over $\{0,1\}^n$, if the network $\mathcal{N}_{(n,\epsilon,\delta)}$ has all its oracle gates instantiated as $QEX(c,\mathcal{D})$ gates, then with probability at least $1-\delta$ the network $\mathcal{N}_{(n,\epsilon,\delta)}$ outputs a representation of a circuit h which is an ϵ-approximator to c under $\mathcal{D}$. The *quantum sample complexity* $T(n,\epsilon,\delta)$ of a quantum PAC algorithm is the query complexity of $\mathcal{N}_{(n,\epsilon,\delta)}$.

4.2. Lower Bounds on Classical and Quantum PAC Learning. Throughout this section for ease of notation we omit the subscript n and write C for C_n. We view each concept $c \in C$ as a subset of $\{0,1\}^n$. For $S \subseteq \{0,1\}^n$, we write $\Pi_C(S)$ to denote $\{c \cap S : c \in C\}$, so $|\Pi_C(S)|$ is the number of different "dichotomies" which the concepts in C induce on the points in S. A subset $S \subseteq \{0,1\}^n$ is said to be *shattered* by C if $|\Pi_C(S)| = 2^{|S|}$, i.e. if C induces every possible dichotomy on the points in S. The *Vapnik-Chervonenkis dimension of* C, VC-DIM(C), is the size of the largest subset $S \subseteq \{0,1\}^n$ which is shattered by C.

Well known results in computational learning theory show that the Vapnik-Chervonenkis dimension of a concept class C characterizes the number of calls to $EX(c,\mathcal{D})$ which are information-theoretically necessary and sufficient to PAC learn C. For the lower bound, the following theorem is a slight simplification of a result due to Blumer *et al.* ([8], Theorem 2.1.ii.b):

THEOREM 4.1. *Let C be any concept class and d = VC-DIM(C). Then any (classical) PAC learning algorithm for $\mathcal{C}$ must have sample complexity* $\Omega(d)$.

Proof sketch. The idea behind Theorem 4.1 is to consider the distribution $\mathcal{D}$ which is uniform over some shattered set S of size d and assigns zero weight to points outside of S. Any learning algorithm which makes only $d/2$ calls to $EX(c,\mathcal{D})$ will have no information about the value of c on at least $d/2$ points in S; moreover, since the set S is shattered by C, any labeling is possible for these unseen points. Since the error of any hypothesis h under $\mathcal{D}$ is the fraction of points in S where h and the target concept disagree, a simple analysis shows that no learning algorithm which perform only $d/2$ calls to $EX(c,\mathcal{D})$ can have high probability (e.g. $1-\delta = 2/3$) of generating a low-error hypothesis (e.g. $\epsilon = 1/10$). □

We now give a quantum analogue of the classical lower bound given by Theorem 4.1:

THEOREM 4.2. *Let $\mathcal{C}$ be any concept class and d = VC-DIM(C). Then any quantum PAC learning algorithm for $\mathcal{C}$ must have quantum sample complexity* $\Omega(\frac{d}{n})$.

Proof. Let $S = \{x^1, \ldots, x^d\}$ be a set which is shattered by C and let $\mathcal{D}$ be the distribution which is uniform on S and assigns zero weight to points outside S. If $h : \{0,1\}^n \rightarrow \{0,1\}$ is a Boolean function on $\{0,1\}^n$, we say that the *relative distance of h and c on S* is the fraction of points in S on which h and c disagree. We will prove the following result which is stronger than Theorem 4.2: Let $\mathcal{N}$ be a quantum network with QMQ gates such that for all $c \in C$, if $\mathcal{N}$'s oracle gates are QMQ_c gates, then with probability at least $2/3$ the output of $\mathcal{N}$ is a hypothesis h such that the relative distance of h and c on S is at most $1/10$. We will show that such a network $\mathcal{N}$ must have query complexity at least $\frac{d}{12n}$. Since any QEX network with query complexity T can be simulated by a QMQ network with query complexity T, taking $\epsilon = 1/10$ and $\delta = 1/3$ will prove Theorem 4.2.

The argument is a modification of the proof of Theorem 3.7 using ideas from error correcting codes. Let $\mathcal{N}$ be a quantum network with query complexity T which satisfies the following condition: for all $c \in C$, if $\mathcal{N}$'s oracle gates are QMQ_c gates, then with probability at least 2/3 the output of $\mathcal{N}$ is a representation of a Boolean circuit h such that the relative distance of h and c on S is at most 1/10. By the well-known Gilbert-Varshamov bound from coding theory (see, e.g., Theorem 5.1.7 of [42]), 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 $d/4$ bit positions, where

$$A \geq \frac{2^d}{\sum_{i=0}^{d/4-1} \binom{d}{i}} \geq \frac{2^d}{\sum_{i=0}^{d/4} \binom{d}{i}} \geq 2^{d(1-H(1/4))} > 2^{d/6}.$$

(Here $H(p) = -p \log p - (1-p) \log(1-p)$ is the binary entropy function.) For each $i = 1, \ldots, A$ let $c_i \in C$ be a concept such that the d-bit string $c_i(x^1) \cdots c_i(x^d)$ is s^i (such a concept c_i must exist since the set S is shattered by C).

For $i = 1, \ldots, A$ let $B_i \subseteq \{0,1\}^m$ be the collection of those basis states which are such that if the final observation performed by $\mathcal{N}$ yields a state from B_i, then the output of $\mathcal{N}$ is a hypothesis h such that h and c_i have relative distance at most 1/10 on S. Since each pair of concepts c_i, c_j has relative distance at least 1/4 on S, the sets B_i and B_j are disjoint for all $i \neq j$.

As in Section 3.2 let $N = 2^n$ and let $X^j = (X_0^j, \ldots, X_{N-1}^j) \in \{0,1\}^n$ where X^j is the N-tuple representation of the concept c_j. By Lemma 3.6, for each $i = 1, \ldots, A$ there is a real-valued multilinear polynomial P_i of degree at most $2T$ such that for all $j = 1, \ldots, A$, the value of $P_i(X^j)$ is precisely the probability that the final observation on $\mathcal{N}$ yields a state from B_i provided that the oracle gates are QMQ_{c_j} gates. Since, by assumption, if c_i is the target concept then with probability at least 2/3 $\mathcal{N}$ generates a hypothesis which has relative distance at most 1/10 from c_i on S, the polynomials P_i have the following properties:

1. $P_i(X^i) \geq 2/3$ for all $i = 1, \ldots, A$;
2. For any $j = 1, \ldots, A$ we have that $\sum_{i \neq j} P_i(X^j) \leq 1/3$ (since the B_i's are disjoint and the total probability across all observations is 1).

Let N_0 and $\tilde{X}$ be defined as in the proof of Theorem 3.7. For $i = 1, \ldots, A$ let $V_i \in \Re^{N_0}$ be the column vector which corresponds to the coefficients of the polynomial P_i, so $V_i^t \tilde{X} = P_i(X)$. Let M be the $A \times N_0$ matrix whose i-th row is the vector V_i^t, so multiplication by M is a linear transformation from $\Re^{N_0}$ to $\Re^A$. The product $M\tilde{X}^j$ is a column vector in $\Re^A$ which has $P_i(X)$ as its i-th coordinate.

Now let L be the $A \times A$ matrix whose j-th column is the vector $M\tilde{X}^j$. As in Theorem 3.7 we have that the transpose of L is diagonally dominant, so L is of full rank and hence $N_0 \geq A$. Since $A \geq 2^{d/6}$ we thus have that $T \geq \frac{d/6}{2 \log_2 N} = \frac{d}{12n}$, and the theorem is proved. □

Since the class of parity functions over $\{0,1\}^n$ has VC-dimension n, as in Theorem 3.7 the n in the denominator of Theorem 4.2 cannot be replaced by any function $g(n) = o(n)$.

