# Learning DNF from Random Walks

Nader Bshouty
Department of Computer Science
Technion
bshouty@cs.technion.ac.il

Elchanan Mossel*
Department of Statistics
University of California, Berkeley
mossel@microsoft.com

Ryan O'Donnell†
Department of Mathematics
MIT
odonnell@theory.lcs.mit.edu

Rocco A. Servedio‡
Department of Computer Science
Columbia University
rocco@cs.columbia.edu

**Abstract**

We consider a model of learning Boolean functions from examples generated by a uniform random walk on $\{0,1\}^n$. We give a polynomial time algorithm for learning decision trees and DNF formulas in this model. This is the first efficient algorithm for learning these classes in a natural passive learning model where the learner has no influence over the choice of examples used for learning.

*Supported by a Miller Postdoctoral Fellowship.

†Supported by NSF grant 99-12342.

‡Supported by an NSF Mathematical Sciences Postdoctoral Fellowship and by NSF grant CCR-98-77049. Most of this research was performed while at Harvard University.

# 1 Introduction

## 1.1 Motivation

One of the most notorious open questions in computational learning theory is whether it is possible to efficiently learn Boolean formulas in disjunctive normal form, or DNF, from random examples. This question was first posed by Valiant [33] in his seminal paper which formalized the Probably Approximately Correct (PAC) model of learning from independent random examples, and has remained stubbornly open ever since. DNF formulas achieve an attractive balance between expressiveness and clarity: any Boolean function can be represented by a sufficiently large DNF, yet DNF formulas are easily understood by humans and seem to be a natural form of knowledge representation.

Provably correct and efficient algorithms for learning DNF from random examples would be a powerful tool for the design of learning systems, and over the past two decades many researchers have sought such algorithms. Despite this intensive effort, the fastest algorithms to date for learning polynomial size DNF formulas in Valiant's original PAC model of learning (where the learner receives independent examples drawn from an arbitrary probability distribution over $\{0,1\}^n$) run in time $2^{\tilde{O}(n^{1/3})}$ [24]. Even if we only consider learning under the uniform distribution, the fastest known algorithms for learning polynomial size DNF from independent uniform examples run in time $n^{O(\log n)}$ [34].

Since learning DNF formulas from random examples seems to be hard, researchers have considered alternate models which give more power to the learning algorithm. The most popular of these is the model of *learning from membership queries*; in this model the learner has access to a black-box oracle for the function to be learned and thus can determine the value of the function on any inputs of its choice. Several polynomial time algorithms have been given for learning in this enhanced model. Kushilevitz and Mansour [26] gave a polynomial time membership query algorithm which can learn any polynomial size decision tree under the uniform distribution (i.e., the error of the final hypothesis is measured with respect to the uniform distribution on $\{0,1\}^n$). Building on the work of [26], Jackson [19] gave a polynomial time algorithm for learning polynomial size DNF formulas under uniform using membership queries.

While learning from membership queries is interesting in its own right, it represents a significant departure from traditional "passive" models of learning (such as the PAC model) in which the learning algorithm has no control over the data which it receives; the assumption that a learning algorithm can actively make queries is a strong one which may limit the usefulness of membership query learning algorithms. Thus an important goal is to design efficient algorithms for learning DNF formulas in natural "passive" learning models. Towards this end, researchers have considered several alternatives to the standard uniform distribution PAC model of learning from independent uniform random examples. Bshouty and Jackson [9] defined a model where the learner can access a uniform quantum superposition of all labelled examples, and showed that DNF formulas can be efficiently learned in this framework. More recently Bshouty and Feldman [2] showed that DNF can be efficiently learned in a model called SQ-$\mathcal{D}_\rho$, which is intermediate in power between standard uniform distribution learning and uniform distribution learning with membership queries; in this model the learner is allowed to make statistical queries about the target function under product distributions of the learner's choosing. While Bshouty and Feldman showed that this model is strictly weaker than the membership query model, it is still an "active" learning model since the learner selects the various distributions which will be used.

### 1.2 Our results: learning from random walks

We consider a natural variant of the standard uniform distribution PAC learning model, called the *(Uniform) Random Walk* model. In this model the learner's examples are not generated independently, but are produced sequentially according to a random walk on the Boolean hypercube (we give a precise definition of the model in Section 2.1). Such learning models have been previously studied [1, 14, 3] but no strong learning results were known. In contrast, we prove that DNF formulas are efficiently learnable in this model. Our main theorem is the following:

**Theorem 1** *The class of $s$-term DNF formulas on $n$ variables can be learned in the Random Walk model to accuracy $\epsilon$ and confidence $1-\delta$ in time* $\mathrm{poly}(n, s, 1/\epsilon, \log(1/\delta))$.

(We note that another class of functions which has been widely studied in learning theory is the class of Boolean decision trees [8, 12, 26]. Since any decision tree of size $s$ can be expressed as an $s$-term DNF, all of our results for learning DNF formulas immediately imply corresponding results for learning decision trees.) Our results give the first efficient algorithm for learning expressive classes of Boolean functions in a natural passive model of learning from random examples only.

We also introduce another learning model which we call the Noise Sensitivity model. We prove that DNF formulas can be efficiently learned in the Noise Sensitivity model as well. Since the Random Walk model can simulate the Noise Sensitivity model but the converse does not seem to be true, the Noise Sensitivity model is the weakest model in which we can learn DNF efficiently.

### 1.3 Previous Work

Variants of PAC learning in which the examples are not i.i.d., but rather are generated according to a stochastic process, were first studied by Aldous and Vazirani [1]. Despite being quite natural, these models have not been studied as intensively as other variants of PAC learning. Gamarnik [14] studied learning under stochastic processes but focused mainly on sample complexity and generalization error and did not give algorithms for learning specific concept classes. Bartlett, Fischer, and Höffgen [3] introduced the Random Walk model which we consider, which is arguably the simplest and most natural model of learning under a stochastic process. Bartlett et al. gave learning algorithms in the Random Walk model for some simple concept classes, namely Boolean threshold functions in which each weight is 0 or 1, parities of two monotone conjunctions, and DNF formulas with two terms.

## 2 Preliminaries

Throughout this paper TRUE and FALSE will be denoted by $-1$ and $+1$ respectively, so the $n$-dimensional Boolean hypercube is $\{+1, -1\}^n$. Since we will be dealing with random walks, we will refer to two different ways of altering a bit in a bit string. *Flipping* a bit $x_i \in \{+1, -1\}$ shall mean replacing $x_i$ with $-x_i$; *updating* the bit $x_i$ shall mean replacing $x_i$ with a uniformly random bit (equivalently, flipping it with probability $\frac{1}{2}$).

