# LEARNING DNF OVER THE UNIFORM DISTRIBUTION USING A QUANTUM EXAMPLE ORACLE

NADER H. BSHOUTY* AND JEFFREY C. JACKSON†

**Abstract.** We generalize the notion of PAC learning from an example oracle to a notion of efficient learning on a quantum computer using a quantum example oracle. This quantum example oracle is a natural extension of the traditional PAC example oracle, and it immediately follows that all PAC-learnable function classes are learnable in the quantum model. Furthermore, we obtain positive quantum learning results for classes that are not known to be PAC learnable. Specifically, we show that DNF is efficiently learnable with respect to the uniform distribution by a quantum algorithm using a quantum example oracle. While it was already known that DNF is uniform-learnable using a membership oracle, we prove that a quantum example oracle with respect to uniform is less powerful than a membership oracle.

**Key words.** quantum example oracle, quantum computing, disjunctive normal form (DNF), machine learning, Fourier transform

**AMS subject classifications.** 68Q20, 68Q05

* Dept. of Computer Science, University of Calgary, 2500 University Drive NW, Calgary, Alberta T2N 1N4 CANADA (bshouty@cpsc.ucalgary.ca). Research was supported by NSERC.

†Dept. of Mathematics and Computer Science, Duquesne University, 600 Forbes Ave, Pittsburgh, PA 15282-1754 (jackson@mathcs.duq.edu). Sponsored by the National Science Foundation under Grant No. CCR-9119319 while at Carnegie Mellon University.

**Contact Info:** Jeffrey Jackson, Math and Computer Science Dept, Duquesne University, Pittsburgh, PA 15282. `jackson@mathcs.duq.edu` Voice: 412/396-4851. Fax: 412/396-5197.

Nader Bshouty, University of Calgary - Dept. of Computer Science, 2500 University Drive NW, Calgary, Alberta T2N 1N4, CANADA. `bshouty@cpsc.ucalgary.ca` Phone: +1 403 220-7682. FAX: +1 403 284-4707.

**1. Introduction.** Recently, there has been significant interest in the question of to what extent quantum physical effects can be used to solve problems that appear to be computationally difficult using traditional methods [16, 8, 7, 6, 33, 31, 30]. In this paper, we apply quantum methods to questions in computational learning theory. In particular, we focus on the problem of learning—from examples alone—the class DNF of polynomial-size Disjunctive Normal Form expressions.

The DNF learning problem has a long history. Valiant [32] introduced the problem and gave efficient algorithms for learning certain subclasses of DNF. Since then, learning algorithms have been developed for a number of other subclasses of DNF [25, 4, 2, 21, 3, 1, 11, 27, 13, 10] and recently for the unrestricted class of DNF expressions [22], but almost all of these results—and in particular the results for the unrestricted class—use membership queries (the learner is told the output value of the target function on learner-specified inputs). While Angluin and Kharitonov [5] have shown that if DNF is PAC learnable in a distribution-independent sense (definitions are given in the next section) with membership queries then it is learnable with respect to polynomial-time computable distributions without these queries, the question of to what extent membership queries are necessary for distribution-dependent DNF learning is still open. In particular, Jackson's Harmonic Sieve [22] learns DNF with respect to uniform using membership queries; can DNF be learned with respect to uniform from a weaker form of oracle?

We show that DNF is efficiently learnable with respect to the uniform distribution by a quantum algorithm that receives its information about the target function from a *quantum example oracle*. This oracle generalizes the traditional PAC example oracle in a natural way. Specifically, the quantum oracle $QEX(f, D)$ is a traditional PAC example oracle $EX(f, D)$ except that $QEX(f, D)$ produces the example $\langle x, f(x)\rangle$ with amplitude $\sqrt{D(x)}$ rather than with probability $D(x)$. We also show that, with respect to the uniform distribution, a quantum example oracle can be simulated by a membership oracle but not vice versa.

To obtain our quantum DNF learning algorithm, we modify the Harmonic Sieve algorithm (`HS`) for learning DNF with respect to uniform using membership queries [22]. In fact, `HS` properly learns the larger class $\widehat{PT}_1$ of functions expressible as a threshold of a polynomial number of parity functions, and our algorithm properly learns this class as well. The Harmonic Sieve uses membership queries to locate parity functions that correlate well with the target function with respect to various near-uniform distributions. The heart of our result is showing that these parities can be located efficiently by a quantum algorithm using only a quantum example oracle.

Our primary result, then, is to show that DNF is quantum-learnable using an oracle that is strictly weaker than a membership oracle. Our algorithm also possesses a somewhat better asymptotic bound on running time than the Harmonic Sieve. We consider the potential significance of these results in more detail in the concluding section.

**2. Definitions and Notation.**

**2.1. Functions and Function Classes.** We will be interested in the learnability of sets (*classes*) of Boolean functions over $\{0, 1\}^n$ for fixed positive values of $n$. It will be convenient to use different definitions for "Boolean" in different contexts within this paper. In particular, at times we will think of a Boolean function as mapping to $\{0, 1\}$ and at other times to $\{-1, +1\}$; the choice will either be indicated explicitly or clear from context. We call $\{0, 1\}^n$ the *instance space* of a Boolean function $f$, an element $x$ in the instance space an *instance*, and the pair $\langle x, f(x)\rangle$ an *example* of $f$. We denote by $x_i$ the $i$th bit of instance $x$.

Intuitively, a learning algorithm should be allowed to run in time polynomial in the complexity of the function $f$ to be learned; we will use the *size* of a function as a measure of its complexity. The size measure will depend on the function class to be learned. In particular, each function class $\mathcal{F}$ that we study implicitly defines a natural class $\mathcal{R}_{\mathcal{F}}$ of representations of the functions in $\mathcal{F}$. We define the *size of a function* $f \in \mathcal{F}$ as the minimum, over all $r \in \mathcal{R}_{\mathcal{F}}$ such that $r$ represents $f$, of the size of $r$, and we define below the size measure for each representation class of interest.

A DNF expression is a disjunction of *terms*, where each term is a conjunction of *literals* and a literal is either a variable or its negation. The *size of a DNF expression* $r$ is the number of terms in $r$. The *DNF function class* is the set of all functions that can be represented as a DNF expression of size polynomial in $n$.

Following Bruck [12], we use $\widehat{PT}_1$ to denote the class of functions on $\{0,1\}^n$ expressible as a depth-2 circuit with a majority gate at the root and polynomially-many parity gates at the leaves. All gates have unbounded fanin and fanout one. The *size of a* $\widehat{PT}_1$ *circuit* $r$ is the number of parity gates in $r$.

**2.2. Quantum Turing Machines.** We now review the model of quantum computation defined by Bernstein and Vazarani [6]. First we define how the specification (program) of a *quantum Turing machine* (QTM) is written down. Then we describe how a QTM operates.

The specification of a QTM is almost exactly the same as the specification of a probabilistic TM (PTM). Recall that the transition table of a PTM specifies, for each state and input symbol, a set of moves—a move is a next state, new tape symbol, and head movement direction—along with associated probabilities that each move will be chosen. Of course, these probabilities must be non-negative and sum to one. In a QTM specification, the transition probabilities between PTM configurations are replaced with complex-valued numbers (*amplitudes*) that satisfy a certain *well-formedness* property. Loosely speaking, if the sum of the squares of the amplitudes for the transitions corresponding to each state/symbol pair is one, then the QTM is satisfies the well-formedness property. A somewhat peculiar aspect of amplitudes is that they may have negative real components, unlike the probabilities of the PTM model. Formally, we define well-formedness as follows. For a QTM $M$, let $R_M$ be the (infinite-dimensional) matrix where each row and each column is labeled with a distinct machine configuration ($c_r$ and $c_c$, respectively) and each entry in $R_M$ is the amplitude assigned by $M$ to the transition from configuration $c_c$ to $c_r$. Then $M$ satisfies the well-formedness property if $R_M$ is unitary ($R_M^{\dagger} R_M = R_M R_M^{\dagger} = I$, where $R_M^{\dagger}$ is the transpose conjugate of $R_M$). A QTM specification also contains a set of states (including all of the final states) in which an *Obs* operation is performed; we define this operation below.