4.3. Quantum and Classical PAC Learning are Equivalent. A well-known theorem due to Blumer *et al.* (Theorem 3.2.1.ii.a of [8]) shows that VC-DIM(C) also upper bounds the number of $EX(c, \mathcal{D})$ calls required for (classical) PAC learning:

THEOREM 4.3. *Let C be any concept class and $d =$ VC-DIM(C). There is a classical PAC learning algorithm for C which has sample complexity $O(\frac{1}{\epsilon}\log\frac{1}{\delta} + \frac{d}{\epsilon}\log\frac{1}{\epsilon})$.*

The proof of Theorem 4.3 is quite complex so we do not attempt to sketch it. As in Section 3.3, this upper bound along with our lower bound from Theorem 4.2 together yield:

Theorem 1.2 *Let $\mathcal{C}$ be any concept class over $\{0,1\}^n$ and let D and Q be such that $\mathcal{C}$ is PAC learnable from D classical examples or from Q quantum examples. Then $D = O(nQ)$.*

We note that a $QEX(c,\mathcal{D})$ oracle can be used to simulate the corresponding $EX(c,\mathcal{D})$ oracle by immediately performing an observation on the QEX gate's outputs[3] (such an observation yields each example $\langle x, c(x)\rangle$ with probability $\mathcal{D}(x)$), and thus $Q \leq D$.

5. Quantum versus Classical Efficient Learnability. We have shown that from an information-theoretic perspective, up to polynomial factors quantum learning is no more powerful than classical learning. However, we now observe that the apparant *computational* advantages of the quantum model yield efficient quantum learning algorithms which seem to have no efficient classical counterparts.

A *Blum integer* is an integer $N = pq$ where $p \neq q$ are ℓ-bit primes each congruent to 3 modulo 4. It is widely believed that there is no polynomial-time classical algorithm which can successfully factor a randomly selected Blum integer with non-negligible success probability.

Kearns and Valiant [28] have constructed a concept class $\mathcal{C}$ whose PAC learnability is closely related to the problem of factoring Blum integers. In their construction each concept $c \in \mathcal{C}$ is uniquely defined by some Blum integer N. Furthermore, c has the property that if $c(x) = 1$ then the prefix of x is the binary representation of N. Kearns and Valiant prove that if there is a polynomial time PAC learning algorithm for $\mathcal{C}$, then there is a polynomial time algorithm which factors Blum integers. Thus, assuming that factoring Blum integers is a computationally hard problem for classical computation, the Kearns-Valiant concept class $\mathcal{C}$ is not efficiently PAC learnable.

On the other hand, in a celebrated result Shor [39] has exhibited a poly(n) size quantum network which can factor any n-bit integer with high success probability. Since each positive example of a concept $c \in \mathcal{C}$ reveals the Blum integer N which defines c, using Shor's algorithm it is easy to obtain an efficient quantum PAC learning algorithm for the Kearns-Valiant concept class. We thus have

OBSERVATION 2. *If there is no polynomial-time classical algorithm for factoring Blum integers, then there is a concept class $\mathcal{C}$ which is efficiently quantum PAC learnable but not efficiently classically PAC learnable.*

The hardness results of Kearns and Valiant were later extended by Angluin and Kharitonov [3]. Using a public-key encryption system which is secure against chosen-cyphertext attack (based on the assumption that factoring Blum integers is computationally hard for polynomial-time algorithms), they constructed a concept class $\mathcal{C}$ which cannot be learned by any polynomial-time learning algorithm which makes membership queries. As with the Kearns-Valiant concept class, though, using Shor's quantum factoring algorithm it is possible to construct an efficient quantum exact learning algorithm for this concept class. Thus, for the exact learning model as well, we have:

[3] As noted in Section 2.3, intermediate observations during a computation can always be simulated by a single observation at the end of the computation.

Observation 3. *If there is no polynomial-time classical algorithm for factoring Blum integers, then there is a concept class $\mathcal{C}$ which is efficiently quantum exact learnable from membership queries but not efficiently classically exact learnable from membership queries.*

In the next sections we prove Theorem 1.3 which establishes a stronger computational separation between the quantum and classical models of exact learning from membership queries than is implied by Observation 3. The proof involves Simon's quantum oracle algorithm, which we briefly describe in the next section.

6. Simon's Algorithm. Let $f : \{0,1\}^n \rightarrow \{0,1\}^n$ be a function and let $0^n \neq s \in \{0,1\}^n$. We say that f is *two-to-one with XOR mask* s if for all $y \neq x$, $f(x) = f(y) \iff y = x \oplus s$. More generally, f is *invariant under XOR mask with* s if $f(x) = f(x \oplus s)$ for all $x \in \{0,1\}^n$ (note that such a function need not be two-to-one).

Simon [40] has given a simple quantum algorithm which takes oracle access to a function $f : \{0,1\}^n \rightarrow \{0,1\}^n$, runs in poly$(n)$ time, and behaves as follows:

1. If f is a permutation on $\{0,1\}^n$, the algorithm outputs an n-bit string y which is uniformly distributed over $\{0,1\}^n$.
2. If f is two-to-one with XOR mask s, the algorithm outputs an n-bit string y which is uniformly distributed over the 2^{n-1} strings such that $y \cdot s = 0$.
3. If f is invariant under XOR mask with s, the algorithm outputs some n-bit string y which satisfies $y \cdot s = 0$.

Simon showed that by running this procedure $O(n)$ times a quantum algorithm can distinguish between Case 1 (f is a permutation) and Case 3 (f is invariant under some XOR mask) with high probability. In Case 1 after $O(n)$ repetitions the strings obtained will with probability $1 - 2^{-O(n)}$ contain a basis for the vector space $(Z_2)^n$ (here we are viewing n-bit strings as vectors over Z_2), while in Case 3 the strings obtained cannot contain such a basis since each string must lie in the subspace $\{y : y \cdot s = 0\}$. Simon also observed that in Case 2 (f is two-to-one with XOR mask s) the algorithm can be used to efficiently identify s with high probability. This is because after $O(n)$ repetitions, with high probability s will be the unique nonzero vector whose dot product with each y is 0; this vector can be found by solving the linear system defined by the y's.

Simon also analyzed the success probability of classical oracle algorithms for this problem. His analysis establishes the following theorem:

Theorem 6.1. *Let $0^n \neq s \in \{0,1\}^n$ be chosen uniformly and let $f : \{0,1\}^n \rightarrow \{0,1\}^n$ be an oracle chosen uniformly from the set of all functions which are two-to-one with XOR mask s. Then (i) there is a polynomial-time quantum oracle algorithm which identifies s with high probability; (ii) any p.p.t. classical oracle algorithm identifies s with probability* $1/2^{\Omega(n)}$.

This surprising ability of quantum oracle algorithms to efficiently find s is highly suggestive in the context of our search for a learning problem which separates efficient classical and quantum computation. Indeed, Simon's algorithm will play a crucial role in establishing that the concept class which we construct in Section 8 is learnable in poly(n) time by a quantum algorithm. Recall that in our learning scenario, though, the goal is to *exactly identify* the unknown target function, not just to identify the string s. Since $2^{\Omega(n)}$ bits are required to specify a randomly chosen function f which is two-to-one with XOR mask s, no algorithm (classical or quantum) can output a description of f in poly(n) time, much less learn f in poly(n) time. Thus it will not do to use truly random functions for our learning problem; instead we use *pseudorandom* functions as described in the next section.

7. Pseudorandomness. A *pseudorandom function family* [23] is a collection of functions $\{f_s : \{0,1\}^{|s|} \to \{0,1\}^{|s|}\}_{s\in\{0,1\}^*}$ with the following two properties:

- *(efficient evaluation)* there is a deterministic algorithm which, given an n-bit seed s and an n-bit input x, runs in time poly(n) and outputs $f_s(x)$;
- *(pseudorandomness)* for all polynomials Q, all p.p.t. oracle algorithms M, and all sufficiently large n, we have that

$$\left| \Pr_{F\in\mathcal{F}_n}[M^F \text{ outputs } 1] - \Pr_{s\in\{0,1\}^n}[M^{f_s} \text{ outputs } 1] \right| < \frac{1}{Q(n)}.$$

Intuitively, the pseudorandomess property ensures that in any p.p.t. computation which uses a truly random function, a randomly chosen pseudorandom function may be used instead without affecting the outcome in a noticeable way. Well known results [23, 25] imply that pseudorandom function families exist if and only if any one-way function exists.