### 2.1 Learning models

Our learning models are based on the widely-studied uniform-distribution version of Valiant's "Probably Approximately Correct" (PAC) model [33] (see e.g. [4, 6, 7, 11, 10, 17, 16, 19, 20, 21, 22, 23, 27, 28, 30, 31, 32, 34, 35] and the references therein).

In uniform-distribution PAC learning, a learning problem is identified with a *concept class* $\mathcal{C} = \cup_{n \geq 1} \mathcal{C}_n$, which is simply a collection of Boolean functions, each $f \in \mathcal{C}_n$ being a function $\{+1,-1\}^n \to \{+1,-1\}$. The goal of a learning algorithm $A$ for $\mathcal{C}$ is to identify an unknown *target function* $f \in \mathcal{C}$ by using random examples from this function only. Algorithm $A$ takes as input an *accuracy parameter* $\epsilon$ and a *confidence parameter* $\delta$; it also has access to an *example oracle* $\mathsf{EX}(f)$ for the target function. Each time it is queried, $\mathsf{EX}(f)$ generates a point $x \in \{+1,-1\}^n$ and provides the learning algorithm with a *labeled example* $\langle x, f(x)\rangle$. The output of $A$ is a *hypothesis* $h$, which is a Boolean function $h\colon \{+1,-1\}^n \to \{+1,-1\}$ (in the form of, say, a circuit). The hypothesis $h$ is said to be *$\epsilon$-close* to $f$ if $\Pr[h(x) = f(x)] \geq 1 - \epsilon$ for $x$ drawn from the uniform distribution. We say that $A$ is a learning algorithm for $\mathcal{C}$ if for all $f \in \mathcal{C}$, when $A$ is run with example oracle $\mathsf{EX}(f)$, with probability at least $1 - \delta$ it outputs a hypothesis which is $\epsilon$-close to $f$. Here the probability is over the random examples $A$ receives from the oracle, and also over any internal randomness of $A$.

The measure of $A$'s efficiency is its running time; this includes both the time which $A$ takes to construct its hypothesis $h$ and the time required to evaluate $h$ on an input $x \in \{+1,-1\}^n$. In general we consider $A$'s running time as a function of $n$, $\epsilon^{-1}$, $\log(1/\delta)$, and a size parameter $s$ for the concept class. For the class of DNF formulas, $s$ is the number of terms in the DNF; for the class of decision trees, $s$ is the number of nodes in the tree.

Since uniform-distribution PAC learning seems to be difficult, relaxed models have also been considered. One common relaxation is to allow the learner to make *membership queries*. In the membership query model the learner has access to a *membership oracle* $MEM(f)$ which, on input $x \in \{+1,-1\}^n$, returns the value $f(x)$. This clearly gives the learner quite a bit of power, and departs from the traditional passive nature of learning from random examples.

We consider a different natural relaxation of the uniform-distribution PAC learning model, which we call the (Uniform) Random Walk model. The Random Walk model uses an oracle $\mathsf{RW}(f)$ which does not produce i.i.d. examples. Instead, the first point which $\mathsf{RW}(f)$ provides to the learning algorithm is uniformly random; succeeding points are given by a uniform random walk on the hypercube $\{+1,-1\}^n$. That is, if the $t$th example given to the learner is $\langle x, f(x)\rangle$, then the $(t+1)$st example will be $\langle x', f(x')\rangle$, where $x'$ is chosen by flipping a uniformly chosen random bit of $x$. Note that the Random Walk model is a passive model of learning; the learner sees only randomly generated examples and has no control over the data used for learning.

For completeness we remind the reader that an *$s$-term DNF formula* is an $s$-way OR of ANDs of Boolean literals. A *decision tree* is a rooted binary tree which is full (each internal node has 0 or 2 children) and which has each internal node labelled with a variable from $x_1, \ldots, x_n$ and each leaf labelled with a bit from $\{+1,-1\}$. Such a tree represents a Boolean function in the obvious way.

## 2.2 Fourier analysis

Fourier analysis of Boolean functions is a useful tool in uniform distribution learning. From this perspective Boolean functions are viewed as real-valued functions $f\colon \{+1,-1\}^n \to \mathbf{R}$ which happen to have range $\{+1,-1\}$. (For our analysis we will also consider non-Boolean functions on $\{+1,-1\}^n$ which do not map to $\{+1,-1\}$.)

For a set $S \subseteq [n]$, let $\chi_S\colon \{+1,-1\}^n \to \{+1,-1\}$ be the parity function $\chi_S(x) = \prod_{i \in S} x_i$. We sometimes write $\chi_S = x_S$. Since $\mathbf{E}[\chi_\emptyset] = 1$, $\mathbf{E}[\chi_S] = 0$ for $S \neq \emptyset$, and $\chi_S \chi_T = \chi_{S \Delta T}$ (where $\Delta$ denotes symmetric difference), the set of functions $\{\chi_S\}_{S \subseteq [n]}$ is an orthonormal basis for the vector space of functions $\{+1,-1\}^n \to \mathbf{R}$. We call $\hat{f}(S) = \mathbf{E}[f(x)\chi_S(x)]$ the *$S$ Fourier coefficient of $f$* and $f = \sum_{S \subseteq [n]} \hat{f}(S)\chi_S$ the *Fourier expansion of $f$*. By a small abuse of language, we call $\hat{f}(S)$ a Fourier coefficient of *degree* $|S|$.

We will consider various *norms* of $f$. We write $||f||_p$ to denote $\mathbf{E}[|f(x)|^p]^{1/p}$ for $p \geq 1$, and we write $||f||_\infty$ to denote $\max_{x\in\{+1,-1\}^n} |f(x)|$. Parseval's well known identity says that $||f||_2 = \sum_{S\subseteq[n]} \hat{f}(S)^2$. Note that Boolean functions $f : \{+1,-1\}^n \to \{+1,-1\}$ have $||f||_p = 1$ for all $p$.

Finally, we will often need to estimate the value of a bounded random variable to within some additive accuracy. Standard tail bounds imply that if $X$ is a random variable such that $|X| < c$ and $\lambda > 0$, then with $O(c^2 \log(1/\delta)/\lambda^2)$ draws from $X$ we can estimate $\mathbf{E}[X]$ to within $\pm\lambda$ with probability at least $1-\delta$.