To describe the operation of a QTM, we use the notion of a *superposition* of configurations. For example, consider a probabilistic Turing machine $M'$ that at step $i$ flips a fair coin and chooses to transition to one of two configurations $c_1$ and $c_2$. While we would generally think of $M'$ as being in exactly one of these configurations at step $i+1$, we can equivalently think of $M'$ as being in *both* states, each with probability $1/2$. Continuing in this fashion, for each step until $M'$ terminates we can think of $M'$ as being in a superposition of states, each state with an associated probability. After $M'$ takes its final step, each of its final states will have some associated probability (we assume without loss of generality that all computation paths in $M'$ have the same length). If $M'$ now "chooses" to be in one of these final states $\sigma_f$ randomly according to the induced probability distribution on final states, then the probability of being in $\sigma_f$ is exactly the same in this model as it is in the traditional PTM model.

In summary, we can view a PTM $M'$ as being in a superposition of configurations at each step, where a superposition is represented by a vector of probabilities, one for each possible configuration of $M'$. Likewise, we view a QTM $M$ as being in a superposition of configurations at each step, but now the superposition vector contains an amplitude for each possible configuration of $M$. The initial superposition vector in both cases is the all-zero vector except for a single 1 in the position corresponding to the initial configuration of the machine. Note that each step of a PTM $M'$ can be accomplished by multiplying the current superposition vector by a matrix $R_{M'}$ which is defined analogously with $R_M$ above. In the same way, each step of a QTM $M$ is accomplished by multiplying its current superposition vector by $R_M$. The difference between the machines comes at the point(s) where $M$ "chooses" to be in a single configuration rather than in a superposition of configurations. $M$ does this (conceptually) by transitioning to a superposition of configurations all of which are in one of the *Obs* states mentioned above. The superposition vector is then changed so that a single configuration has amplitude 1 and all others are 0. This is exactly analogous to the PTM $M'$ choosing its final state, except that the probability of choosing each configuration $c_i$ is now the *square* of the magnitude of the amplitude associated with $c_i$ in $M$'s current superposition vector. (We formalize the definition of *Obs* below.)

The notation

$$\sum_x a_x|x\rangle$$

denotes a superposition of configurations $x$ each having amplitude $a_x$. While in general this sum is over all possible configurations of the QTM, when we use this notation it will be the case (unless otherwise noted) that all of the configurations having nonzero amplitude are in the same state and have the tape head at the same position. In this case, the configurations $x$ are only distinguished by their tape contents, so we will treat $x$ as if it is merely the tape content and ignore the other configuration parameters. We will also assume that all tapes contain the same number of non-blank characters unless otherwise noted.

Given this notation, we now more formally define the *Obs* operation. Intuitively, an *Obs* will collapse a superposition $S$ to one of two possible superpositions $S_0$ or $S_1$, with the choice of superposition based on a probability that is a function of the value of the first bit of the tape (we are implicitly assuming that when an *Obs* is performed the configurations in a superposition differ only in terms of what is on the respective tapes, which will be sufficient for our purposes). Specifically, let $b \in \{0, 1\}$. Then

$$Obs\left(\sum_{bx} a_{bx}|bx\rangle\right) =$$

$$\begin{cases} \sum_x \frac{a_{0x}}{\sum_y |a_{0y}|^2}|0x\rangle & \text{with probability } \sum_x |a_{0x}|^2 \\ \sum_x \frac{a_{1x}}{\sum_y |a_{1y}|^2}|1x\rangle & \text{with probability } \sum_x |a_{1x}|^2. \end{cases}$$

Note that by permuting bits of the tape and performing successive *Obs* operations we can simulate the informal definition of *Obs* given earlier. We say that a language $L$

is in $BQP$ if there exists a QTM $M$ such that, at the end of a number of steps by $M$ polynomial in the length of the input to $M$, an *Obs* fixes the first tape cell to 1 with probability at least 2/3 if the input is in $L$ and fixes it to 0 with probability at least 2/3 otherwise. We will also sometimes think of an *Obs* as simply computing the probability that the first cell will be fixed to 1.

Finally, we will at times want to introduce deterministic transitions into our QTM's while preserving the well-formedness property of the transition matrices. It can be shown [15] that as long as a deterministic computation is *reversible* then the computation can be carried out on a QTM. Informally, a computation is reversible if given the result of the computation it is possible to determine the input to the computation (see [6] for a formal definition and discussion of reversibility in the context of quantum computation).

**2.3. Standard Learning Models.** We begin by defining the well-known PAC learning model and then generalize this to a quantum model of learning. First, we define several supporting concepts. Given a function $f$ and probability distribution $D$ on the instance space of $f$, we say that the Boolean function $h$ having as its domain the instance space of $f$ is an *$\epsilon$-approximator for $f$ with respect to $D$* if $\Pr_D[h = f] \geq 1 - \epsilon$. An *example oracle for $f$ with respect to $D$* ($EX(f, D)$) is an oracle that on request draws an instance $x$ at random according to probability distribution $D$ and returns the example $\langle x, f(x)\rangle$. A *membership oracle for $f$* ($MEM(f)$) is an oracle that given any instance $x$ returns the value $f(x)$. Let $\mathcal{D}_n$ denote a nonempty set of probability distributions on $\{0,1\}^n$. Any set $\mathcal{D} = \cup_n \mathcal{D}_n$ is called a *distribution class.* We let $\mathcal{U}_n$ represent the uniform distribution on $\{0,1\}^n$ and call $\mathcal{U} = \cup_n \mathcal{U}_n$ simply the *uniform distribution.*

Now we formally define the Probably Approximately Correct (PAC) model of learnability [32]. Let $\epsilon$ and $\delta$ be positive values (called the *accuracy* and *confidence* of the learning procedure, respectively). Then we say that the function class $\mathcal{F}$ is *(strongly) PAC learnable* if there is an algorithm $\mathcal{A}$ such that for any $\epsilon$ and $\delta$, any $f \in \mathcal{F}$ (the *target function*), and any distribution $D$ on the instance space of $f$ (the *target distribution*), with probability at least $1 - \delta$ algorithm $\mathcal{A}(EX(f, D), \epsilon, \delta)$ produces an $\epsilon$-approximation for $f$ with respect to $D$ in time polynomial in $n$, the size of $f$, $1/\epsilon$, and $1/\delta$. The probability that $\mathcal{A}$ succeeds is taken over the random choices made by $EX$ and $\mathcal{A}$ (if any). We generally drop the "PAC" from "PAC learnable" when the model of learning is clear from context.

We will consider several variations on the basic learning models. Let $\mathcal{M}$ be any model of learning (e.g., PAC). If $\mathcal{F}$ is $\mathcal{M}$-learnable by an algorithm $\mathcal{A}$ that requires a membership oracle then $\mathcal{F}$ is *$\mathcal{M}$-learnable using membership queries.* If $\mathcal{F}$ is $\mathcal{M}$-learnable for $\epsilon = 1/2 - 1/p(n, s)$, where $p$ is a fixed polynomial and $s$ is the size of $f$, then $\mathcal{F}$ is *weakly $\mathcal{M}$-learnable.* We say that $\mathcal{F}$ *is $\mathcal{M}$-learnable by $\mathcal{H}$* if $\mathcal{F}$ is $\mathcal{M}$-learnable by an algorithm $\mathcal{A}$ that always outputs a function $h \in \mathcal{H}$. If $\mathcal{F}$ is $\mathcal{M}$-learnable by $\mathcal{F}$ then we say that $\mathcal{F}$ is *properly* $\mathcal{M}$-learnable. Finally, note that the PAC model places no restriction on the example distribution $D$; we sometimes refer to such learning models as *distribution-independent.* If $\mathcal{F}$ is $\mathcal{M}$-learnable for all distributions $D$ in distribution class $\mathcal{D}$ then $\mathcal{F}$ is *$\mathcal{M}$-learnable with respect to $\mathcal{D}$.* Learning models which place a restriction on the distributions learned against we call *distribution-dependent* models.