A *pseudorandom permutation family* is a pseudorandom function family with the added property that each function $f_s : \{0,1\}^{|s|} \to \{0,1\}^{|s|}$ is a permutation. Luby and Rackoff [30] gave the first construction of a pseudorandom permutation family from any pseudorandom function family. In their construction each permutation $f_s : \{0,1\}^n \to \{0,1\}^n$ has a seed s of length $|s| = 3n/2$ rather than n as in our definition above. Subsequent constructions [33, 34, 35] of pseudorandom permutation families $\{f_s : \{0,1\}^n \to \{0,1\}^n\}$ use n-bit seeds and hence match our definition exactly. (Our definition of pseudorandomness could easily be extended to allow seed lengths other than n. For our construction in Section 8 it will be convenient to have n-bit seeds.)

7.1. Pseudorandom Functions Invariant under XOR Mask. Our main cryptographic result, stated below, is proved in Appendix A:

THEOREM 7.1. *If any one-way function exists, then there is a pseudorandom function family $\{g_s : \{0,1\}^{|s|} \to \{0,1\}^{|s|}\}$ such that $g_s(x) = g_s(x\oplus s)$ for all $|x| = |s|$.*

A first approach to constructing such a family is as follows: given any pseudorandom function family $\{f_s\}$, let $\{g_s\}$ be defined by

$$g_s(x) \stackrel{\text{def}}{=} f_s(x)\oplus f_s(x\oplus s). \tag{7.1}$$

This simple construction ensures that each function g_s is invariant under XOR mask with s, but the family $\{g_s\}$ need not be pseudorandom just because $\{f_s\}$ is pseudorandom. Indeed, if $\{h_s\}$ is similarly defined by $h_s(x) \stackrel{\text{def}}{=} g_s(x)\oplus g_s(x\oplus s)$, then $\{h_s\}$ is not pseudorandom since

$$h_s(x) \;=\; (f_s(x)\oplus f_s(x\oplus s))\oplus(f_s(x\oplus s)\oplus f_s(x\oplus s\oplus s)) \;=\; 0^n.$$

While this example shows that (1) does not always preserve pseudorandomness, it leaves open the possibility that (1) may preserve pseudorandomness for certain function families $\{f_s\}$. In Appendix A we show that if $\{f_s\}$ is a pseudorandom function family which is constructed from any one-way function in a particular way, then the family $\{g_s\}$ defined by (1) is indeed pseudorandom.

It may at first appear that the pseudorandom function family $\{g_s\}$ given by Theorem 7.1 immediately yields a concept class which separates efficient quantum learning from efficient classical learning. The pseudorandomness of $\{g_s\}$ ensures that no p.p.t. algorithm can learn successfully; on the other hand, if Simon's quantum algorithm is given oracle access to a function which is two-to-one with XOR mask s, then it can efficiently find s with high probability. Hence it may seem that given access to g_s Simon's quantum algorithm can efficiently identify the seed s and thus learn the target concept.

The flaw in this argument is that each function g_s from Theorem 7.1, while invariant under XOR mask with s, need not be two-to-one. Indeed g_s could conceivably be invariant under XOR mask with, say, $\sqrt{n}$ linearly independent strings $s = s^1, s^2, \ldots, s^{\sqrt{n}}$. Such a set of strings spans a $2^{\sqrt{n}}$-element subspace of $\{0,1\}^n$; even if Simon's algorithm could identify this subspace, it would not indicate which element of the subspace is the true seed s. Hence a more sophisticated construction is required.

8. Proof of Theorem 1.

8.1. The Concept Class C. We describe concepts over $\{0,1\}^m$ where $m = n + 2\log n + 1$. Each concept in C_m is defined by an $(n+1)$-tuple $(y, s^1, \ldots, s^n)$ where $y = y_1 \ldots y_n \in \{0,1\}^n$ and each $s^i \in \{0,1\}^n \setminus \{0^n\}$, so C_m contains $2^n(2^n - 1)^n$ distinct concepts. For brevity we write $\tilde{s}$ to stand for $s^1, \ldots, s^n$ below.

Roughly speaking, each concept in C_m comprises n pseudorandom functions; as explained below the string y acts as a "password" and the strings $s^1, \ldots, s^n$ are the seeds to the pseudorandom functions. Each concept $c \in C_m$ takes m-bit strings as inputs; we view such an m-bit input as a 4-tuple (b, x, i, j) where $b \in \{0,1\}$, $x \in \{0,1\}^n$, and $i, j \in \{0,1\}^{\log n}$ each represent a number in the range $\{1, 2, \ldots, n\}$.

Let $\{h_s^0 : \{0,1\}^{|s|} \rightarrow \{0,1\}^{|s|}\}_{s \in \{0,1\}^*}$ be a pseudorandom permutation family and let $\{h_s^1 : \{0,1\}^{|s|} \rightarrow \{0,1\}^{|s|}\}_{s \in \{0,1\}^*}$ be the pseudorandom function family from Theorem 7.1, so $h_s^1(x) = h_s^1(x \oplus s)$. The concept $c_{y,\tilde{s}}$ is defined as follows on input (b, x, i, j) :

- **If $b = 0$**: A query $(0, x, i, j)$ is called a *function query.* The value of $c_{y,\tilde{s}}(0, x, i, j)$ is $h_{s^i}^{y_i}(x)_j$, i.e. the j-th bit of the n-bit string $h_{s^i}^{y_i}(x)$. Thus the bit y_i determines whether the i-th pseudorandom function used is a permutation or is invariant under XOR mask with s^i.
- **If $b = 1$**: A query $(1, x, i, j)$ is called a *seed query.* The value of $c_{y,\tilde{s}}(1, x, i, j)$ is 0 if $x \neq y$ and is s_j^i (the j-th bit of the i-th seed s^i) if $x = y$.

The intuition behind our construction is simple: in order to learn the target concept successfully a learning algorithm must identify each seed string $s^1, \ldots, s^n$. These strings can be identified by making seed queries $(1, y, i, j)$, but in order to make the correct seed queries the learning algorithm must know y. Since each bit y_i corresponds to whether an oracle is a permutation or is XOR-mask invariant, a quantum algorithm can determine each y_i and thus can learn successfully. However, no p.p.t. algorithm can distinguish between these two types of oracles (since in either case the oracle is pseudorandom and hence is indistinguishable from a truly random function), so no p.p.t. algorithm can learn y.

8.2. A Quantum Algorithm Which Learns C in Polynomial Time. The main result of this section is the following:

Theorem 8.1. *The concept class C described above is polynomial-time learnable from quantum membership queries.*

Proof. Let $c_{y,\tilde{s}} \in C_m$ be the target concept. Each function $h_{s^i}^{y_i}$ is a permutation iff $y_i = 0$ and is XOR-mask invariant iff $y_i = 1$ (this is why we do not allow $s^i = 0^n$ in the definition of the concept class). Using quantum membership queries, a poly(n)-time quantum algorithm can run Simon's procedure n times, once for each function $h_{s^i}^{y_i}$, and thus determine each bit y_i with high probability. (One detail which arises here is that Simon's algorithm uses an oracle $\{0,1\}^n \to \{0,1\}^n$ whereas in our learning setting the oracle outputs one bit at a time. This is not a problem since it is possible to simulate any call to Simon's oracle by making n sequential calls, bit by bit, to our oracle.) Given the string $y = y_1 \dots y_n$, the algorithm can then make n^2 queries on inputs $(1, y, i, j)$ for $1 \le i, j \le n$ to learn each of the n strings $s^1, \dots, s^n$. Once y and $s^1, \dots, s^n$ are known it is straightforward to output a circuit for $c_{y,\tilde{s}}$. □

8.3. No Classical Algorithm Learns C in Polynomial Time. The main result of this section is the following:

THEOREM 8.2. *C is not polynomial-time learnable from classical membership queries.*

Let $C'_m \supset C_m$, $|C'_m| = 2^{n^2+n}$ be the concept class $C'_m = \{c_{y,\tilde{s}} : y, s^1, \dots, s^n \in \{0,1\}^n\}$; thus C'_m includes concepts in which s^i may be 0^n. The following lemma states that it is hard to learn a target concept chosen uniformly from C'_m :

LEMMA 8.3. *For all polynomials Q, all p.p.t. learning algorithms A, and all sufficiently large n,*

$$\Pr_{c_{y,\tilde{s}} \in C'_m}[A^{c_{y,\tilde{s}}} \textit{ outputs a hypothesis } h \equiv c_{y,\tilde{s}}] < \frac{1}{Q(n)}.$$