# 3 The Random Walk model

In this section we make some straightforward but useful observations about how the Random Walk model compares with other learning models.

We first observe that having access to membership queries is at least as powerful as having examples generated from a random walk. In fact, one can show that uniform-distribution learning with membership queries is *strictly* easier than learning in the Random Walk model, under a standard cryptographic assumption (see Appendix A for the proof):

**Proposition 2** *If one-way functions exist then there is a concept class $\mathcal{C}$ which is learnable in polynomial time under the uniform distribution with membership queries, but is not learnable in polynomial time in the Random Walk model.*

We next describe a slight variation on the Random Walk oracle $\mathsf{RW}(f)$ which is of equivalent power. We call this variant the *updating Random Walk oracle*. In the updating Random Walk oracle, the first example given to the learner is again uniformly random, but each succeeding example is given by *updating* the previous one, and announcing the bit updated. That is, if the $t$th example given to the learner is $\langle x, f(x)\rangle$, then for the $(t+1)$'st example, the updating oracle picks $i \in [n]$ uniformly at random, forms $x'$ by updating the $i$th bit of $x$, and tells the learner $\langle i, x', f(x')\rangle$. Note that with probability $\frac{1}{2}$ we have $x = x'$ and the learner gains no new information.

It is easy to see that the usual Random Walk oracle and the updating oracle are of equivalent power. The updating oracle can trivially simulate the usual oracle with only constant factor slow-down. The reverse simulation is also easy. Given access to the original Random Walk oracle, to simulate the updating oracle the learner first tosses a fair coin. On heads, it draws a new example from the standard Random Walk oracle, noting which input bit was flipped. On tails, it chooses a random bit position $i$ and pretends that the updating oracle announced that the $i$th bit was updated but did not change. We will pass freely between these two versions of the Random Walk oracle; $\mathsf{RW}(f)$ will denote the original Random Walk oracle unless otherwise specified.

Finally, we note that learning under Random Walks is at least as easy as PAC learning under the uniform distribution. To see this we need only note that a learner with access to the Random Walk oracle $\mathsf{RW}(f)$ can simulate access to i.i.d. uniform examples. This is because the updating random walk on the hypercube mixes rapidly; if a learner discards $O(n \log n)$ successive examples from the updating oracle, then the next example will be uniformly random and independent of all previous examples.[1]

[1] Strictly speaking, the example will only be very nearly independent and uniformly random; more precisely we have that with probability $1-\delta$ the example is independently and uniformly random, where $\delta$ goes to 0 exponentially fast (i.e. we allocate some portion of the confidence parameter $\delta$ for this). Throughout this paper all considerations involving $\delta$ are standard and we will frequently gloss over them for clarity.

## 4 The Bounded Sieve

In this section we describe tools previously used to learn decision trees and DNF, and identify those which we will use for learning under Random Walks.

Kushilevitz and Mansour [26] first gave a polynomial time membership query algorithm for learning decision trees under the uniform distribution. Their algorithm uses a subroutine (often called KM), based on the list-decoding algorithm of Goldreich and Levin [15], which finds and estimates all "large" Fourier coefficient of the target function using membership queries. Subsequently Jackson [19] extended the KM algorithm and combined it with the hypothesis boosting algorithm of Freund [13] to give the *Harmonic Sieve* algorithm, which uses membership queries to learn DNF under the uniform distribution in polynomial time. Bshouty and Feldman [2] later observed that a certain algorithmic variant of KM, which they called the *Bounded Sieve*, is all that is necessary for Jackson's algorithm to work.

We now define the Bounded Sieve. Performing the Bounded Sieve essentially entails finding all large, low-degree Fourier coefficients:

**Definition 3** *Let $f\colon \{+1,-1\}^n \to \mathbf{R}$ be a real-valued Boolean function. An algorithm with some form of oracle access to $f$ is said to perform the* Bounded Sieve *if, given input parameters $\theta > 0$, $\ell \in [n]$, and $\delta > 0$, with probability at least $1-\delta$ it outputs a list of subsets of $[n]$ such that every set $S \subseteq [n]$ satisfying $|S| \leq \ell$ and $\hat{f}(S)^2 \geq \theta$ appears in the list.*

Bshouty and Feldman implicitly observe that the following results follow from Kushilevitz-Mansour [26] and Jackson [19]:

**Theorem 4** *Let $\mathcal{A}$ be an algorithm performing the Bounded Sieve which runs in time $t(n, ||f||_\infty, \theta, \ell, \delta)$. Then:*

- *[26] there is a $\mathrm{poly}(n, 1/\epsilon, \log(1/\delta)) \cdot t(n, 1, \epsilon/8s, \log(8s/\epsilon), \delta)$ time algorithm which $(\epsilon,\delta)$-learns $n$-variable, size-$s$ decision trees using $\mathcal{A}$ as a black box and access to independent uniform random examples for $f$; and*
- *[19] for $T = \mathrm{poly}(n, s, 1/\epsilon, \log(1/\delta))$, there is a $T \cdot t(n, \mathrm{poly}(1/\epsilon), 1/(2s+1), \log(s/\mathrm{poly}(\epsilon)), \delta/T)$ time algorithm which $(\epsilon,\delta)$-learns $n$-variable, $s$-term DNF formulas using $\mathcal{A}$ as a black box and independent uniform random examples.*

We will show that the Bounded Sieve can be performed under the Random Walk model in time $\mathrm{poly}(n, ||f||_\infty, 1/\theta, 2^\ell, \log(1/\delta))$. From this we get Theorem 1: $s$-term DNF can be learned in the Random Walk model in time $\mathrm{poly}(n, s, 1/\epsilon, \log(1/\delta))$.

## 5 The Bounded Sieve via Noise Sensitivity estimates

The KM algorithm works by estimating certain sums of squares of the Fourier coefficients of the target function. We show that the Bounded Sieve can be performed in the required time bound given access to certain weighted sums of squares of Fourier coefficients.