**2.4. The Fourier Transform.** We will make substantial use of the discrete Fourier transform in our analysis; this approach was introduced in machine learning by Linial, Mansour, and Nisan [28]. In this section we give some basic definitions and standard theorems.

For each bit vector $a \in \{0,1\}^n$ we define the function $\chi_a : \{0,1\}^n \to \{-1,+1\}$ as

$$\chi_a(x) = (-1)^{\sum_{i=1}^n a_i x_i} = 1 - 2\left(\sum_{i=1}^n a_i x_i \bmod 2\right).$$

That is, $\chi_a(x)$ is the Boolean function that is 1 when the parity of the bits in $x$ indexed by $a$ is even and is $-1$ otherwise. With inner product defined by[1] $\langle f, g\rangle = \mathbf{E}_x[f(x)\cdot g(x)] \equiv \mathbf{E}[fg]$ and norm defined by $\|f\| = \sqrt{\mathbf{E}[f^2]}$, $\{\chi_a \mid a \in \{0,1\}^n\}$ is an orthonormal basis for the vector space of real-valued functions on the Boolean cube $\mathbf{Z}_2^n$. That is, every function $f : \{0,1\}^n \to \mathbf{R}$ can be uniquely expressed as a linear combination of parity functions:

$$f = \sum_{a\in\{0,1\}^n} \hat{f}(a)\chi_a,$$

where $\hat{f}(a) = \mathbf{E}[f\chi_a]$. We call the vector of coefficients $\hat{f}$ the *Fourier transform* of $f$. Note that for $f$ mapping to $\{-1,+1\}$, $\hat{f}(a)$ represents the correlation of $f$ and $\chi_a$ with respect to the uniform distribution. Also, let $0^n$ represent the vector of $n$ zeros. Then $\hat{f}(0^n) = \mathbf{E}[f\chi_{0^n}] = \mathbf{E}[f]$, since $\chi_{0^n}$ is the constant function $+1$.

By Parseval's identity, for every real-valued function $f$, $\mathbf{E}[f^2] = \sum_{a\in\{0,1\}^n} \hat{f}^2(a)$. For $f$ mapping to $\{-1,+1\}$ it follows that $\sum_a \hat{f}^2(a) = 1$. More generally, it can be shown that for any real-valued functions $f$ and $g$, $\mathbf{E}[fg] = \sum_a \hat{f}(a)\hat{g}(a)$.

**3. The Quantum Example Oracle.** While DNF has been shown to be learnable with respect to uniform using membership queries [22], it is desirable to have a DNF learning algorithm that can learn from examples alone. This seems to be a hard problem using conventional computing paradigms. However, other problems—such as integer factorization—which had seemed to be hard have recently been shown to have efficient quantum solutions [30]. Thus it is natural to ask the following question: is there a QTM $M$ such that, given access to a traditional PAC example oracle $EX(f,D)$ for any function $f$ in DNF, $M$ efficiently learns an $\epsilon$-approximation to $f$?

In this paper, we consider a related question that we hope may shed some light on the question posed above. Specifically, we consider the question of learning DNF from a *quantum example oracle*. This oracle, which we define below, generalizes the PAC example oracle to the quantum setting in a natural way. It should be noted that questions about the power of quantum computing relative to oracles has been investigated previously; for example, Berthiaume and Brassard [8, 7] consider the relative abilities of quantum and more traditional Turing machines to answer decision questions relative to a membership oracle.

We now define the quantum example oracle. Note that each call to the traditional PAC example oracle $EX(f,D)$ can be viewed as defining a superposition of $2^n$ configurations, each containing a distinct $\langle x, f(x)\rangle$ pair and having probability of occurrence $D(x)$. We generalize this to the quantum setting in a natural way. A *quantum example oracle for $f$ with respect to $D$* ($QEX(f,D)$) is an oracle running

[1] Expectations and probabilities here and elsewhere are with respect to the uniform distribution over the instance space unless otherwise indicated.

coherently with a QTM $M$ that changes $M$'s tape $|y\rangle$ to

$$\sum_x \sqrt{D(x)}|y, x, f(x)\rangle.$$

That is, $QEX$ defines a superposition of $2^n$ configurations much as $EX$ does, but $QEX$ assigns each configuration an amplitude $\sqrt{D(x)}$. As with the *Obs* operation, calls to $QEX$ will be invoked by transitioning into designated states of $M$, and we will assume that any valid QTM program has the property that at each step either all of the states with nonzero amplitude in a superposition call $QEX$ or none do. Note that for any $f$ and $D$, a call to $QEX(f, D)$ followed by an appropriate *Obs* operation is equivalent to a call to $EX(f, D)$. We say that $\mathcal{F}$ is *quantum learnable* if $\mathcal{F}$ is PAC learnable by a QTM $M$ using a quantum example oracle. Because every efficient TM computation can be simulated efficiently by a QTM [6] and because $EX$ can be simulated by $QEX$, we have that every PAC-learnable function class is also quantum learnable.

For both standard and quantum example oracles, we use $EX(f)$ (resp. $QEX(f)$) to represent learning with respect to the uniform distribution, that is, $EX(f, \mathcal{U}_\backslash)$ (resp. $QEX(f, \mathcal{U}_\backslash)$).

**4. Learning DNF from a Membership Oracle.** In the next section we present our primary result, that DNF is quantum learnable with respect to the uniform distribution. Our result builds on the Harmonic Sieve (`HS`) algorithm for learning DNF with respect to the uniform distribution using membership queries [22]. In this section we briefly review the Harmonic Sieve.

**4.1. Overview of `HS`.** The `HS` algorithm depends on a key fact about DNF expressions: for every DNF $f$ and distribution $D$ there is a parity $\chi_a$ that is a weak approximator to $f$ with respect to $D$ [22]. Also, for any function weakly approximable with respect to uniform by a parity function, a technique originally due to Goldreich and Levin [20] and first applied within a learning algorithm (`KM`) by Kushilevitz and Mansour [26] can be used to *find* such a parity (using membership queries). Combining these two facts gives that the `KM` algorithm weakly learns DNF with respect to uniform [9].

An obvious method to consider for turning this weak learner into a strong learner is some form of hypothesis boosting [29, 18, 17, 19]. In fact, `HS` is based on a particularly simple and efficient version of boosting discovered by Freund [18]. Each stage $i$ of Freund's boosting algorithm explicitly defines a distribution $D_i$ and calls on a weak learner to produce a weak approximator with respect to $D_i$. Distribution $D_i$ is defined in terms of the performance of the weak hypotheses produced at the preceding boosting stages and in terms of the target distribution $D$. After a polynomial number of stages, a majority vote over the weak hypotheses gives a strong approximator with respect to $D$.