To see that Lemma 8.3 implies Theorem 8.2, we note that the uniform distribution over C'_m and the uniform distribution over C_m are nearly identical (the two distributions have total variation distance $O(n/2^n)$). Lemma 8.3 thus has the following analogue for C_m which clearly implies Theorem 8.2:

LEMMA 8.4. *For all polynomials Q, all p.p.t. learning algorithms A, and all sufficiently large n,*

$$\Pr_{c_{y,\tilde{s}} \in C_m}[A^{c_{y,\tilde{s}}} \textit{ outputs a hypothesis } h \equiv c_{y,\tilde{s}}] < \frac{1}{Q(n)}.$$

The proof of Lemma 8.3 proceeds as follows: we say that a learning algorithm A *hits* y if at some point during its execution A makes a seed query $(1, y, i, j)$, and we say that A *misses* y if A does not hit y. We have that

$$\begin{aligned}\Pr_{c_{y,\tilde{s}} \in C'_m}[A^{c_{y,\tilde{s}}} \text{ outputs } h \equiv c_{y,\tilde{s}}] &= \Pr[A^{c_{y,\tilde{s}}} \text{ outputs } h \equiv c_{y,\tilde{s}} \ \ \& \ \ A^{c_{y,\tilde{s}}} \text{ hits } y] + \\ &\quad \Pr[A^{c_{y,\tilde{s}}} \text{ outputs } h \equiv c_{y,\tilde{s}} \ \ \& \ \ A^{c_{y,\tilde{s}}} \text{ misses } y] \\ &\le \Pr[A^{c_{y,\tilde{s}}} \text{ hits } y] + \\ &\quad \Pr[A^{c_{y,\tilde{s}}} \text{ outputs } h \equiv c_{y,\tilde{s}} \ \mid \ A^{c_{y,\tilde{s}}} \text{ misses } y].\end{aligned}$$

Lemma 8.3 thus follows from the following two lemmas:

LEMMA 8.5. *For all polynomials Q, all p.p.t. learning algorithms A, and all sufficiently large n,*

$$\Pr_{c_{y,\tilde{s}} \in C'_m}[A^{c_{y,\tilde{s}}} \textit{ hits } y] < \frac{1}{Q(n)}.$$

LEMMA 8.6. *For all polynomials Q, all p.p.t. learning algorithms A, and all sufficiently large n,*

$$\Pr_{c_{y,\tilde{s}} \in C'_m}[A^{c_{y,\tilde{s}}} \textit{ outputs } h \equiv c_{y,\tilde{s}} \ \mid \ A^{c_{y,\tilde{s}}} \textit{ misses } y] < \frac{1}{Q(n)}.$$

8.3.1. Proof of Lemma 8.5.. The idea of the proof is as follows: before hitting y for the first time algorithm A gets 0 as the answer to each seed query, so A might as well be querying a modified oracle which answers 0 to *every* seed query. We show that no p.p.t. algorithm which has access to such an oracle can output y with inverse polynomial success probability (intuitively this is because such an oracle consists entirely of pseudorandom functions and hence can provide no information to any p.p.t. algorithm), and thus A's probability of hitting y must be less than $1/\text{poly}(n)$ as well.

More formally, let A be any p.p.t. learning algorithm. Without loss of generality we may suppose that A always makes exactly $q(n)$ seed queries during its execution for some polynomial q. Let $X^1, \dots, X^{q(n)}$ be the sequence of strings in $\{0,1\}^n$ on which $A^{c_{y,\tilde{s}}}$ makes its seed queries, i.e. $A^{c_{y,\tilde{s}}}$ uses $(1, X^t, i_t, j_t)$ as its t-th seed query. Each X^t is a random variable over the probability space defined by the uniform choice of $c_{y,\tilde{s}} \in C'_m$ and any internal randomness of algorithm A.

For each $c_{y,\tilde{s}} \in C'_m$ let $\tilde{c}_{y,\tilde{s}} : \{0,1\}^m \to \{0,1\}$ be a modified version of $c_{y,\tilde{s}}$ which answers 0 to all seed queries, i.e. $\tilde{c}_{y,\tilde{s}}(b,x,i,j)$ is $c_{y,\tilde{s}}(b,x,i,j)$ if $b = 0$ and is 0 if $b = 1$. Consider the following algorithm B which takes access to an oracle for $\tilde{c}_{y,\tilde{s}}$ and outputs an n-bit string. B executes algorithm $A^{\tilde{c}_{y,\tilde{s}}}$ (note that the oracle used is $\tilde{c}_{y,\tilde{s}}$ rather than $c_{y,\tilde{s}}$), then chooses a uniform random value $1 \le t \le q(n)$ and outputs $\tilde{X}^t$, the string on which $A^{\tilde{c}_{y,\tilde{s}}}$ made its t-th seed query. Like the X^ts, each $\tilde{X}^t$ is a random variable over the probability space defined by a uniform choice of $c_{y,\tilde{s}} \in C'_m$ and any internal randomness of A.

The following two lemmas together imply Lemma 8.5:

LEMMA 8.7. $2q(n)^2 \cdot \Pr_{c_{y,\tilde{s}} \in C'_m}[B^{\tilde{c}_{y,\tilde{s}}} \textit{ outputs } y] \ge \Pr_{c_{y,\tilde{s}} \in C'_m}[A^{c_{y,\tilde{s}}} \textit{ hits } y]$.

LEMMA 8.8. *For all polynomials Q and all sufficiently large n, we have*

$$\left| \Pr_{c_{y,\tilde{s}} \in C'_m}[B^{\tilde{c}_{y,\tilde{s}}} \textit{ outputs } y] - \frac{1}{2^n} \right| < \frac{1}{Q(n)}$$

Proof of Lemma 8.7. We have that

$$\sum_{t=1}^{q(n)} \Pr[X^t = y \;\;\&\;\; X^\tau \neq y \text{ for } \tau < t] \le \sum_{t=1}^{q(n)} \Pr[X^t = y \;\mid\; X^\tau \neq y \text{ for } \tau < t].$$

Since the left side of this inequality is exactly $\Pr_{c_{y,\tilde{s}} \in C'_m}[A^{c_{y,\tilde{s}}} \text{ hits } y]$, for some value $1 \le t_0 \le q(n)$ we have

$$\Pr[X^{t_0} = y \;\mid\; X^\tau \neq y \text{ for } \tau < t_0] \;\ge\; \Pr[A^{c_{y,\tilde{s}}} \text{ hits } y]/q(n). \tag{8.1}$$

Since the distribution of responses to function queries which A makes prior to its first seed query is the same regardless of whether the oracle is $c_{y,\tilde{s}}$ or $\tilde{c}_{y,\tilde{s}}$, it is clear that the random variables X^1 and $\tilde{X}^1$ are identically distributed. An inductive argument shows that for all $t \ge 1$, the conditional random variables $X^t \mid (X^\tau \neq y \text{ for } \tau < t)$ and $\tilde{X}^t \mid (\tilde{X}^\tau \neq y \text{ for } \tau < t)$ are identically distributed (in each case the conditioning ensures that the distribution of responses to seed queries which A makes prior to its t-th seed query is the same, i.e. all 0).