**Definition 5** *Given $f\colon \{+1,-1\}^n \to \mathbf{R}$, $I \subseteq [n]$, and $\rho \in (0,1)$ a constant, define:*

$$\mathcal{T}_\rho^{(I)}(f) = \sum_{S \supseteq I} \rho^{|S|} \hat{f}(S)^2. \tag{1}$$

*When $f$ and $\rho$ are clear from context, we write simply $\mathcal{T}(I)$.*

Note that $\mathcal{T}(I)$ is monotone in $I$ in the sense that $I \subseteq J$ implies $\mathcal{T}(I) \geq \mathcal{T}(J)$. Weighted sums of squares as in (1) frequently arise in the study of the *noise sensitivity* of Boolean functions, see e.g. [5, 29]. In particular, the noise sensitivity of $f$ at $\frac{1}{2} - \frac{1}{2}\rho$, denoted $\mathsf{NS}_{\frac{1}{2}-\frac{1}{2}\rho}(f)$, equals $1 - 2\mathcal{T}_\rho^{(\emptyset)}(f)$ [10, 5, 29].

We show that if $\mathcal{T}_\rho^{(I)}(f)$ can be estimated efficiently then the Bounded Sieve can be performed efficiently. To prove this we first need a lemma which bounds the sum of $\mathcal{T}_\rho^{(I)}(f)$ over all sets $I$ of some fixed size:

**Lemma 6** *For any $f\colon \{+1,-1\}^n \to \mathbf{R}$, $0 \leq j \leq n$, and $\rho \in (0,1)$, we have $\sum_{|I|=j} \mathcal{T}_\rho^{(I)}(f) \leq ||f||_\infty^2 \rho^j (1-\rho)^{-j-1}$.*

**Proof:** Writing $\mathcal{T}(I) = \mathcal{T}_\rho^{(I)}(f)$, we have:

$$\begin{aligned}
\sum_{|I|=j} \mathcal{T}(I) &= \sum_{|I|=j} \sum_{S \supseteq I} \rho^{|S|} \hat{f}(S)^2 \\
&= \sum_{|S| \geq j} \binom{|S|}{j} \rho^{|S|} \hat{f}(S)^2 \\
&\leq \sum_{|S| \geq j} \hat{f}(S)^2 \sum_{t=j}^{\infty} \binom{t}{j} \rho^t \\
&= ||f||_2^2 \rho^{-1} \left( \frac{\rho}{1-\rho} \right)^{j+1} \\
&\leq ||f||_\infty^2 \rho^j (1-\rho)^{-j-1},
\end{aligned}$$

where the third equality follows from standard generating function identities and the fact that $\rho \in (0,1)$. $\square$

We now show how to perform the Bounded Sieve given the ability to estimate $\mathcal{T}_\rho^{(I)}(f)$ for any fixed $\rho \in (0,1)$:

**Theorem 7** *Fix $\rho \in (0,1)$. Let $\mathcal{B}$ be an algorithm with some form of oracle access to $f$ which runs in time $u(n, \rho, |I|, \gamma, \delta)$ and, with probability $1-\delta$, outputs an estimate of $\mathcal{T}_\rho^{(I)}(f)$ accurate to within $\pm\gamma$. Then there is an algorithm using black-box access to $\mathcal{B}$ and independent uniform random examples from $f$ which performs the Bounded Sieve in time $U \cdot \log(1/\delta) u(n, \rho, \ell, \rho^\ell \theta/2, \delta/U)$, where $U = \mathrm{poly}(n, ||f||_\infty, 1/\theta, (1-\rho)^{-\ell})$.*

**Proof:** For simplicity in this proof we assume that all estimates from are correct to within the desired tolerance; the full analysis for $\delta$ is standard and is omitted.

Consider the directed graph on all subsets of $[n]$ in which there is an edge from $I$ to $J$ if $I \subset J$ and $|J \setminus I| = 1$. The nodes $I$ are divided into $n$ layers according to the value of $|I|$. Our Bounded Sieve algorithm for $f$ performs a breadth-first search on this graph, starting at the node $I = \emptyset$. For each active node in the search, the algorithm estimates $\mathcal{T}(I)$ to within $\pm\rho^\ell\theta/2$ and $\hat{f}(I)^2$ to within $\pm\theta/2$. The first estimate uses $\mathcal{B}$ and takes time $u(n, \rho, |I|, \rho^\ell\theta/2)$. The second estimate is performed via empirical sampling using independent uniform random examples from $f$ and takes time $\mathrm{poly}(n, ||f||_\infty, 1/\theta)$. If the estimate of $\hat{f}(I)^2$ has magnitude at least $\theta/2$ then the algorithm adds $I$ to the list of $f$'s large Fourier coefficients. Thus if $\hat{f}(I)^2 \geq \theta$ then $I$ will certainly be added to the list.

The breadth-first search proceeds to the neighbors of $I$ only if $|I| < \ell$ and the estimate of $\mathcal{T}(I)$ is at least $\rho^\ell\theta/2$. The proof is complete given two claims: first, we claim the algorithm finds all Fourier coefficients $\hat{f}(S)$ with $\hat{f}(S)^2 \geq \theta$ and $|S| \leq \ell$; and second, we claim the algorithm ends its search after visiting at most $\mathrm{poly}(||f||_\infty, 1/\theta, (1-\rho)^{-\ell})$ sets $I$.

For the first claim, note that if $|S| \leq \ell$ and $\hat{f}(S)^2 \geq \theta$, then this Fourier coefficient contributes at least $\rho^\ell\theta$ to the value of $\mathcal{T}(I)$ for all $I \subseteq S$. Thus by the monotonicity of $\mathcal{T}$, the search will proceed all the way to $S$.

For the second claim, note that by Lemma 6, the number of "active nodes" at layer $j$ in the breadth-first search can be at most:

$$\frac{||f||_\infty^2 \rho^j (1-\rho)^{-j-1}}{\rho^j\theta/2} = 2||f||_\infty^2 \theta^{-1}(1-\rho)^{-j-1}.$$

Since $j$ is never more than $\ell$, the total number of nodes the breadth-first search ever encounters is at most $2||f||_\infty^2\theta^{-1}(1-\rho)^{-(\ell+1)} = \mathrm{poly}(||f||_\infty, 1/\theta, (1-\rho)^{-\ell})$, as claimed. $\square$

By combining Theorems 4 and 7, we get:

**Corollary 8** *If there is an algorithm $\mathcal{B}$ with some form of oracle access to $f\colon \{+1,-1\}^n \to \mathbf{R}$ which, for some $\rho \in (0,1)$, can with probability $1-\delta$ estimate $\mathcal{T}_\rho^{(I)}(f)$ to within $\pm\gamma$ in time* $\mathrm{poly}(n, ||f||_\infty, 1/\gamma, [\rho(1-\rho)]^{-|I|}, \delta)$*, then $s$-term DNF on $n$-variables can $(\epsilon,\delta)$-learned using black-box access to $\mathcal{B}$ and independent uniform random examples from $f$ in time* $\mathrm{poly}(n, s^{c_0}, \epsilon^{-c_0}, \log(1/\delta))$*, where $c_0 = -\log(\rho(1-\rho))$.*