So in order to strongly learn DNF with respect to a distribution $D$, all that is needed is an efficient algorithm for weakly learning DNF with respect to the set of distributions $\{D_i\}$ defined by Freund's algorithm in the process of boosting with respect to $D$. While the question of whether or not such an algorithm exists for arbitrary $D$ is an open problem, it is known that when boosting with respect to *uniform* a modified version of the `KM` algorithm can efficiently find a weakly-approximating parity for any DNF $f$ with respect to any of the distributions $D_i$ defined by Freund's algorithm [22]. When learning with respect to such a distribution $D_i$, the modified `KM` algorithm must

**Invocation:** $h \leftarrow \mathtt{HS}(n, s, MEM(f), \epsilon, \delta)$
**Input:** $n$; $s =$ size of DNF $f$; $MEM(f)$; $\epsilon > 0$; $\delta > 0$
**Output:** with probability at least $1-\delta$ (over random choices made by `HS`), `HS` returns $h$ such that $\Pr[f = h] \geq 1 - \epsilon$

$\gamma \leftarrow 1/(8s+4)$
$k \leftarrow \frac{1}{2}\gamma^{-2}\ln(4/\epsilon)$
$w_0 \leftarrow \mathtt{WDNF}(n, s, MEM(f), \mathcal{U}_n, \delta/2k)$
**for** $i \leftarrow 1, \ldots, k-1$ **do**
    $B(j; n, p) \equiv \binom{n}{j} p^j (1-p)^{n-j}$
    $\beta_r^i \equiv B(\lfloor k/2 \rfloor - r; k-i-1, 1/2+\gamma)$ if $i - k/2 < r \leq k/2$, $\beta_r^i \equiv 0$ otherwise
    $\alpha_r^i \equiv \beta_r^i / \max_{r=0,\ldots,i-1}\{\beta_r^i\}$.
    $r_i(x) \equiv |\{0 \leq j < i \mid w_j(x) = f(x)\}|$
    $\Theta \equiv c_2 \epsilon^3$
    $E_\alpha \leftarrow \mathbf{Est}(\mathbf{E}_x[\alpha^i_{r_i(x)}], EX(f), \Theta/3, \delta/2k)$
    **if** $E_\alpha \leq 2\Theta/3$ **then**
        $k \leftarrow i$
        **break do**
    **endif**
    $\tilde{D}_i(x) \equiv \alpha^i_{r_i(x)} / 2^n E_\alpha$
    $w_i \leftarrow \mathtt{WDNF}(n, s, MEM(f), \tilde{D}_i(x), \delta/2k)$
**enddo**
$h(x) \equiv MAJ(w_0(x), w_1(x), \ldots, w_{k-1}(x))$
**return** h

FIG. 4.1. *Harmonic Sieve (`HS`) algorithm for learning DNF from a membership oracle.* $\mathbf{Est}(E, EX(f), \epsilon, \delta)$ *uses random sampling from $EX(f)$ to efficiently estimate the value of $E$, producing a value within an additive factor of $\epsilon$ of the true value with probability at least $1-\delta$. $c_2$ represents a fixed constant ($1/57$ is sufficient).*

**Invocation:** $w_i \leftarrow \mathtt{WDNF}(n, s, MEM(f), cD, \delta)$
**Input:** $n$; $s =$ size of DNF $f$; $MEM(f)$; $cD$, an oracle that given $x$ returns $c \cdot D(x)$, where $c$ is a constant in $[1/2, 3/2]$ and $D$ is a probability distribution on $\{0,1\}^n$; $\delta > 0$
**Output:** with probability at least $1-\delta$ (over random choices made by `WDNF`), `WDNF` returns $h$ such that $\Pr_D[f = h] \geq 1/2 + 1/(8s+4)$

$g(x) \equiv 2^n f(x) c D(x)$
find (using membership queries and with probability at least $1-\delta$) $\chi_a$ such that $|\mathbf{E}[g\chi_a]| \geq c/(4s+2)$
$h(x) \equiv \mathrm{sign}(\mathbf{E}[g\chi_a]) \cdot \chi_a(x)$
**return** h

FIG. 4.2. `WDNF` *subroutine called by* `HS`.

be given not only a membership oracle for $f$ but also an oracle for the distribution $D_i$, that is, a function which given an instance $x$ returns the weight that $D_i$ places on $x$. Because Freund's booster explicitly defines the distribution $D_i$ at each boosting stage $i$, an oracle for each $D_i$ can be simulated and therefore the modified `KM` algorithm can be boosted by a modified version of Freund's algorithm that supplies $D_i$ to the weak learner at each stage $i$. This then gives a strong learning algorithm for DNF with respect to uniform using uses membership queries.

**4.2. Algorithmic Details.** The HS algorithm and its primary subroutine WDNF (the modified KM) are sketched in somewhat more detail in Figures 4.1 and 4.2. We will assume here and elsewhere that the number of terms $s$ in the target function's representation as a DNF is known. This assumption can be relaxed by placing a standard guess-and-double loop around the body of the HS program (see, e.g., [24] for details); this increases the running time of the algorithm by at most a factor of $\log s$. The HS algorithm runs for $O(s^2 \log(1/\epsilon))$ stages. At each stage $i$, $r_i(x)$ (line 8) represents the number of weak hypotheses $w_j$ among those hypotheses produced before stage $i$ that are "right" on $x$. For uniform target distribution, the distributions $D_i$ defined by Freund's booster are given by

$$D_i(x) = \frac{\alpha^i_{r_i(x)}}{\sum_y \alpha^i_{r_i(y)}}, \tag{4.1}$$

where $\alpha_{r_i(x)}$ is defined in Figure 4.1. Note that while $D_i$ is explicitly defined, it is not computationally feasible to compute $D_i$ exactly because of the sum over an exponential number of terms in the denominator of (4.1). However, by a Chernoff bound argument this sum, and therefore $D_i$, can be closely approximated in time polynomial in the standard parameters. The function $\tilde{D}_i$ is HS's approximation to $D_i$. Note that because of the bound on the variable $E_\alpha$ and the accuracy with which $E_\alpha$ estimates $\mathbf{E}_x[\alpha^i_{r_i(x)}]$, with probability at least $1 - \delta/2k$, $\mathbf{E}_x[\alpha^i_{r_i(x)}] = c_3 E_\alpha$ for fixed $c_3 \in [1/2, 3/2]$. With the same probability, then, $\tilde{D}_i(x) = c_3 D_i(x)$ for all $x$.

Therefore, while we show WDNF in Figure 4.1 being called with an argument $\tilde{D}_i(x)$, the corresponding parameter in Figure 4.2 is $cD$, an oracle representing the product of a constant and a probability distribution. This oracle along with the membership oracle for the target $f$ is used by WDNF to simulate an oracle $g$. We have omitted the details of line 2 of WDNF because this is the main point at which our quantum algorithm will differ from HS. Rather than using membership queries to locate the required parity $\chi_a$, the new algorithm will use a quantum example oracle. Both algorithms depend on a key fact about DNF expressions [22]: for every DNF $f$ with $s$ terms and for every distribution $D$ there exists a parity $\chi_a$ such that

$$|\mathbf{E}_D[f\chi_a]| \geq \frac{1}{2s+1}.$$

It follows from the definition of expectation that for either $h = \chi_a$ or $h = -\chi_a$

$$\mathbf{Pr}_D[f = h] \geq \frac{1}{2} + \frac{1}{4s+2}.$$

The WDNF algorithm can only guarantee to find a parity which is nearly optimally correlated with the target, which is why another factor of two is given up by the algorithm. Finally, WDNF also relies on the easily-verified fact that for $g(x) = 2^n f(x) cD(x)$, $\mathbf{E}[g\chi_a] = c\mathbf{E}_D[f\chi_a]$. In essence, by combining the target $f$ and distribution $D$ we create a function $g$ with the property that the large Fourier coefficients of $g$ correspond to parities that are weak approximators to $f$ with respect to $D$. This is why the hypothesis returned by WDNF is appropriate.

The version of WDNF modified to utilize a quantum example oracle is based on a similar idea of learning with respect to uniform in order to find a weak hypothesis with respect to a nonuniform distribution, but the implementation of the idea is quite different. We develop the modified algorithm in detail in the next section.

**5. Learning DNF from a Quantum Example Oracle.** In this section we show how to modify the Harmonic Sieve in order to uniform-learn DNF using a quantum example oracle rather than a membership oracle. First, consider the call to `WDNF` at line 16 of `HS` for a fixed $i$, and for notational convenience let $D \equiv D_i$ and $\alpha(x) \equiv \alpha^i_{r_i(x)}$. Then given the discussion concerning $\tilde{D}_i$ above and the definition of $\tilde{D}_i$ at line 15 in Figure 4.1, with high probability there is a $c_3 \in [1/2, 3/2]$ such that for all $\chi_a$,

$$\sum_x f(x)\chi_a(x)\tilde{D}_i(x) = c_3 \mathbf{E}_D[f\chi_a] = \mathbf{E}[\alpha f \chi_a]/E_\alpha.$$

Also, again with high probability, the call to **Est** at line 10 is successful in estimating $\mathbf{E}[\alpha]$ to within the desired accuracy and therefore $E_\alpha \geq 2\Theta/3$. Applying this observation to the equation above and invoking the key DNF fact cited earlier gives that with high probability, each time `WDNF` is called there exists some $\chi_a$ such that

$$|\mathbf{E}[\alpha f \chi_a]| \geq \frac{\Theta}{3(2s+1)} = \frac{c_2\epsilon^3}{3(2s+1)} \tag{5.1}$$

and this $\chi_a$ (or its inverse) is a $(1/(4s+2))$-approximator to $f$. Our goal will be to find such a $\chi_a$, or at least one that is nearly as good an approximator, using only a quantum example oracle for $f$.