We consider two possible cases. If $\Pr_{c_{y,\tilde{s}} \in C'_m}[\tilde{X}^\tau \neq y \text{ for } \tau < t_0] > 1/2$, then

$$\Pr_{c_{y,\tilde{s}} \in C'_m}[B^{\tilde{c}_{y,\tilde{s}}} \text{ outputs } y] \ge \Pr[B^{\tilde{c}_{y,\tilde{s}}} \text{ chooses } t_0] \cdot \Pr[\tilde{X}^{t_0} = y \;\&\; \tilde{X}^\tau \neq y \text{ for } \tau < t_0]$$

$$
\begin{aligned}
&= \frac{\Pr[\tilde{X}^{t_0}= y \mid \tilde{X}^{\tau}\neq y \text{ for } \tau < t_0] \cdot \Pr[\tilde{X}^{\tau}\neq y \text{ for } \tau < t_0]}{q(n)} \\
&> \Pr[\tilde{X}^{t_0}= y \mid \tilde{X}^{\tau}\neq y \text{ for } \tau < t_0]/2q(n) \\
&= \Pr[X^{t_0}= y \mid X^{\tau}\neq y \text{ for } \tau < t_0]/2q(n) \\
&\geq \Pr[A^{c_{y,\bar{s}}} \text{ hits } y]/2q(n)^2. && \text{by (2)}
\end{aligned}
$$

Otherwise if $\Pr_{c_{y,\bar{s}}\in C'_m}[\tilde{X}^{\tau} \neq y \text{ for } \ \tau < t_0] \leq 1/2$, then $\sum_{t=1}^{t_0-1} \Pr[\tilde{X}^t = y] \geq 1/2$ and hence $\Pr_{c_{y,\bar{s}}\in C'_m}[B^{\tilde{c}_{y,\bar{s}}} \text{ outputs } y]$ is at least

$$\sum_{t=1}^{t_0-1} \Pr[B^{\tilde{c}_{y,\bar{s}}} \text{ chooses } t] \cdot \Pr[\tilde{X}^t = y] \geq \frac{1}{2q(n)} \geq \frac{\Pr[A^{c_{y,\bar{s}}} \text{ hits } y]}{2q(n)^2}. \qquad \square$$

Proof of Lemma 8.8. For $z, \zeta \in \{0,1\}^n$ let

$$p_\zeta^z = \Pr_{c_{y,\bar{s}}\in C'_m}[B^{\tilde{c}_{y,\bar{s}}} \text{ outputs } z \ \mid \ y = \zeta].$$

For $\ell \in \{1,\ldots,n\}$ let $\zeta||\ell$ denote ζ with the ℓ-th bit flipped. Similarly, for $S \subseteq \{1,\ldots,n\}$ let $\zeta||S$ denote ζ with bits flipped in all positions corresponding to S.

Fix $z, \zeta \in \{0,1\}^n$ and $\ell \in \{1,\ldots,n\}$ and consider the following algorithm $D_{z,\zeta,\ell}$ which takes access to an oracle $f : \{0,1\}^n \to \{0,1\}^n$ and outputs a single bit: For all $i \neq \ell$ algorithm $D_{z,\zeta,\ell}$ first chooses a random n-bit string s^i. $D_{z,\zeta,\ell}$ then runs algorithm B, simulating the oracle for B as follows:

- queries $(0,x,\ell,j)$ are answered with the bit $f(x)_j$
- for $i \neq \ell$ queries $(0,x,i,j)$ are answered with the bit $h_{s^i}^{\zeta_i}(x)_j$
- all queries $(1,x,i,j)$ are answered with the bit 0.

Finally algorithm $D_{z,\zeta,\ell}$ outputs 1 if B's output is z and outputs 0 otherwise.

It is easy to verify that for all z, ζ, ℓ we have

$$p_\zeta^z = \Pr_{s\in\{0,1\}^n}[D_{z,\zeta,\ell}^{h_s^{\zeta_\ell}} \text{ outputs } 1]$$

and

$$p_{\zeta||\ell}^z = \Pr_{s\in\{0,1\}^n}[D_{z,\zeta,\ell}^{h_s^{1-\zeta_\ell}} \text{ outputs } 1].$$

From the definition of pseudorandomness and the triangle inequality it follows that $|p_\zeta^z - p_{\zeta||\ell}^z| < \frac{1}{nQ(n)}$. Making $|S| \leq n$ applications of this inequality and using the triangle inequality, we find that

$$|p_\zeta^z - p_{\zeta||S}^z| < \frac{1}{Q(n)}.$$

We thus have that $|p_\zeta^z - p_z^z| < \frac{1}{Q(n)}$ for all $z, \zeta \in \{0,1\}^n$. Since $\sum_{z\in\{0,1\}^n} p_\zeta^z = 1$, we have that

$$
\begin{aligned}
\left| \Pr_{c_{y,\bar{s}}\in C'_m}[B^{\tilde{c}_{y,\bar{s}}} \text{ outputs } y] - \frac{1}{2^n} \right| &= \left| \frac{1}{2^n}\left(\sum_{z\in\{0,1\}^n} p_z^z \right) - \frac{1}{2^n} \right| \\
&= \frac{1}{2^n} \left| \sum_{z\in\{0,1\}^n} (p_z^z - p_\zeta^z) \right| \\
&< \frac{1}{Q(n)}. \qquad \square
\end{aligned}
$$

8.3.2. Proof of Lemma 8.6.. The idea here is that conditioning on the event that A misses y ensures that the only information which A has about y and $\tilde{s}$ comes from querying oracles for the pseudorandom functions $h^{y_i}_{s^i}$. Since these pseudorandom functions are indistinguishable from truly random functions, no p.p.t. algorithm can learn successfully.

Formally, let A be any p.p.t. learning algorithm. Consider the following algorithm B which takes access to an oracle $h^{y_n}_{s^n} : \{0,1\}^n \to \{0,1\}^n$ and outputs a representation of a function $g : \{0,1\}^n \to \{0,1\}^n$. Algorithm B first chooses $\hat{y} = y_1 \dots y_{n-1}$ uniformly from $\{0,1\}^{n-1}$ and chooses $n-1$ strings $s^1, \dots, s^{n-1}$ each uniformly from $\{0,1\}^n$. B then runs algorithm $A^{\tilde{c}_{y,\tilde{s}}}$ (observe that B can simulate the oracle $\tilde{c}_{y,\tilde{s}}$ since it has access to an oracle for $h^{y_n}_{s^n}$ and knows y_i, s^i for $i \neq n$) which generates some hypothesis h. Finally B outputs the function $g : \{0,1\}^n \to \{0,1\}^n$ defined by $g(x) \stackrel{\text{def}}{=} h(0,x,n,1)h(0,x,n,2)\dots h(0,x,n,n)$.

The following two lemmas together imply Lemma 8.6:

LEMMA 8.9. *For all sufficiently large n,*

$$\Pr_{y_n \in \{0,1\}, s^n \in \{0,1\}^n}[B^{h^{y_n}_{s^n}} \textit{ outputs } g \equiv h^{y_n}_{s^n}] > \Pr_{c_{y,\tilde{s}} \in C'_m}[A^{c_{y,\tilde{s}}} \textit{ outputs } h \equiv c_{y,\tilde{s}} \mid A^{c_{y,\tilde{s}}} \textit{ misses } y]/2.$$

LEMMA 8.10. *For all polynomials Q and all sufficiently large n, we have*

$$\Pr_{y_n \in \{0,1\}, s^n \in \{0,1\}^n}[B^{h^{y_n}_{s^n}} \textit{ outputs } g \equiv h^{y_n}_{s^n}] < \frac{1}{Q(n)}.$$

Proof of Lemma 8.9. It is easy to see that if $A^{\tilde{c}_{y,\tilde{s}}}$ outputs a hypothesis which is equivalent to $c_{y,\tilde{s}}$, then g will be equivalent to $h^{y_n}_{s^n}$. For sufficiently large n we thus have that $\Pr_{y_n \in \{0,1\}, s^n \in \{0,1\}^n}[B^{h^{y_n}_{s^n}} \text{ outputs } g \equiv h^{y_n}_{s^n}]$ is at least

$$\begin{aligned}
\Pr_{c_{y,\tilde{s}} \in C'_m}[A^{\tilde{c}_{y,\tilde{s}}} \text{ outputs } h \equiv c_{y,\tilde{s}}] &\geq \Pr[A^{\tilde{c}_{y,\tilde{s}}} \text{ outputs } h \equiv c_{y,\tilde{s}} \;\;\&\;\; A^{\tilde{c}_{y,\tilde{s}}} \text{ misses } y] \\
&= \Pr[A^{\tilde{c}_{y,\tilde{s}}} \text{ outputs } h \equiv c_{y,\tilde{s}} \mid A^{\tilde{c}_{y,\tilde{s}}} \text{ misses } y] \cdot \\
&\quad \Pr[A^{\tilde{c}_{y,\tilde{s}}} \text{ misses } y] \\
&> \Pr[A^{\tilde{c}_{y,\tilde{s}}} \text{ outputs } h \equiv c_{y,\tilde{s}} \mid A^{\tilde{c}_{y,\tilde{s}}} \text{ misses } y]/2
\end{aligned}$$

where the last inequality follows from Lemma 8.5.