# 6 Estimating $\mathcal{T}_\rho^{(I)}(f)$ via Random Walks

To complete the proof of Theorem 1, we need to show how to estimate $\mathcal{T}_\rho^{(I)}(f)$ as in Corollary 8 for some constant $\rho \in (0,1)$ under the Random Walk model. This is done in the following theorem:

**Theorem 9** *Let $f\colon \{+1,-1\}^n \to \mathbf{R}$, let $I \subseteq [n]$, and let $\rho \in (0,1)$. Then there is an algorithm using access to the Random Walk oracle $\mathsf{RW}(f)$ which with probability $1-\delta$ estimates $\mathcal{T}_\rho^{(I)}(f)$ to within $\pm\gamma$ in time* $\mathrm{poly}(n, ||f||_\infty, 1/\gamma, \log(1/\delta), \log(1/\rho), \max\{1, (1/\rho-1)^{-|I|}\})$.

**Proof:** Let $\alpha = \ln(1/\rho)$, let $\lambda = \alpha n/2$, and let $M$ be a Poisson distributed random value with mean $\lambda$; i.e., $M$ is chosen to be $m \in \mathbf{Z}^{\geq 0}$ with probability $p_m = \frac{e^{-\lambda}\lambda^m}{m!}$. Note that $M = O(\lambda) = O(\log(1/\rho)n)$ with very high probability. Let $x$ be a uniform random string in $\{+1,-1\}^n$, and let $y$ be obtained by taking a random walk from $x$ of length exactly $M$. Let $T$ be a random subset of $I$ chosen by selecting each index in $I$ to be in $T$ independently with probability $\frac{1}{1+\rho}$. We claim that:

$$\mathop{\mathbf{E}}_{T}\mathop{\mathbf{E}}_{M,x,y}[(-1)^{|I\setminus T|}x_T y_T f(x)f(y)] = (1/\rho-1)^{|I|}\sum_{S\supseteq I}\rho^{|S|}\hat{f}(S)^2 = (1/\rho-1)^{|I|}\mathcal{T}_\rho^{(I)}(f). \tag{2}$$

Note that we can generate the pairs $(x,y)$ and their labels $f(x), f(y)$ using the Random Walk oracle for $f$. Since $|(-1)^{|I\setminus T|}x_T y_T f(x)f(y)| \leq ||f||_\infty^2$, by standard empirical averaging we can estimate $\mathcal{T}_\rho^{(I)}$ to within $\pm\gamma(1/\rho-1)^{-|I|}$ in the claimed time bound.

We now prove Equation (2). We begin by analyzing the quantity $\mathbf{E}_{M,x,y}[x_U y_V]$ where $U, V \subseteq [n]$.

Suppose first that $U \neq V$; in particular, suppose that $i \in V \setminus U$. Then for each way of choosing $M, x, y$, there is a corresponding way to choose $M, x, y$ which differs only in that $x$ and $y$ each have the $i$th bit flipped. Since $x$ is chosen uniformly, these two outcomes clearly have the same probability. But since $i \in V \setminus U$, the values of $x_U y_V$ are opposite in these two outcomes. Pairing up all outcomes in this way, we have that $E_{M,x,y}[x_U y_V] = 0$. A similar argument holds when $U \setminus V \neq \emptyset$.

It remains to consider $\mathbf{E}_{M,x,y}[x_U y_U] = \sum_{m \geq 0} p_m \mathbf{E}_{x,y}[(xy)_U \mid M = m]$. If we let $\mathbf{1}_i$ denote the random variable which is 1 if the $i$th step of the random walk is in $U$, and 0 otherwise, we have $\mathbf{E}_{x,y}[(xy)_U \mid M = m] = \mathbf{E}[\prod_{i=1}^m (-1)^{\mathbf{1}_i}] = \prod_{i=1}^m \mathbf{E}[(-1)^{\mathbf{1}_i}] = (1 - 2|U|/n)^m$. Thus $\mathbf{E}_{M,x,y}[x_U y_U] = \sum_{m \geq 0} p_m (1 - 2|U|/n)^m = \exp(-\lambda) \cdot \exp(\lambda(1 - 2|U|/n)) = \exp(\lambda(-2|U|/n)) = \rho^{|U|}$.

Now we can analyze Equation (2):

$$
\begin{aligned}
\mathbf{E}_T \mathbf{E}_{M,x,y}[(-1)^{|I \setminus T|} x_T y_T f(x) f(y)] &= \mathbf{E}_T \left[ (-1)^{|I \setminus T|} \sum_{U,V \subseteq [n]} \hat{f}(U) \hat{f}(V) \mathbf{E}_{M,x,y}[x_{T \Delta U} y_{T \Delta V}] \right] \\
&= \sum_{U \subseteq [n]} \hat{f}(U)^2 \mathbf{E}_T[(-1)^{|I \setminus T|} \mathbf{E}_{M,x,y}[x_{T \Delta U} y_{T \Delta U}]] \\
&= \sum_{U \subseteq [n]} \hat{f}(U)^2 \mathbf{E}_T[(-1)^{|I \setminus T|} \rho^{|T \Delta U|}] \\
&= \sum_{U \subseteq [n]} \hat{f}(U)^2 \rho^{|U|} \mathbf{E}_T \left[ \left( \prod_{j \in I \cap U} -(-\rho)^{-\mathbf{1}_j} \right) \left( \prod_{j \in I \setminus U} -(-\rho)^{\mathbf{1}_j} \right) \right],
\end{aligned}
$$

where for $j \in I$, $\mathbf{1}_j$ is the indicator variable for $j \in T$. Note that $\mathbf{E}[-(-\rho)^{-\mathbf{1}_j}] = \frac{\rho^{-1}}{1+\rho} + \frac{-\rho}{1+\rho} = (1/\rho - 1)$ whereas $\mathbf{E}[-(-\rho)^{\mathbf{1}_j}] = \frac{\rho}{1+\rho} + \frac{-\rho}{1+\rho} = 0$. Thus $\mathbf{E}_T \mathbf{E}_{M,x,y}[(-1)^{|I \setminus T|} x_T y_T f(x) f(y)] = (1/\rho - 1)^{|I|} \sum_{U \supseteq I} \rho^{|U|} \hat{f}(U)^2$ as claimed. □

# 7 Learning DNF in the Noise Sensitivity model

Since we can learn DNF in polynomial time in the Random Walk model, it is natural to ask: What is the weakest model in which we can learn DNF efficiently (with respect to the uniform distribution)? Toward this end, we now introduce a new passive model of learning from random examples, the *Noise Sensitivity* model.