Conceptually, to find such a $\chi_a$ we will repeatedly run a quantum subprogram that randomly selects one $\chi_a$ each time the subprogram runs. On any given run each $\chi_a$ is selected with probability proportional to $\mathbf{E}^2[\alpha f \chi_a]$. The technique we use to perform this random sampling from the set of $\chi_a$'s is similar to a quantum algorithm of Bernstein and Vazarani that samples the $\chi_a$'s with probability $\hat{f}^2(a) = \mathbf{E}^2[f\chi_a]$. However, there are two difficulties with using their technique directly. First, their algorithm uses calls to the function $f$ (membership queries), and we want an algorithm that uses only quantum example queries. Second, their technique works for Boolean ($\{-1,+1\}$-valued) functions, but the pairwise product $\alpha f$, viewed as representing a single function, is clearly not Boolean in general. We address each of these difficulties in turn below.

**5.1. Randomly selecting parities using a quantum example oracle.** Our first step in modifying `WDNF` to learn from a quantum example oracle for Boolean $f$—$QEX(f)$—rather than from a membership oracle is to show that we can randomly select parity functions with probability proportional to $\hat{f}^2$ using only $QEX(f)$. The proof of the following lemma presents the required algorithm `QSAMP`.

LEMMA 1. *There is a quantum program* `QSAMP` *that, given any quantum example oracle $QEX(f)$ for $f : \{0,1\}^n \to \{-1,+1\}$, returns $\chi_a$ with probability $\hat{f}^2(a)/2$.*
**Proof:** `QSAMP` begins by calling $QEX(f)$ on a blank tape to get the superposition

$$\frac{1}{2^{n/2}} \sum_x |x, f(x)\rangle.$$

`QSAMP` next replaces $f(x)$ with $(1-f(x))/2$ (call this $f'(x)$); note that $(-1)^{f'(x)} = f(x)$. Then we will apply a Fourier operator $F$ to the entire tape contents. We define $F$ as

$$F(|a\rangle) \equiv \frac{1}{2^{n/2}} \sum_y (-1)^{a \cdot y} |y\rangle$$

where $|a| = |y| = n$. This operation can be performed in $n$ steps by a quantum Turing machine [6]. Also recall that $(-1)^{a \cdot y} \equiv \chi_a(y) \equiv \chi_y(a)$. Thus applying $F$ gives us

$$
\begin{aligned}
& F\left(\frac{1}{2^{n/2}} \sum_x |x, f'(x)\rangle\right) \\
&= \frac{1}{2^{n/2}} \sum_x F(|x, f'(x)\rangle) \\
&= \frac{1}{2^{n+1/2}} \sum_{x,y,z} (-1)^{x \cdot y}(-1)^{f'(x)z}|y, z\rangle \\
&= \frac{1}{\sqrt{2}} \sum_y \mathbf{E}_x[\chi_y(x) f(x)]|y, 1\rangle \\
&\quad + \frac{1}{\sqrt{2}} \sum_y \mathbf{E}_x[\chi_y(x)]|y, 0\rangle \\
&= \frac{1}{\sqrt{2}} \sum_y \hat{f}(y)|y, 1\rangle + \frac{1}{\sqrt{2}}|\bar{0}, 0\rangle
\end{aligned}
$$

where $|y| = |x| = n$ and $|z| = 1$ and the final line follows by orthonormality of the parity basis. An *Obs* operation at this point produces $|y, 1\rangle$ with probability $\hat{f}^2(y)/2$, as desired. □

**5.2. Sampling parity according to coefficients of non-Boolean functions.** While algorithm QSAMP is a good first step toward relaxing HS's requirement for a membership oracle, it is not enough. As noted above, we need to sample the parity functions according to the coefficients of the non-Boolean function $\alpha f$. We will do this indirectly by sampling over individual bits of the function. First, note that we can limit the accuracy of $\alpha$ and still compute an adequate approximation to $\mathbf{E}[\alpha f \chi_a]$. Let $T$ denote the quantity on the right-hand side of equation (5.1). Also let $d = \lceil \log(3/T) \rceil = O(\log(s/\epsilon^3))$. Then since $0 \le \alpha(x) \le 1$ for all $x$, for $\theta(x) = \lfloor 2^d \alpha(x) \rfloor 2^{-d}$ we have

$$|\mathbf{E}[\theta f \chi_a]| \ge |\mathbf{E}[\alpha f \chi_a]| - \frac{T}{3}$$

for all $\chi_a$. Furthermore, note that any value $\theta$ taken on by $\theta(x)$ can be written as

$$\theta = \theta_1 2^{-1} + \theta_2 2^{-2} + \cdots + \theta_d 2^{-d} + k 2^{-d}$$

where for each $j \in [1, d]$, $\theta_j \in \{-1, +1\}$ and $k \in \{-1, 0, 1\}$. Thus

$$|\mathbf{E}[\theta f \chi_a]| \le \max_j |\mathbf{E}[\theta_j f \chi_a]| + \frac{T}{3}.$$

By (5.1), with high probability there exists $\chi_a$ such that $|\mathbf{E}[\alpha f \chi_a]| \ge T$. Therefore, for any such $\chi_a$ there is a fixed polynomial $p_1$ and an index $j$ such that $|\mathbf{E}[\theta_j f \chi_a]| \ge 1/p_1(s, 1/\epsilon)$. Furthermore, for each $j$, the number of $\chi_a$'s such that $|\mathbf{E}[\theta_j f \chi_a]| \ge 1/p_1(s, 1/\epsilon)$ is at most $p_1^2(s, 1/\epsilon)$ by Parseval's. This suggests that to find a weak approximator to $f$ with respect to $D$ defined as in equation (4.1), we define $\theta$ as above and then apply quantum sampling using each of the $d$ Boolean functions $\theta_j \cdot f$. We formalize this idea in the next section.

**Invocation:** $h \leftarrow \texttt{HS'}(n, s, QEX(f), \epsilon, \delta)$
**Input:** $n$; $s =$ size of DNF $f$; quantum example oracle $QEX(f)$; $\epsilon > 0$; $\delta > 0$
**Output:** with probability at least $1 - \delta$ (over random choices made by $\texttt{HS}'$), $\texttt{HS}'$ returns $h$ such that $\Pr[f = h] \geq 1 - \epsilon$

$\gamma$, $k$, $\alpha_r^i$, $r_i(x)$, $\Theta$, $\tilde{D}_i(x)$ are defined as in HS
$w_0 \leftarrow \texttt{WDNF'}(n, s, MEM(f), \mathcal{U}_n, \delta/2k)$
**for** $i \leftarrow 1, \ldots, k-1$ **do**
  $EX(f) \equiv Obs(QEX(f))$
  $E_\alpha \leftarrow \mathbf{Est}(\mathbf{E}_x[\alpha^i_{r_i(x)}], EX(f), \Theta/3, \delta/2k)$
  **if** $E_\alpha \leq 2\Theta/3$ **then**
    $k \leftarrow i$
    **break do**
  **endif**
  $w_i \leftarrow \texttt{WDNF'}(n, s, QEX(f), \tilde{D}_i(x), \delta/2k)$
**enddo**
$h(x) \equiv MAJ(w_0(x), w_1(x), \ldots, w_{k-1}(x))$
**return** h