Let $TRANS(A^{c_{y,\tilde{s}}})$ ($TRANS(A^{\tilde{c}_{y,\tilde{s}}})$ respectively) denote a complete transcript of algorithm A's execution on oracle $c_{y,\tilde{s}}$ ($\tilde{c}_{y,\tilde{s}}$ respectively). $TRANS(A^{c_{y,\tilde{s}}})$ and $TRANS(A^{\tilde{c}_{y,\tilde{s}}})$ are each random variables over the probability space defined by a uniform choice of $c_{y,\tilde{s}} \in C'_m$ and any internal randomness of algorithm A. An easy induction shows that the two conditional random variables $TRANS(A^{\tilde{c}_{y,\tilde{s}}}) \mid (A^{\tilde{c}_{y,\tilde{s}}}$ misses $y)$ and $TRANS(A^{c_{y,\tilde{s}}}) \mid (A^{c_{y,\tilde{s}}}$ misses $y)$ are identically distributed. This implies that

$$\Pr_{c_{y,\tilde{s}} \in C'_m}[A^{\tilde{c}_{y,\tilde{s}}} \text{ outputs } h \equiv c_{y,\tilde{s}} | A^{\tilde{c}_{y,\tilde{s}}} \text{ misses } y] = \Pr_{c_{y,\tilde{s}} \in C'_m}[A^{c_{y,\tilde{s}}} \text{ outputs } h \equiv c_{y,\tilde{s}} | A^{c_{y,\tilde{s}}} \text{ misses } y]$$

which combined with the inequality above proves the lemma. □

Proof of Lemma 8.10. The following fact, which follows easily from the pseudorandomness of $\{h^0\}$ and $\{h^1\}$, states that $\{h^b_s\}_{b \in \{0,1\}, s \in \{0,1\}^n}$ is a pseudorandom function family:

FACT 4. *For all polynomials Q, p.p.t. oracle algorithms A, and sufficiently large n, we have*

$$\left| \Pr_{b \in \{0,1\}, s \in \{0,1\}^n} [A^{h_s^b} \text{ outputs } 1] - \Pr_{F \in \mathcal{F}_n} [A^F \text{ outputs } 1] \right| < \frac{1}{Q(n)}.$$

Intuitively the pseudorandomness of $\{h_s^b\}$ should make it hard for $B^{h_{s^n}^{y_n}}$ to output $h_{s^n}^{y_n}$ since clearly no p.p.t. algorithm, given oracle access to a truly random function F, could output a function equivalent to F. Formally, we consider an algorithm D which takes oracle access to a function $f : \{0,1\}^n \rightarrow \{0,1\}^n$ and outputs a single bit. D runs B^f to obtain a function g and then selects a string $z \in \{0,1\}^n$ which was not used as an oracle query in the computation of B^f. D calls the oracle to obtain $f(z)$, evaluates g to obtain $g(z)$, and ouputs 1 if the two values are equal and 0 otherwise.

Clearly $\Pr[D^f \text{ outputs } 1] \geq \Pr[B^f \text{ outputs } g \equiv f]$. Since $\Pr_{F \in \mathcal{F}_n}[D^F \text{ outputs } 1] = 1/2^n$, using Fact 4 we find that

$$\left| \Pr_{y_n \in \{0,1\}, s^n \in \{0,1\}^n} [D^{h_{s^n}^{y_n}} \text{ outputs } 1] - \frac{1}{2^n} \right| < \frac{1}{2Q(n)}$$

and hence

$$\Pr_{y_n \in \{0,1\}, s^n \in \{0,1\}^n} [B^{h_{s^n}^{y_n}} \text{ outputs } g \equiv h_{s^n}^{y_n}] < \frac{1}{Q(n)}. \qquad \square$$

9. Breaking Classical Cryptography in a Quantum Setting. Our constructions highlight some interesting issues concerning the relation between quantum oracle computation and classical cryptography. It is clear that a quantum algorithm, given access to a quantum black-box oracle for an unknown function, can efficiently distinguish between truly random functions and pseudorandom functions drawn from the family $\{g_s\}$ of Theorem 7.1. Our construction of $\{g_s\}$ thus shows that cryptographic constructions which are provably secure in the classical model can fail in a quantum setting. We emphasize that this failure does *not* depend on the ability of polynomial-time quantum algorithms to invert particular one-way functions such as factoring; even if no quantum algorithm can efficiently invert the one-way function used to construct $\{g_s\}$, our results show that a polynomial-time quantum algorithm can be a successful distinguisher. It would be interesting to obtain stronger constructions of pseudorandom functions which are provably secure in the quantum oracle framework.

10. Conclusion and Future Directions. While we have shown that quantum and classical learning are information-theoretically equivalent up to polynomial factors, we have not attempted to obtain the tightest possible bounds relating the two query complexities. In another direction, while we have shown the existence of concept classes which separate efficient quantum and classical learning, many questions remain about the relationship between efficient quantum and classical learnability for natural concept classes studied in learning theory. It would be interesting to develop efficient quantum learning algorithms for natural concept classes, such as the polynomial-time quantum algorithm of Bshouty and Jackson [13] for learning DNF formulae from uniform quantum examples.

11. Acknowledgements. We thank R. de Wolf for helpful comments and suggestions and A. Klivans for stimulating discussions.

REFERENCES

[1] A. Ambainis. Quantum lower bounds by quantum arguments, *in* "Proc. 32nd ACM Symp. on Theory of Computing," (2000), 636-643. quant-ph/0002066.

[2] D. Angluin. Queries and concept learning, *Machine Learning* **2** (1988), 319-342.

[3] D. Angluin and M. Kharitonov. When won't membership queries help? *J. Comp. Syst. Sci.* **50** (1995), 336-355.

[4] M. Anthony and N. Biggs. *Computational Learning Theory: an Introduction.* Cambridge Univ. Press, 1997.

[5] R. Beals, H. Buhrman, R. Cleve, M. Mosca and R. de Wolf. Quantum lower bounds by polynomials, *in* "Proc. 39th IEEE Symp. on Found. of Comp. Sci.," (1998), 352-361. quant-ph/9802049.

[6] C. Bennett, E. Bernstein, G. Brassard and U. Vazirani. Strengths and weaknesses of quantum computing, *SIAM J. Comput.* **26**(5) (1997), 1510-1523.

[7] E. Bernstein and U. Vazirani. Quantum complexity theory, *SIAM J. Comput.,* **26**(5) (1997), 1411-1473.

[8] A. Blumer, A. Ehrenfeucht, D. Haussler and M. K. Warmuth. Learnability and the Vapnik-Chervonenkis Dimension, *J. ACM* **36**(4) (1989), 929-965.

[9] M. Boyer, G. Brassard, P. Høyer, A. Tapp. Tight bounds on quantum searching, *Forschritte der Physik* **46**(4-5) (1998), 493-505.

[10] G. Brassard, P. Høyer and A. Tapp. Quantum counting, *in* "Proc. 25th Internat. Conf. on Automata, Languages and Programming" (1998) 820-831. quant-ph/9805082.

[11] G. Brassard and P. Høyer. An exact quantum polynomial-time algorithm for Simon's problem, *in* "Fifth Israeli Symp. on Theory of Comp. and Systems" (1997), 12-23.

[12] N. Bshouty, R. Cleve, R. Gavaldà, S. Kannan and C. Tamon. Oracles and queries that are sufficient for exact learning, *J. Comput. Syst. Sci.* **52**(3) (1996), 421-433.

[13] N. Bshouty and J. Jackson. Learning DNF over the uniform distribution using a quantum example oracle, *SIAM J. Comput.* **28**(3) (1999), 1136-1153.

[14] H. Buhrman, R. Cleve, R. de Wolf and C. Zalka. Reducing error probability in quantum algorithms, *in* "Proc. 40th IEEE Symp. on Found. of Computer Science," (1999), 358-368. quant-ph/9904019.

[15] H. Buhrman, R. Cleve and A. Wigderson. Quantum vs. classical communication and computation, *in* "Proc. 30th ACM Symp. on Theory of Computing," (1998), 63-68. quant-ph/9802040.

[16] R. Cleve. An introduction to quantum complexity theory, *to appear in* "Collected Papers on Quantum Computation and Quantum Information Theory," ed. by C. Macchiavello, G.M. Palma and A. Zeilinger. quant-ph/9906111.