For each value of $\rho \in [0, 1]$ the *$\rho$-Noise Sensitivity example oracle* $\mathsf{NS\text{-}EX}_\rho(f)$ is defined as follows. At each invocation, $\mathsf{NS\text{-}EX}_\rho(f)$ independently selects a uniform input $x \in \{+1, -1\}^n$, forms $y$ by flipping each bit of $x$ independently with probability $\frac{1}{2} - \frac{1}{2}\rho$, and outputs the tuple $\langle x, f(x), y, f(y) \rangle$. We note that this oracle is equivalent to an "updating" $\rho$-Noise Sensitivity oracle which outputs $\langle x, f(x), y, f(y), S \rangle$ where $x$ is independent and uniform over $\{+1, -1\}^n$, $y$ is formed by updating each bit of $x$ independently with probability $1 - \rho$, and $S \subseteq [n]$ is the set of indices of $x$ which were updated to yield $y$. This is because the extra information $S$ can be simulated from access to the usual $\mathsf{NS\text{-}EX}_\rho(f)$ oracle: upon receiving $\langle x, f(x), y, f(y) \rangle$ from $\mathsf{NS\text{-}EX}_\rho(f)$, the learner constructs $S$ by including each bit position in which $x$ and $y$ differ with probability 1, and including each other bit position independently with probability $\frac{1-\rho}{1+\rho}$. A straightforward calculation shows that this gives the right distribution.

## 7.1 Comparison to other models

Let us consider the different learning models obtained by varying $\rho$. The cases $\rho = 0$ and $\rho = 1$ are trivially equivalent to the usual PAC model of learning under the uniform distribution. For values $\rho \in (0, 1)$, learning with $\mathsf{NS\text{-}EX}_\rho(f)$ is clearly at least as easy as learning under the uniform distribution. For different constants $\rho \neq \rho' \in (0,1)$ it seems that the $\rho$- and $\rho'$-Noise Sensitivity models may be of incomparable strength. We will show that DNF can be efficiently learned in the $\rho$-Noise Sensitivity model for any constant $\rho \in (0,1)$, and thus learning in each of these models seems to be strictly easier than learning under the usual uniform distribution PAC model.

We now show that each $\rho$-Noise Sensitivity model is a weakening of the Random Walk model:

**Proposition 10** *For any $\rho \in [0,1]$, any $\rho$-Noise Sensitivity learning algorithm can be simulated in the Random Walk model with only a multiplicative $O(n \log n)$ slowdown in running time.*

**Proof:** Fix $\rho \in [0,1]$. We show how to simulate the oracle $\mathsf{NS\text{-}EX}_\rho$ using the Random Walk model's updating oracle. To get an example $\langle x, f(x), y, f(y)\rangle$, we first draw $O(n \log n)$ examples from the updating oracle to get to a uniformly random point $x$; this point and its label $f(x)$ will be the first part of our $\mathsf{NS\text{-}EX}_\rho$ example. We now need to generate a point $y$ which is formed from $x$ by *updating* each bit with probability $1-\rho$. This is equivalent to drawing a value $u \sim \mathrm{Bin}(n, 1-\rho)$ and updating a random subset of precisely $u$ of $x$'s bits. Accordingly, in our simulation we randomly choose an integer $0 \le u \le n$ according to $\mathrm{Bin}(n, 1-\rho)$. We then repeatedly draw examples from the Random Walk updating oracle until $u$ distinct bit positions have been updated. The resulting point is distributed as if a random subset of $u$ bit positions had been updated (note that updating an input position more than once has no extra effect). Therefore, if we call this point $y$ and output $\langle x, f(x), y, f(y)\rangle$, then the simulation of $\mathsf{NS\text{-}EX}_\rho$ is correct. (Note that even if $u$ is as large as $n$, it only takes $O(n \log n)$ samples to get a string in which all $u = n$ distinct bit positions of $x$ have been updated.) $\square$

## 7.2 Learning DNF under $\mathsf{NS\text{-}EX}_\rho$

Having shown that the Noise Sensitivity models are no stronger than the Random Walk model, we now show that for any constant $\rho \in (0,1)$, DNF can be learned efficiently under $\mathsf{NS\text{-}EX}_\rho$.

**Theorem 11** *Let $\rho \in (0,1)$, let $f\colon \{+1,-1\}^n \to \mathbf{R}$, and let $I \subseteq [n]$. There is an algorithm using access to $\mathsf{NS\text{-}EX}_\rho(f)$ which with probability $1-\delta$ estimates $\mathcal{T}_\rho^{(I)}(f)$ to within $\pm\gamma$ in time* $\mathrm{poly}(n, ||f||_\infty, 1/\gamma, (1-\rho)^{-|I|}, 2^{|I|}, \log(1/\delta))$.

**Proof:** Given $\rho$ and $I$, consider the joint probability distribution $\mathcal{D}_\rho^{(I)}$ defined over pairs of strings $(x, y) \in (\{+1,-1\}^n)^2$ as follows: First $x$ is picked uniformly at random; then $y$ is formed by updating each bit of $x$ in $I$ with probability 1 and updating each bit of $x$ not in $I$ with probability $1-\rho$. We claim that access to pairs from this distribution and their values under $f$ can be simulated by access to $\mathsf{NS\text{-}EX}_\rho(f)$, with slowdown $\mathrm{poly}((1-\rho)^{-|I|})$. This simulation is done simply by calling the updating version of the $\mathsf{NS\text{-}EX}_\rho(f)$ oracle repeatedly until it returns a tuple $\langle x, f(x), y, f(y), S\rangle$ which has $I \subseteq S$. The pair $(x, y)$ thus generated is indeed drawn precisely from $\mathcal{D}_\gamma^{(I)}$, and the overhead of the simulation is $\mathrm{poly}((1-\rho)^{-|I|})$ with high probability.