FIG. 5.1. *Modified Harmonic Sieve ($\texttt{HS}'$) algorithm for learning DNF from a quantum example oracle $QEX(f)$.*

**Invocation:** $w_i \leftarrow \texttt{WDNF'}(n, s, QEX(f), \alpha, \delta)$
**Input:** $n$; $s =$ size of DNF $f$; $QEX(f)$; $\alpha$, an oracle that given $x$ returns $\alpha(x) \in [0, 1]$; $\delta > 0$
**Output:** with probability at least $1 - \delta$ (over random choices made by $\texttt{WDNF}'$), $\texttt{WDNF}'$ returns $h$ such that $\Pr_D[f = h] \geq 1/2 + 1/(8s + 4)$

$T \equiv c_2\epsilon^3/(3(2s + 1))$
$d \equiv \lceil \log(3/T) \rceil$
**for** $j \leftarrow 1, \ldots, n$ **do**
  **for** $\ell \leftarrow 1, \ldots, 2^{2d+1}\ln(2n/\delta)$ **do**
    $h = \texttt{QSAMP'}(QEX(f), \theta_j)$
    $EX(f) \equiv Obs(QEX(f))$
    $E_c \leftarrow \textbf{Est-a}(h, EX(f), \alpha, 1/(8s + 4), \delta/(n2^{2d+2}\ln(2n/\delta)))$
    **if** $|E_c| \geq 3/(8s + 4)$ **then**
      **return** $\mathrm{sign}(E_c) \cdot h$
    **endif**
  **enddo**
**enddo**
**return** 1

FIG. 5.2. *$\texttt{WDNF}'$ subroutine called by $\texttt{HS}'$. Procedure $\textbf{Est-a}(h, EX(f), \alpha, \epsilon, \delta)$, described in the text, estimates $\mathbf{E}_D[fh]$ within accuracy $\epsilon$ with probability at least $1-\delta$, where $D(x) = \alpha(x)/\sum_y \alpha(y)$. $\theta_j$ represents the $j$th bit of $\alpha$.*

**5.3. Modified HS algorithm.** Combining the above observations, a quantum algorithm for learning DNF with respect to uniform using a quantum example oracle is obtained by modifying HS as shown in Figures 5.1 through 5.3. The first difference between the new algorithm and the original is that **Est** will now be given a simulated example oracle $EX(f)$—simulated using $QEX(f)$ as explained in Section 3—in order to estimate expected values.

**Invocation:** $h \leftarrow$ `QSAMP'`$(QEX(f), \theta_j)$
**Input:** Quantum example oracle $QEX(f)$; Boolean function $\theta_j(x)$.
**Output:** For $y \neq \bar{0}$, returns $\chi_y$ with probability $\hat{f}^2(y)/2$. Returns $\chi_{\bar{0}}$ with probability $1/2 + \hat{f}^2(\bar{0})/2$.

1. Call $QEX(f)$ on blank tape
2. In superposition, replace (reversibly) $f(x)$ with $(1 - f(x)\theta_j(x))/2$.
3. In superposition, apply Fourier operator $F$ to the $n+1$ bits on the tape.
4. Perform an $Obs$ operation.
5. Return $\chi_y$, where $y$ represents the first $n$ bits on the tape.

FIG. 5.3. `QSAMP′` *subroutine called by* `WDNF′`.

The second, more important, difference is that in order to find a weak approximator (line 2 of `WDNF`) we will use the quantum approach outlined above. That is, for each value of $j \in [1, d]$ we will sample the $\chi_a$'s in such a way that the probability of seeing each $\chi_a$ is exactly $\mathbf{E}^2[\theta_j f \chi_a]/2$. We do this by running a modified `QSAMP` that, after calling $QEX(f)$, replaces $f(x)$ with $\theta_j(x) \cdot f(x)$ expressed as a $\{0,1\}$-valued function. This is a reversible operation because $x$ is still on the tape and $\theta_j^2(x) = 1$ for all $x$; therefore, this operation can be performed by a QTM. As shown in the previous section, there exists some $\chi_a$ and some $j$ such that $\chi_a$ is a weak approximator to $f$ and $|\mathbf{E}[\theta_j f \chi_a]| \geq T/3 \geq 2^{-d}$. Therefore, if we run the modified `QSAMP′` $2^{2d+1}\ln(2n/\delta)$ times for each value of $j$ then—with probability at least $1 - \delta/2$—at least one of the $\chi_a$'s returned by the quantum sampler will be this weak approximator.

Finally, we will again simulate $EX(f)$—this time within `WDNF′`—in order to test whether or not a given $h$ returned by `QSAMP′` is a weak approximator. In order to perform this test, $\mathbf{E}_D[fh]$ is estimated (where $D$ is the distribution defined by $\alpha$ as in equation (4.1)) by procedure **Est-a**. This procedure, given the uniform-distribution example oracle $EX(f)$ and the function $\alpha(x)$, simulates the example oracle $EX(f, D)$ using the same method as boost-by-filtering algorithms such as Freund's. Specifically, it queries $EX(f)$ and receives the pair $\langle x, f(x) \rangle$. It then flips a biased coin which produces heads with probability $\alpha(x)$. If the coin comes up heads then the algorithm uses this pair in subsequent processing. Otherwise, it discards the pair, queries $EX(f)$ again, and repeats the coin-flip test. It can be shown that this process efficiently simulates $EX(f, D)$ for the $\alpha$'s produced by our algorithm (see, e.g., [18]). Finally, given $EX(f, D)$, by a standard Chernoff argument we can estimate $\mathbf{E}_D[fh]$ with the required accuracy and confidence with a number of samples polynomial in the appropriate parameters. Overall, `WDNF′` allocates half of the confidence $\delta$ to estimating the $\mathbf{E}_D[fh]$'s. This gives us

THEOREM 2. *DNF is quantum learnable with respect to uniform.*

Also, $\widehat{PT}_1$, the class of functions expressible as a threshold of a polynomial number of parity functions, has the property that for every $f \in \widehat{PT}_1$ and every distribution $D$ there exists a parity function $\chi_a$ that weakly approximates $f$ with respect to $D$ [22]. This was the only property of DNF that we used in the above arguments. Therefore

THEOREM 3. *$\widehat{PT}_1$ is quantum learnable with respect to uniform.*

We now briefly examine the asymptotic time bound of this algorithm. First, Chernoff bounds tell us that to estimate the expected value of a Boolean variable to within an additive tolerance $\lambda$ with confidence $\delta$ requires a sample (and time) $\tilde{O}(\lambda^2)$ (the notation $\tilde{O}(\cdot)$ is the same as standard big-O notation with log factors suppressed; in this case, $\delta$ contributes only a log factor and therefore does not appear in the bound).

Using this fact and the earlier description of the algorithm, it is straightforward to show that the algorithm as given has a time bound of $\tilde{O}(ns^6/\epsilon^{12})$. This compares with a bound on the Harmonic Sieve of $\tilde{O}(ns^8/\epsilon^{18})$. Furthermore, as noted in [24], the $\epsilon^3$ factor that appears in the Harmonic Sieve can be brought arbitrarily close to $\epsilon^2$; with this improvement the bounds improve to approximately $\tilde{O}(ns^6/\epsilon^8)$ and $(\tilde{O}(ns^8/\epsilon^{12})$, respectively. While neither bound is a "small" polynomial, the quantum algorithm is somewhat of an improvement. It is also reasonable to suspect that future improvements in the running time of the original algorithm would lead to similar improvements in the quantum algorithm as well.

**6. Membership Oracle vs. Quantum Example Oracle.** In practice, it is not clear how a quantum example oracle could be constructed without using a membership oracle. Furthermore, because a QTM uses interference over an entire superposition to perform its computations, it might seem that perhaps there is some way to simulate a membership oracle given only a quantum example oracle by choosing a clever interference pattern. In this section we show that this is not the case.