[17] D. Deutsch and R. Jozsa. Rapid solution of problems by quantum computation, *Proc. Royal Society of London A,* **439** (1992), 553-558.

[18] R. de Wolf, personal communication, 2000.

[19] E. Farhi, J. Goldstone, S. Gutmann and M. Sipser. How many functions can be distinguished with k quantum queries?, available at `http:/www.arxiv.org/abs/quant-ph/9901012`, 1999.

[20] S. Fenner, L. Fortnow, S. Kurtz and L. Li. An oracle builder's toolkit, *in* "Proc. Eighth Structure in Complexity Theory Conference" (1993), 120-131.

[21] L. Fortnow and J. Rogers. Complexity limitations on quantum computation. *Journal of Comput. and Syst. Sci.* **59**(2) (1999), 240-252.

[22] R. Gavaldà. The complexity of learning with queries, *in* "Proc. Ninth Structure in Complexity Theory Conference" (1994), 324-337.

[23] O. Goldreich, S. Goldwasser and S. Micali. How to construct random functions, *J. ACM* **33**(4) (1986), 792-807.

[24] L. K. Grover. A fast quantum mechanical algorithm for database search, *in* "Proc. 28th Symp. on Theory of Computing" (1996), 212-219.

[25] J. Håstad, R. Impagliazzo, L. Levin and M. Luby. A pseudorandom generator from any one-way function, *SIAM J. Comp.* **28**(4) (1999), 1364-1396.

[26] T. Hegedűs. Generalized teaching dimensions and the query complexity of learning, *in* "Proc. Eighth Conf. on Comp. Learning Theory," (1995), 108-117.

[27] L. Hellerstein, K. Pillaipakkamnatt, V. Raghavan and D. Wilkins. How many queries are needed to learn? *J. ACM* **43**(5) (1996), 840-862.

[28] M. Kearns and L. Valiant. Cryptographic limitations on learning boolean formulae and finite automata, *J. ACM* **41**(1) (1994), 67-95.

[29] M. Kearns and U. Vazirani. *An Introduction to Computational Learning Theory.* MIT Press, 1994.

[30] M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudrandom functions, *SIAM J. Comp.* **17**(2) (1988), 373-386.
[31] M. Nielsen and I. Chuang. *Quantum computation and quantum information.* Cambridge University Press, 2000.
[32] J. Ortega. *Matrix Theory: a second course.* Plenum Press, 1987.
[33] J. Patarin. How to construct pseudorandom and super pseudorandom permutations from one single pseudorandom function, *in* "Adv. in Crypt. – EUROCRYPT '92" (1992), 256-266.
[34] J. Pierpzyk. How to construct pseudorandom permutations from single pseudorandom functions. *in* "Adv. in Crypt. – EUROCRYPT '90" (1990), 140-150.
[35] J. Pierpzyk and B. Sadeghiyan. A construction for super pseudorandom permutations from a single pseudorandom function. *in* "Adv. in Crypt. – EUROCRYPT '92" (1992), 267-284.
[36] J. Preskill. Lecture notes on quantum computation (1998). Available at `http://www.theory.caltech.edu/people/preskill/ph229/`
[37] R. Servedio and S. Gortler. Quantum versus classical learnability, *in* "Proc. 16th Conf. on Computational Complexity" (2001), 138-148.
[38] R. Servedio. Separating quantum and classical learning, *in* "Proc. 28th Internat. Conf. on Automata, Languages, and Programming" (2001), 1065-1080.
[39] P. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, *SIAM J. Comput.* **26**(5) (1997), 1484-1509.
[40] D. Simon. On the power of quantum computation, *SIAM J. Comput.* **26**(5) (1997), 1474-1483.
[41] L. G. Valiant. A theory of the learnable, *Comm. ACM* **27**(11) (1984), 1134-1142.
[42] J. H. Van Lint. *Introduction to Coding Theory.* Springer-Verlag, 1992.
[43] V.N. Vapnik and A.Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities, *Theory of Probability and its Applications,* **16**(2) (1971), 264-280.
[44] A. Yao. Theory and applications of trapdoor functions, *in* "Proc. 23rd FOCS" (1982), 80-91.
[45] A.C. Yao. Quantum circuit complexity, *in* "Proc. 34th Symp. on Found. of Comp. Sci." (1993), 352-361.
[46] C. Zalka. Grover's quantum searching algorithm is optimal. *Physical Review A* **60** (1999), 2746–2751.

Appendix A. Proof of Theorem 7.1.

We say that a polynomial-time deterministic algorithm $G : \{0,1\}^n \to \{0,1\}^{2n}$ is a *pseudorandom generator* if for all polynomials Q, all p.p.t. algorithms A, and all sufficiently large n,

$$\left| \Pr_{z \in \{0,1\}^n}[A(G(z)) \text{ outputs } 1] - \Pr_{z \in \{0,1\}^{2n}}[A(z) \text{ outputs } 1] \right| < \frac{1}{Q(n)}.$$

Thus a pseudorandom generator is an efficient algorithm which converts an n-bit random string into a $2n$-bit string which "looks random" to any polynomial-time algorithm. Håstad et al. [25] have shown that pseudorandom generators exist if any one-way function exists.

For G a pseudorandom generator and $s \in \{0,1\}^n$ we write $G_0(s)$ to denote the first n bits of $G(s)$ and $G_1(s)$ to denote the last n bits of $G(s)$. For $x, s \in \{0,1\}^n$ let $f_s : \{0,1\}^n \to \{0,1\}^n$ be defined as

$$f_s(x) \stackrel{\text{def}}{=} G_{x_n}(G_{x_{n-1}}(\cdots(G_{x_2}(G_{x_1}(s)))\cdots)).$$

In [23] it is shown that $\{f_s\}$ is a pseudorandom function family. We now show that the family $\{g_s\}$ defined by $g_s(x) \stackrel{\text{def}}{=} f_s(x) \oplus f_s(x \oplus s)$ is pseudorandom.

Let $\mathcal{F}'_n$ be the following probability distribution over functions from $\{0,1\}^n$ to $\{0,1\}^n$: a function F' is drawn from $\mathcal{F}'_n$ by drawing a random function F from $\mathcal{F}_n$, drawing a random string $s \in \{0,1\}^n$, and letting F' be the function defined as $F'(x) = F(x) \oplus F(x \oplus s)$. Theorem 7.1 follows from the following two lemmas:

LEMMA A.1. *For all polynomials Q, all p.p.t. oracle algorithms M, and all sufficiently large n,*

$$\left|\Pr_{F\in\mathcal{F}_n}[M^F \text{ outputs } 1] - \Pr_{F'\in\mathcal{F}'_n}[M^{F'} \text{ outputs } 1]\right| < \frac{1}{Q(n)}.$$

Proof. Consider an execution of M with an oracle $F' \in \mathcal{F}'_n$ defined by $F'(x) = F(x)\oplus F(x\oplus s)$. Let $S = \{x^1, \ldots, x^t\} \subset \{0,1\}^n$ be the set of strings which M uses as queries to F'. We say that M *finds* s if $x^i = x^j\oplus s$ for some $x^i, x^j \in S$. If M does not find s, then the distribution of answers which M receives from F' is identical to the distribution which M would receive if it were querying a random function $F \in \mathcal{F}_n$, since in both cases each distinct query is answered with a uniformly distributed n-bit string. Thus the left side of the inequality above is at most $\Pr[M \text{ finds } s]$. A simple inductive argument given in the proof of Theorem 3.3 of [40] shows that this probability is at most $\sum_{k=1}^{t}(k/(2^n-(k-2)(k-1)/2))$. Since M is polynomial-time, t is at most $\text{poly}(n)$ and the lemma follows. □

LEMMA A.2. *For all polynomials Q, all p.p.t. oracle algorithms A, and all sufficiently large n,*

$$\left|\Pr_{F'\in\mathcal{F}'_n}[M^{F'} \text{ outputs } 1] - \Pr_{s\in\{0,1\}^n}[M^{g_s} \text{ outputs } 1]\right| < \frac{1}{Q(n)}.$$

Proof. We require the following fact which is due to Yao [44]:

FACT 5. *Let G be a pseudorandom generator, let $q(n)$ and $Q(n)$ be polynomials, and let M_* be a p.p.t. algorithm which takes as input $q(n)$ strings each of length $2n$ bits. Then for all sufficiently large n we have*

$$|p_n^G - p_n^U| < \frac{1}{Q(n)},$$

where p_n^U is the probability that M_* outputs 1 on input $q(n)$ random strings in $\{0,1\}^{2n}$ and p_n^G is the probability that M_* outputs 1 on input $q(n)$ strings each of which is obtained by applying G to a random string from $\{0,1\}^n$. We prove Lemma A.2 by contradiction; so suppose that there exists a p.p.t. oracle algorithm M and a polynomial Q such that for infinitely many values of n,

$$\left|\Pr_{F'\in\mathcal{F}'_n}[M^{F'} \text{ outputs } 1] - \Pr_{s\in\{0,1\}^n}[M^{g_s} \text{ outputs } 1]\right| \geq \frac{1}{Q(n)}. \tag{A.1}$$

We will show that there is a p.p.t. algorithm M_* which contradicts Fact 5.