Define $\mathcal{T}'(I)$ to be $\mathbf{E}_{(x,y)\leftarrow\mathcal{D}_\gamma^{(I)}}[f(x)f(y)]$. Since access to $\mathsf{NS\text{-}EX}_\rho(f)$ lets us obtain pairs from $\mathcal{D}_\rho^{(I)}$ and their values under $f$, we can estimate $\mathcal{T}'(I)$ to within $\pm\nu$ with probability $1-\delta'$ by empirical averaging in time $\mathrm{poly}(n, ||f||_\infty, 1/\nu, (1-\rho)^{-|I|}, \log(1/\delta'))$. We now observe that the quantity $\mathcal{T}'(I)$ is very closely related to $\mathcal{T}(I)$; in particular, an argument very similar to the one used in the proof of Theorem 9 gives the following claim:

**Claim 12** $\mathcal{T}'(I) = \sum_{S \cap I = \emptyset} \rho^{|S|} \hat{f}(S)^2$.

Let us now define $\mathcal{T}''(I) = \mathcal{T}'(\emptyset) - \mathcal{T}'(I)$; this is also a quantity we can estimate to within $\pm\nu$ in time $\text{poly}(n, ||f||_\infty, 1/\nu, (1-\rho)^{-|I|}, \log(1/\delta'))$. We have $\mathcal{T}''(I) = \sum_{S \cap I \neq \emptyset} \rho^{|S|} \hat{f}(S)^2$. Thus if we estimate $\mathcal{T}''(J)$ for all $J \subseteq I$, it is straightforward to estimate $\mathcal{T}(I) = \sum_{S \supseteq I} \rho^{|S|} \hat{f}(S)^2$ using inclusion-exclusion. Since there are only $2^{|I|}$ subsets $J$ of $I$, we can take $\nu = \gamma/2^{|I|}$ and $\delta' = \delta/2^{|I|}$ and thus estimate $\mathcal{T}(I)$ to within $\pm\gamma$ with probability $1-\delta$ in time $\text{poly}(n, ||f||_\infty, 1/\gamma, (1-\rho)^{-|I|}, 2^{|I|}, \log(1/\delta))$, as claimed. □

**Note:** We close by observing that for any constant $\rho \in (0,1)$ the $\rho$-Noise Sensitivity model is similar to a "partially observable Random Walk" model in which examples are generated as in the usual Random Walk scenario but the learner is only allowed to observe the location of the random walk once every $C \cdot n$ steps for some constant $C > 0$ (depending on $\rho$). Using techniques similar to the above, it can be shown that DNF are efficiently learnable in such a partially observable Random Walk model; we omit the details.

# 8 Discussion

## 8.1 Noise tolerance

We observe that our algorithms can tolerate any rate $\eta < \frac{1}{2}$ of random classification noise in the labelling of examples. More precisely, suppose that in each labelled example received by the learner the correct label $f(x)$ is corrupted (flipped) with probability $\eta$ and this possibly noisy label is instead presented to the learner. A standard analysis shows that our algorithms will still succeed, at the cost of a $\text{poly}(\frac{1}{1-2\eta})$ factor slowdown in running time (the number of samples we must use in order to estimate $\mathcal{T}(I)$ to within the desired accuracy will increase by this factor).

## 8.2 Lower bounds on sample size

Our algorithm uses a random walk sample of size $\text{poly}(n, s)$ to learn decision trees or DNF of size $s$. We observe here that any Random Walk algorithm for these classes must have a polynomial sample size dependence on both $n$ and $s$ (the proof is in Appendix B):

**Claim 13** *Learning the class of DNF expressions of size $s$ (or decision trees of size $s$) in the Random Walk model requires sample size $\Omega(\frac{sn}{\log s})$.*

This is in contrast with the membership query model in which $\text{poly}(s, \log n)$ queries are sufficient for a polynomial time algorithm to learn $s$-term DNF or size-$s$ decision trees under the uniform distribution [10].

## 8.3 Questions for further work

An interesting question for further work is whether a broader class of Boolean functions than polynomial size DNF can be shown to be efficiently learnable in the Random Walks model. Jackson's uniform distribution membership query algorithm for learning DNF can in fact learn any polynomial-weight threshold-of-parity circuit (sometimes called a TOP) in polynomial time. Since any $s$-term DNF on $n$ variables can be expressed as a TOP of weight $O(ns^2)$ [19, 25], this class properly includes the class of polynomial size DNF (the inclusion is proper since DNF formulas require exponential size to compute the parity function). A direct application of our approach to majority of parity does not seem to work since the parity functions can be as large as $\Theta(n)$. It would be interesting to devise a stronger algorithm which can efficiently learn an arbitrary polynomial weight majority of parities using random walks.

## References


[1] D. Aldous and U. Vazirani. A Markovian extension of Valiant's learning model. In *Proceedings of the Thirty-First Symposium on Foundations of Computer Science*, pages 392–396, 1990.

[2] N. Bshouty andV. Feldman. On using extended statistical queries to avoid membership queries. *Journal of Machine Learning Research*, 2:359–395, 2002.

[3] P. Bartlett, P. Fischer, and K.U. Höffgen. Exploiting random walks for learning. In *Proceedings of the Seventh Annual Conference on Computational Learning Theory*, pages 318–327, 1994.

[4] M. Bellare. A technique for upper bounding the spectral norm with applications to learning. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 62–70, 1992.

[5] I. Benjamini, G. Kalai, and O. Schramm. Noise sensitivity of boolean functions and applications to percolation. *Inst. Hautes Études Sci. Publ. Math.*, 90:5–43, 1999.

[6] 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 Twenty-Sixth Annual Symposium on Theory of Computing*, pages 253–262, 1994.

[7] D. Boneh and R. Lipton. Amplification of weak learning over the uniform distribution. In *Proceedings of the Sixth Annual Workshop on Computational Learning Theory*, pages 347–351, 1993.

[8] N. Bshouty. Exact learning via the monotone theory. *Information and Computation*, 123(1):146–153, 1995.

[9] N. Bshouty and J. Jackson. Learning DNF over the uniform distribution using a quantum example oracle. *SIAM J. on Computing*, 28(3):1136–1153, 1999.

[10] N. Bshouty, J. Jackson, and C. Tamon. More efficient PAC learning of DNF with membership queries under the uniform distribution. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, pages 286–295, 1999.

[11] N. Bshouty and C. Tamon. On the Fourier spectrum of monotone functions. *Journal of the ACM*, 43(4):747–770, 1996.

[12] A. Ehrenfeucht and D. Haussler. Learning decision trees from random examples. *Information and Computation*, 82(3):231–246, 1989.