DEFINITION 4. *We say that* membership queries can be quantum-example simulated for function class $\mathcal{F}$ *if there exists a $BQP$ algorithm $\mathcal{A}$ and a distribution $D$ such that for all $f \in \mathcal{F}$ and all $x$, running $\mathcal{A}$ on input $x$ with quantum example oracle $QEX(f, D)$ produces $f(x)$.*

THEOREM 5. *Membership queries cannot be quantum-example simulated for DNF.*

Before proving this theorem, we develop some intuition. Consider two functions $f_0$ and $f_1$ that differ in exactly one input $x$. Then the superpositions returned by $QEX(f_0, D)$ and $QEX(f_1, D)$ are very similar for "almost all" $D$. In particular, if we think of superpositions as vectors in an inner product space of dimension $2^n$, then there is in general an exponentially small angle between the superpositions generated by these two oracles. This angle will not be changed by unitary transformations. So in general, an observation will be unable to detect a difference between the superpositions produced by $QEX(f_0, D)$ and $QEX(f_1, D)$. Therefore a $BQP$ algorithm with only a quantum example oracle $QEX(f_i, D)$, $i \in \{0, 1\}$, will be unable to correctly answer a membership query on $x$ for both $f_0$ and $f_1$.

We now present two lemmas that will help us to formalize this intuition.

LEMMA 6. *Let $\mathcal{A}$ be a quantum algorithm that makes at most $t$ calls to $QEX(f, D)$ for $f : \{0, 1\}^n \rightarrow \{0, 1\}$. Then there is an equivalent quantum program (modulo a slowdown polynomial in $n$ and $t$) that makes all $t$ calls at the beginning of the program.*

**Proof:** Let $R_M$ be the unitary matrix representing the transitions of the QTM $M$. Let the configurations of $M$ be encoded by bit strings in any reasonable way. Then suppose $M$ is initially in a superposition

$$S_0 = \sum_y a_y|y\rangle,$$

where the sum is over all possible configurations $y$ of $M$. After a single transition $\mu$, $M$ will be in the superposition

$$S_1 = \sum_{y,z} a_y R_M[z, y]|z\rangle.$$

Now assume that all of the configurations $z$ with nonzero amplitude in $S_1$ cause $QEX(f, D)$ to be called (recall that by definition, either all of the configurations in a superposition cause this to happen or none of them do). The resulting superposition will be

$$S_2 = \sum_{x,y,z} \sqrt{D(x)} a_y R_M[z,y] |z, x, f(x)\rangle$$

(by $z, x, f(x)$ we mean the configuration that results when $(x, f(x))$ is appended to the tape contents specified by the configuration $z$). But notice that there is a machine $M'$ that, beginning with $S_0$, first calls $QEX(f, D)$, producing

$$S_1' = \sum_{x,y} \sqrt{D(x)} a_y |y, x, f(x)\rangle,$$

and then simulates the transition $\mu$. A technical detail of this simulation is that a transition that corresponds to writing in a blank cell of $y$'s tape necessitates shifting $x$ and $f(x)$ to the right one cell first. Thus $M'$ takes at most polynomially (in $n$) many steps and produces $S_2$ given $S_0$. A simple inductive argument completes the proof. □

Before presenting the next lemma, we need several definitions.

DEFINITION 7. *Define Obs over* any *linear combination of configurations (i.e., we no longer require that the sum of squared amplitudes be 1) as*

$$Obs\left(\sum_x u_x |x\rangle\right) = \sum_{x:x_1=1} |u_x|^2.$$

*Define the* length *of a linear combination of configurations* $S = \sum_x u_x |x\rangle$ *to be* $\|S\| = \sqrt{\sum_x |u_x|^2}$. *For any linear combination of configurations $S$ we define for* $i \in \{0, 1\}$

$$S^{(i)} = \sum_{x:x_1=i} u_x |x\rangle.$$

LEMMA 8. *Let $S_1$ and $S_2$ be superpositions and let $S$ be any linear combination of configurations. Let $W$ be any sequence of valid quantum operations. Then*

*1)* $Obs(S) \le \|S\|^2$.

*2)* $\|WS\| = \|S\|$.

*3)* $|\sqrt{Obs(S_1)} - \sqrt{Obs(S_2)}| \le \sqrt{Obs(S_1 - S_2)}$.

**Proof:** For 1) we have

$$Obs(S) = \|S^{(1)}\|^2 \le \|S^{(0)}\|^2 + \|S^{(1)}\|^2 = \|S\|^2.$$

Part 2) follows from the fact that $W$ is a unitary operation that preserves length.

To prove 3) we have

$$\begin{aligned} |\sqrt{Obs(S_1)} - \sqrt{Obs(S_2)}| &= \left| \|S_1^{(1)}\| - \|S_2^{(1)}\| \right| \\ &\le \|S_1^{(1)} - S_2^{(1)}\| \\ &= \sqrt{Obs(S_1 - S_2)}. \end{aligned}$$

□

**Proof of Theorem 5:** By way of contradiction, assume that $M$ is a QTM that can quantum-simulate membership queries for any DNF $h$ using calls to $QEX(h, D)$.

By Lemma 6, we can assume without loss of generality that all of the calls to $QEX(h, D)$ occur at the beginning of the $M$'s program. Take $f(x) = 0$ and $g(x) = x_1^{c_1} \wedge ... \wedge x_n^{c_n}$, where $x^d = 1$ if and only $x = d$. The second function is zero for all assignments to $x$ except $c = (c_1, \ldots, c_n)$. We want to use the simulator $M$ to find $h(c)$ for $h \in \{f, g\}$. The simulator will first make $t$ calls to $QEX(h, D)$ giving

$$S_h = \sum_{z_1,\ldots,z_t} \sqrt{D(z_1)\cdots D(z_t)} |z_1, h(z_1), \ldots, z_t, h(z_t)\rangle.$$

After that, the computation for both $f$ and $g$ is the same in the sense that both computations consist of a series of applications of $R_M$ to $S_h$. The superpositions $S_f$ and $S_g$ differ only in configurations

$$|z_1, h(z_1), \ldots, z_t, h(z_t)\rangle$$

where one of the $z_i$ is $c$.

Therefore

$$S_f = S_g + E_{f,g},$$

where

$$\begin{aligned} E_{f,g} = & \sum_{(\exists i) z_i = c} \sqrt{D(z_1)\cdots D(z_t)} |z_1, f(z_1), \ldots, z_t, f(z_t)\rangle \\ & - \sum_{(\exists i) z_i = c} \sqrt{D(z_1)\cdots D(z_t)} |z_1, g(z_1), \ldots, z_t, g(z_t)\rangle. \end{aligned}$$

If $W$ represents the sequence of transitions after the initial calls to $QEX(f, D)$, then at the end of the computation by $M$ we observe $Obs(WS_f)$ and $Obs(WS_g)$ for $f$ and $g$, respectively. By Lemma 8

$$\begin{aligned} & |\sqrt{Obs(WS_f)} - \sqrt{Obs(WS_g)}|^2 \\ & \le Obs(WE_{f,g}) \\ & \le \|WE_{f,g}\|^2 \\ & = \|E_{f,g}\|^2 \\ & = 2 \sum_{(\exists i) z_i = c} \left(\sqrt{D(z_1)\cdots D(z_t)}\right)^2 \\ & = 2(1 - (1 - D(c))^t) \\ & \le 2tD(c). \end{aligned}$$

For any fixed $D$, any fixed polynomial $p_1$, and large enough $n$, almost all choices of $c$ are such that $D(c) < 1/p_1(n)$. For all such $c$ and appropriately chosen $p_1$ the observations $Obs(WS_f)$ and $Obs(WS_g)$ are indistinguishable. □

While it is not possible to simulate membership queries in polynomial time given only a quantum example oracle, it is a simple matter to simulate a uniform quantum example oracle with membership queries.

LEMMA 9. *For every Boolean function $f$, $QEX(f,\mathcal{U})$ can be simulated by a QTM making a single call to $MEM(f)$.*

**Proof:** $QEX(f,\mathcal{U})$ can be simulated by applying the Fourier transform $F$ to the tape $|\bar{0}\rangle$ and then calling $MEM(f)$. □

Thus a membership oracle for $f$ is strictly more powerful than a uniform quantum example oracle for $f$.