As in the proof in [23] that $\{f_s\}$ is a pseudorandom function family, we use a so-called "hybrid" argument. Consider the following algorithms A_i ($i = 0, 1, \ldots, n$), each of which defines a mapping from $\{0,1\}^n$ to $\{0,1\}^n$ and hence could conceivably be used as an oracle to answer M's queries. Conceptually, each algorithm A_i contains a full binary tree of depth n in which the root (at depth 0) is labeled with a random n-bit string s; if $i > 0$ then each node at depth i is also labeled with an independently chosen random n-bit string. Each node at depth $j > i$ also has an n-bit label determined as follows: if node v has label z then the left child of v has label $G_0(z)$ and the right child of v has label $G_1(z)$. Each node in the tree has an *address* which is a binary string: the root's address is the empty string, and if a node has address $\alpha \in \{0,1\}^*$ then its left child has address $\alpha 0$ and its right child has address $\alpha 1$ (so each leaf has a different

n-bit string as its address). Let $L(x)$ denote the label of the node whose address is x. Algorithm A_i answers a query $x \in \{0,1\}^n$ with the n-bit string $L(x) \oplus L(x \oplus s)$.

(Note that algorithm A_i need not precompute any leaf labels. Instead, A_i can run in poly(n) time at each invocation by randomly choosing s once and for all the first time it is invoked and labeling the necessary portion of the tree "on the fly" at each invocation by choosing random strings for the depth-i nodes as required and computing descendents' labels as described above. A_i must store the random strings which it uses to label depth-i nodes so as to maintain consistency over successive invocations.)

For $i = 0, 1, \ldots, n$ let p_n^i denote $\Pr[M^{A_i}$ outputs 1], i.e. the probability that M outputs 1 if its oracle queries are answered by algorithm A_i. Let p_n^g denote $\Pr_{s \in \{0,1\}^n}[M^{g_s}$ outputs 1] and $p_n^{F'}$ denote $\Pr_{F' \in \mathcal{F}'_n}[M^{F'}$ outputs 1]. We have that $p_n^0 = p_n^g$ since algorithm A_0 behaves exactly like an oracle for g_s where s is a random n-bit string. We also have that $p_n^n = p_n^{F'}$ since algorithm A_n behaves exactly like an oracle for $F' \in \mathcal{F}'_n$. Inequality (3) thus implies that $|p_n^0 - p_n^n| \geq 1/Q(n)$ for infinitely many values of n.

Now we describe the algorithm M_* which distinguishes between sets of strings. Let $q(n)$ be a polynomial which bounds the running time of M on inputs of length n (so M makes at most $q(n)$ oracle queries given access to an oracle from $\{0,1\}^n$ to $\{0,1\}^n$). The algorithm M_* takes as input a set U_n of $2q(n)$ strings of length $2n$. M_* works by first selecting a uniform random value $0 \leq i \leq n-1$ and a uniform random string $s \in \{0,1\}^n$. M_* then runs algorithm M, answering M's oracle queries as follows (there are two cases depending on whether or not the prefix $s_1 \ldots s_i$ is 0^i):

- **Case 1:** $\boldsymbol{s_1 \ldots s_i \neq 0^i}$**.** Let $x = x_1 \ldots x_n$ be the query string. If no earlier query string had prefix $x_1 \ldots x_i$ or $(x_1 \oplus s_1) \ldots (x_i \oplus s_i)$, then M_* takes the next two $2n$-bit strings from U_n; call these strings $u^1 = u_0^1 u_1^1$ and $u^2 = u_0^2 u_1^2$ where $|u_j^i| = n$. M_* stores the four pairs

$$(x_1 \ldots x_i 0, u_0^1),\ (x_1 \ldots x_i 1, u_1^1),\ ((x_1 \oplus s_1) \ldots (x_i \oplus s_i) 0, u_0^2),\ \text{and}$$
$$((x_1 \oplus s_1) \ldots (x_i \oplus s_i) 1, u_1^2)$$

and answers with the string

$$G_{x_n}(G_{x_{n-1}}(\ldots G_{x_{i+2}}(u^1_{x_{i+1}}) \ldots)) \oplus$$
$$G_{x_n \oplus s_n}(G_{x_{n-1} \oplus s_{n-1}}(\ldots G_{x_{i+2} \oplus s_{i+2}}(u^2_{x_{i+1} \oplus s_{i+1}}) \ldots)). \qquad (*)$$

Otherwise, if an earlier query string had prefix $x_1 \ldots x_i$ or $(x_1 \oplus s_1) \ldots (x_i \oplus s_i)$, then instead M_* retrieves the two previously stored pairs

$$(x_1 \ldots x_i x_{i+1}, u^1_{x_{i+1}}) \text{ and } ((x_1 \oplus s_1) \ldots (x_i \oplus s_i)(x_{i+1} \oplus s_{i+1}), u^2_{x_{i+1} \oplus s_{i+1}})$$

and answers with $(*)$ as above.
- **Case 2:** $\boldsymbol{s_1 \ldots s_i = 0^i}$**.** Let $x = x_1 \ldots x_n$ be the query string. If no earlier query string had prefix $x_1 \ldots x_i$, then M_* takes the next $2n$-bit string from U_n; call this string $u = u_0 u_1$ where $|u_0| = |u_1| = n$. M_* stores the two pairs

$$(x_1 \ldots x_i 0, u_0),\ (x_1 \ldots x_i 1, u_1)$$

and answers with

$$G_{x_n}(G_{x_{n-1}}(\ldots G_{x_{i+2}}(u_{x_{i+1}}) \ldots)) \oplus$$
$$G_{x_n \oplus s_n}(G_{x_{n-1} \oplus s_{n-1}}(\ldots G_{x_{i+2} \oplus s_{i+2}}(u_{x_{i+1} \oplus s_{i+1}}) \ldots)). \qquad (**)$$

Otherwise, if an earlier query string had prefix $x_1 \dots x_i$, then M_* retrieves the two pairs

$$(x_1 \dots x_i x_{i+1}, u_{x_{i+1}}) \text{ and } (x_1 \dots x_i(x_{i+1} \oplus s_{i+1}), u_{x_{i+1} \oplus s_{i+1}})$$

(these two pairs are the same if $s_{i+1} = 0$) and answers with $(**)$ as above.

The crucial properties of algorithm M_*, which are straightforwardly verified, are the following: If each string in U_n is generated by applying G to a random n-bit string (scenario 1), then M_* simulates a computation of M with oracle A_i. On the other hand, if each string in U_n is chosen uniformly from $\{0,1\}^{2n}$ (scenario 2), then M_* simulates a computation of M with oracle A_{i+1}.

It is easy to see now that in scenario 1 we have $\Pr[M_* \text{ outputs } 1] = \sum_{i=0}^{n-1} p_n^i/n$ while in scenario 2 we have $\Pr[M_* \text{ outputs } 1] = \sum_{i=1}^{n} p_n^{i+1}/n$. These two probabilities differ by $(1/n) \cdot |p_n^0 - p_n^n|$, which is at least $1/nQ(n)$ for infinitely many values of n. Now by Fact 5 the existence of M_* contradicts the fact that G is a pseudorandom generator, and the lemma is proved. $\square$