[13] Y. Freund. Boosting a weak learning algorithm by majority. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 202–216, 1990.

[14] D. Gamarnik. Extension of the PAC framework to finite and countable Markov chains. In *Proceedings of the 12th Annual Conference on Computational Learning Theory*, pages 308–317, 1999.

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

[16] T. Hancock. *The complexity of learning formulas and decision trees that have restricted reads.* PhD thesis, Harvard University, 1992.

[17] T. Hancock and Y. Mansour. Learning monotone $k$-$\mu$ DNF formulas on product distributions. In *Proceedings of the Fourth Annual Conference on Computational Learning Theory*, pages 179–193, 1991.

[18] J. Hastad, R. Impagliazzo, L. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[19] J. Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. *Journal of Computer and System Sciences*, 55:414–440, 1997.

[20] J. Jackson, A. Klivans, and R. Servedio. Learnability beyond $\mathrm{AC}^0$. In *Proceedings of the 34th ACM Symposium on Theory of Computing*, 2002.

[21] R. Khardon. On using the Fourier transform to learn disjoint DNF. *Information Processing Letters*, 49:219–222, 1994.

[22] M. Kharitonov. Cryptographic hardness of distribution-specific learning. In *Proceedings of the Twenty-Fifth Annual Symposium on Theory of Computing*, pages 372–381, 1993.

[23] A. Klivans and R. Servedio. Boosting and hard-core sets. In *Proceedings of the Fortieth Annual Symposium on Foundations of Computer Science*, pages 624–633, 1999.

[24] A. Klivans and R. Servedio. Learning DNF in time $2^{\tilde{O}(n^{1/3})}$. In *Proceedings of the Thirty-Third Annual Symposium on Theory of Computing*, pages 258–265, 2001.

[25] M. Krause and P. Pudlak. On the computational power of depth 2 circuits with threshold and modulo gates. In *Proceedings of the Twenty-Sixth Annual Symposium on Theory of Computing*, pages 48–57, 1994.

[26] E. Kushilevitz and Y. Mansour. Learning decision trees using the Fourier spectrum. *SIAM J. on Computing*, 22(6):1331–1348, 1993.

[27] N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, Fourier transform and learnability. *Journal of the ACM*, 40(3):607–620, 1993.

[28] Y. Mansour. An $O(n^{\log \log n})$ learning algorithm for DNF under the uniform distribution. *Journal of Computer and System Sciences*, 50:543–550, 1995.

[29] R. O'Donnell. Hardness amplification within NP. In *Proceedings of the 34th ACM Symposium on Theory of Computing*, 2002.

[30] Y. Sakai and A. Maruoka. Learning monotone log-term DNF formulas under the uniform distribution. *Theory of Computing Systems*, 33:17–33, 2000.

[31] R. Servedio. On PAC learning using Winnow, Perceptron, and a Perceptron-like algorithm. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, pages 296–307, 1999.

[32] R. Servedio. On learning monotone DNF under product distributions. In *Proceedings of the Fourteenth Annual Conference on Computational Learning Theory*, pages 473–489, 2001.

[33] L. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[34] K. Verbeurgt. Learning DNF under the uniform distribution in quasi-polynomial time. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 314–326, 1990.

[35] K. Verbeurgt. Learning sub-classes of monotone DNF on the uniform distribution. In *Proceedings of the Ninth Conference on Algorithmic Learning Theory*, pages 385–399, 1998.

# A Proof of Proposition 2

**Proof:** It is well known that the existence of one-way functions implies the existence of pseudorandom function families [18]. Let $\{f_s\colon \{+1,-1\}^n \to \{+1,-1\}\}_{s\in\{+1,-1\}^n}$ be any pseudorandom function family. For $s \in \{+1,-1\}^n$ let $g_s\colon \{+1,-1\}^n \to \{+1,-1\}$ be defined by:

$$c_s(x) = \begin{cases} s_i & \text{if } x = e_i \text{ for some } i \in [n], \\ f_s(x) & \text{otherwise.} \end{cases}$$

(Here $e_i$ denotes the string $(-1,\ldots,-1,+1,-1,\ldots,-1)$, with the $+1$ in the $i$th position.) We will show that the concept class $\mathcal{C} = \{g_s\}_{s\in\{+1,-1\}^n}$ has the desired properties.

It is easy to see that any $g_s \in \mathcal{C}$ can be learned exactly in polynomial time if membership queries are allowed. The algorithm simply queries $e_1,\ldots,e_n$ to learn all bits $s_1,\ldots,s_n$ of $s$ and outputs a representation of $g_s$. On the other hand, a random walk which proceeds for only poly$(n)$ steps will with probability $1 - 2^{-\Omega(n)}$ miss all the points $e_i$. A straightforward argument shows that conditioned on missing all these points, it is impossible to learn $g_s$ in polynomial time. (To see this, note that an algorithm which has oracle access to a pseudorandom function $f_s$ can easily simulate a random walk which misses all $e_i$. Thus if it were possible to learn $g_s$ in polynomial time from a random walk conditioned on missing all $e_i$, it would be possible to learn the class $\{f_s\}$ given oracle access to $f_s$. But this is easily seen to contradict the definition of a pseudorandom function family.) $\square$

# B Proof of Claim 13

We suppose that the target function is selected uniformly at random from the set of all $2^s$ Boolean functions which depend only on bits $x_1,\ldots,x_{\log s}$. (Note that each such function has a DNF of size $s$ and a decision tree of size $s$). We will show that with very high probability a random walk of fewer than $\frac{sn}{24\log s}$ steps will realize at most $s/4$ of the $s$ possible settings for the first $\log s$ variables. Since the target function is randomly selected as described, any hypothesis has expected error (over the choice of the random target) exactly $1/2$ on all unseen settings. Thus conditioned on at most $s/4$ of the settings having been seen, with very high probability the hypothesis has error at least $1/3$ on the unseen settings (which have probability weight at least $3/4$), so the overall error rate is at least $1/4$.

Thus it suffices to prove the following fact: a random walk of $\frac{sn}{24\log s}$ steps on $\{0,1\}^n$ will with probability at least .99 realize at most $s/4$ settings of the first $\log s$ bits. But this is easily seen: the expected number of times that such a walk flips one of the first $\log s$ bits is is $s/24$, so a standard Chernoff bound implies that such a walk flips at most $s/4$ bits with probability at most $2^{-s/4} < 0.01$.