**7. Concluding Remarks.** We have defined the notion of a quantum example oracle and argued that it is a natural quantum extension of the standard PAC example oracle. We then showed the learnability of DNF (and $\widehat{PT}_1$) with respect to uniform given access to such an oracle. Such an oracle is also shown to be weaker (with respect to uniform) than a membership oracle.

While we believe that these results are interesting theoretically, like many oracle results in complexity theory the practical significance of our results are not clear. Our algorithm at the very least offers some potential speed-up over an implementation of the unmodified Harmonic Sieve for learning DNF from a membership oracle on a quantum computer. Also, while we do not currently see how to implement a quantum example oracle without recourse to a membership oracle, it is conceivable that there may be some way to build a quantum example oracle from something (much) less than a full membership oracle, which could add substantially to our algorithm's relevance to practical machine-learning problems. Finally, we hope that these results may provide stepping stones toward answering the larger question of whether DNF can be learned by a QTM from a standard PAC example oracle. A positive answer to this question has potentially great practical significance.

An earlier version of this paper [14] claimed that our quantum algorithm would learn DNF in a generalized persistent noise model. This propagated an erroneous claim by Jackson that the Harmonic Sieve was noise-tolerant [22]. While a modified version of the Harmonic Sieve has subsequently been shown to tolerate persistent classification noise [23], we do not consider the quantum extension of that algorithm in this paper, leaving this problem open for further study.

**Acknowledgments.** The first author thanks Richard Cleve for an enlightening seminar on quantum computation.

REFERENCES

[1] H. Aizenstein, L. Hellerstein, and L. Pitt, *Read-thrice DNF is hard to learn with membership and equivalence queries*, in Proceedings of the 33rd Annual Symposium on Foundations of Computer Science, 1992, pp. 523–532.

[2] H. Aizenstein and L. Pitt, *Exact learning of read-twice DNF formulas*, in Proceedings of the 32nd Annual Symposium on Foundations of Computer Science, 1991, pp. 170–179.

[3] ———, *Exact learning of read-k disjoint DNF and not-so-disjoint DNF*, in Proceedings of the Fifth Annual Workshop on Computational Learning Theory, 1992, pp. 71–76.

[4] D. Angluin, M. Frazier, and L. Pitt, *Learning conjunctions of Horn clauses*, Machine Learning, 9 (1992), pp. 147–164.

[5] D. Angluin and M. Kharitonov, *When won't membership queries help?*, J. of Comput. Syst. Sci., 50 (1995), pp. 336–355. Earlier version appeared in 23rd STOC, 1991.

[6] E. Bernstein and U. Vazirani, *Quantum complexity theory*, in Proceedings of the 25th Annual ACM Symposium on Theory of Computing, 1993, pp. 11–20.

[7] A. Berthiaume and G. Brassard, *Oracle quantum computing*, in Proceedings of the Workshop on the Physics of Computation, IEEE Press, 1992, pp. 195–199.

[8] ———, *The quantum challenge to structural complexity theory*, in Proceedings of 7th IEEE Conference on Structure in Complexity Theory, 1992, pp. 132–137.

[9] A. Blum, M. Furst, J. Jackson, M. Kearns, Y. Mansour, and S. Rudich, *Weakly learning DNF and characterizing statistical query learning using Fourier analysis*, in Proceedings of the 26th Annual ACM Symposium on Theory of Computing, 1994, pp. 253–262.

[10] A. Blum, R. Khardon, E. Kushilevitz, L. Pitt, and D. Roth, *On learning read-k-satisfy-j DNF*, in Proc. 7th Annu. ACM Workshop on Comput. Learning Theory, ACM Press, New York, NY, 1994, pp. 110–117.

[11] A. Blum and S. Rudich, *Fast learning of k-term DNF formulas with queries*, J. of Comput. Syst. Sci., 51 (1995), pp. 367–373.

[12] J. Bruck, *Harmonic analysis of polynomial threshold functions*, SIAM Journal of Discrete Mathematics, 3 (1990), pp. 168–177.

[13] N. H. Bshouty, *Exact learning via the monotone theory*, in Proceedings of the 34th Annual Symposium on Foundations of Computer Science, 1993, pp. 302–311.

[14] N. H. Bshouty and J. C. Jackson, *Learning DNF over the uniform distribution using a quantum example oracle*, in Proceedings of the Eighth Annual Workshop on Computational Learning Theory, 1995, pp. 118–127.

[15] D. Deutsch, *Quantum theory, the Church-Turing principle and the universal quantum computer*, Proceesings of the Royal Society of London, A400 (1985), pp. 97–117.

[16] D. Deutsch and R. Jozsa, *Rapid solution of problems by quantum computation*, Proceesings of the Royal Society of London, A439 (1992), pp. 553–558.

[17] Y. Freund, *An improved boosting algorithm and its implications on learning complexity*, in Proceedings of the Fifth Annual Workshop on Computational Learning Theory, 1992, pp. 391–398.

[18] Y. Freund, *Boosting a weak learning algorithm by majority*, Information and Computation, 121 (1995), pp. 256–285. Also appeared in COLT90.

[19] Y. Freund and R. E. Schapire, *A decision-theoretic generalization of on-line learning and an application to boosting*, in Proceedings of the Second Annual European Conference on Computational Learning Theory, 1995.

[20] O. Goldreich and L. A. Levin, *A hard-core predicate for all one-way functions*, in Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing, 1989, pp. 25–32.

[21] T. R. Hancock, *Learning $2\mu$DNF formulas and $k\mu$ decision trees*, in Proceedings of the Fourth Annual Workshop on Computational Learning Theory, 1991, pp. 199–209.

[22] J. Jackson, *An efficient membership-query algorithm for learning DNF with respect to the uniform distribution*, in Proceedings of the 35th Annual Symposium on Foundations of Computer Science, 1994, pp. 42–53.

[23] J. Jackson, E. Shamir, and C. Shwartzman, *Learning with queries corrupted by classification noise*, in Proceedings of the Fifth Israel Symposium on the Theory of Computing and Systems, 1997.

[24] J. C. Jackson, *The Harmonic Sieve: A Novel Application of Fourier Analysis to Machine Learning Theory and Practice*, PhD thesis, Carnegie Mellon University, Aug. 1995. Available as technical report CMU-CS-95-183.

[25] M. Kearns, M. Li, L. Pitt, and L. Valiant, *On the learnability of Boolean formulae*, in Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, 1987, pp. 285–295.

[26] E. Kushilevitz and Y. Mansour, *Learning decision trees using the Fourier spectrum*, SIAM Journal on Computing, 22 (1993), pp. 1331–1348. Earlier version appeared in *Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing*, pages 455–464, 1991.

[27] E. Kushilevitz and D. Roth, *On learning visual concepts and DNF formulae*, in Proceedings of the Sixth Annual Workshop on Computational Learning Theory, 1993, pp. 317–326.

[28] N. Linial, Y. Mansour, and N. Nisan, *Constant depth circuits, Fourier transform, and learnability*, J. Assoc. Comput. Mach., 40 (1993), pp. 607–620. Earlier version appeared in *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 574–579, 1989.

[29] R. E. Schapire, *The strength of weak learnability*, Machine Learning, 5 (1990), pp. 197–227.

[30] P. W. Shor, *Algorithms for quantum computation: Discrete logarithms and factoring*, in Proceedings of the 35th Annual Symposium on Foundations of Computer Science, 1994, pp. 124–134.

[31] D. R. Simon, *On the power of quantum computation*, in Proceedings of the 35th Annual Symposium on Foundations of Computer Science, 1994, pp. 116–123.

[32] L. G. Valiant, *A theory of the learnable*, Comm. ACM, 27 (1984), pp. 1134–1142.

[33] A. C.-C. Yao, *Quantum circuit complexity*, in Proceedings of the 34th Annual Symposium on Foundations of Computer Science, 1993, pp. 352–361.