Approximating the influence of a monotone Boolean function in $O(\sqrt{n})$ query complexity

Dana Ron[*] Ronitt Rubinfeld [†] Muli Safra[‡] Omri Weinstein [§]

January 28, 2011

Abstract

The *Total Influence* (*Average Sensitivity*) of a discrete function is one of its fundamental measures. We study the problem of approximating the total influence of a monotone Boolean function $f : \{0,1\}^n \to \{0,1\}$, which we denote by $I[f]$. We present a randomized algorithm that approximates the influence of such functions to within a multiplicative factor of $(1 \pm \epsilon)$ by performing $O\left(\frac{\sqrt{n}\log n}{I[f]}\mathrm{poly}(1/\epsilon)\right)$ queries. We also prove a lower bound of $\Omega\left(\frac{\sqrt{n}}{\log n \cdot I[f]}\right)$ on the query complexity of any constant-factor approximation algorithm for this problem (which holds for $I[f] = \Omega(1)$), hence showing that our algorithm is almost optimal in terms of its dependence on n. For general functions we give a lower bound of $\Omega\left(\frac{n}{I[f]}\right)$, which matches the complexity of a simple sampling algorithm.

Keywords: influence of a Boolean function, sublinear approximation algorithms, random walks.

[*]School of Electrical Engineering at Tel Aviv University, `danar@eng.tau.ac.il`. This work was supported by the Israel Science Foundation (grant number 246/08).

[†]CSAIL at MIT, and the Blavatnik School of Computer Science at Tel Aviv University, `ronitt@csail.mit.edu`. This work was supported by NSF grants 0732334 and 0728645, Marie Curie Reintegration grant PIRG03-GA-2008-231077 and the Israel Science Foundation grant nos. 1147/09 and 1675/09

[‡]Blavatnik School of Computer Science at Tel Aviv University, `safra@post.tau.ac.il`.

[§]Computer Science Department, Princeton University, `oweinste@cs.princeton.edu`.

1 Introduction

The influence of a function, first introduced by Ben-Or and Linial [2] in the context of "collective coin-flipping", captures the notion of the sensitivity of a multivariate function. More precisely, for a Boolean function $f : \{0,1\}^n \to \{0,1\}$, the *individual influence* of coordinate i on f is defined as $I_i[f] \stackrel{\rm def}{=} \Pr_{x\in\{0,1\}^n}[f(x) \neq f(x^{(\oplus i)})]$, where x is selected uniformly[1] in $\{0,1\}^n$ and $x^{(\oplus i)}$ denotes x with the $i^{\rm th}$ bit flipped. The *total influence* of a Boolean function f (which we simply refer to as *the influence* of f) is $I[f] = \sum_i I_i[f]$.

The study of the influence of a function and its individual influences (distribution) has been the focus of many papers ([2, 21, 7, 16, 34, 8, 35, 14, 6, 15, 28, 11] to mention a few – for a survey see [17]). The influence of functions has played a central role in several areas of computer science. In particular, this is true for distributed computing (e.g., [2, 21]), hardness of approximation (e.g., [12, 22]), learning theory (e.g., [18, 9, 29, 30, 10])[2] and property testing (e.g., [13, 4, 5, 26, 31]). The notion of influence also arises naturally in the context of probability theory (e.g., [32, 33, 3]), game theory (e.g., [24]), reliability theory (e.g., [23]), as well as theoretical economics and political science (e.g., [1, 19, 20]).

Given that the influence is such a basic measure of functions and it plays an important role in many areas, we believe it is of interest to study the algorithmic question of approximating the influence of a function as efficiently as possible, that is by querying the function on as few inputs as possible. Specifically, the need for an efficient approximation for a function's influence might arise in the design of sublinear algorithms, and in particular property testing algorithms.

As we show, one cannot improve on a standard sampling argument for the problem of estimating the influence of a general Boolean function, which requires $\Omega(\frac{n}{I[f]})$ queries to the function, for any constant multiplicative estimation factor.[3] This fact justifies the study of subclasses of Boolean functions, among which the family of monotone functions is a very natural and central one. Indeed, we show that the special structure of monotone functions implies a useful behavior of their influence, and thus the computational problem of approximating the influence of such functions becomes significantly easier.

1.1 Our results and techniques

We present a randomized algorithm that approximates the influence of a monotone Boolean function to within any multiplicative factor of $(1 \pm \epsilon)$ in $O\left(\frac{\sqrt{n}\log n}{I[f]}\mathrm{poly}(1/\epsilon)\right)$ expected query complexity. We also prove an almost matching lower bound of $\Omega\left(\frac{\sqrt{n}}{\log n \cdot I[f]}\right)$ on the query complexity of any constant-factor approximation algorithm for this problem (which holds for $I[f] = \Omega(1)$).

As noted above, the influence of a function can be approximated by sampling random edges (i.e., pairs $(x, x^{(\oplus i)})$ that differ on a single coordinate) from the $\{0,1\}^n$ lattice. A random edge has probability $\frac{I[f]}{n}$ to be influential (i.e, satisfy $f(x) \neq f(x^{(\oplus i)})$), so a standard sampling argument implies that it suffices to ask $O(\frac{n}{I[f]}\mathrm{poly}(1/\epsilon))$ queries in order to approximate this probability to within $(1 \pm \epsilon)$.[4]

[1]The influence can be defined with respect to other probability spaces (as well as for non-Boolean functions), but we focus on the above definition.

[2]Here we referenced several works in which the influence appears explicitly. The influence of variables plays an implicit role in many learning algorithms, and in particular those that build on Fourier analysis, beginning with [25].

[3] If one wants an *additive* error of ϵ, then $\Omega((n/\epsilon)^2)$ queries are necessary (when the influence is large) [27].

[4]We also note that in the case of monotone functions, the total influence equals twice the sum of the Fourier coefficients

In order to achieve better query complexity, we would like to increase the probability of hitting an influential edge in a single trial. The algorithm we present captures this intuition, by taking random walks down the $\{0,1\}^n$ lattice[5], and then averaging the total number of influential edges encountered in all walks over the number of walks taken. The crucial observation on which the algorithm relies, is that a monotone function can have at most one influential edge in a single path, and thus it is sufficient to query only the start and end points of the walk to determine whether any influential edge was traversed.

Before continuing the technical discussion concerning the algorithm and its analysis, we make the following more conceptual note. Random walks have numerous applications in Computer Science as they are an important tool for mixing and sampling almost uniformly. In our context, where the walk is performed on the domain of an unknown function, it is used for a different purpose. Namely, by querying only the two endpoints of a random walk (starting from a uniformly sampled element) we (roughly) simulate the process of taking a much larger sample of elements.

The main issue that remains is determining the length of the walk, which we denote by w. Let $p_w(f)$ denote the probability that a walk of length w (down the lattice and from a uniformly selected starting point) passes through some influential edge.[6] We are interested in analyzing how $p_w(f)$ increases as a function of w. We show that for w that is $O(\epsilon\sqrt{n/\log n})$, the value of $p_w(f)$ increases almost linearly with w. Namely, it is $(1 \pm \epsilon) \cdot \frac{w}{n} \cdot I[f]$. Thus, by taking w to be $\Theta(\epsilon\sqrt{n/\log n})$ we get an improvement by a factor of roughly $\sqrt{n}$ on the basic sampling algorithm. We note though that by taking w to be larger we cannot ensure in general the same behavior of $p_w(f)$ as a function of w and $I[f]$, since the behavior might vary significantly depending on f.

The way we prove the aforementioned dependence of $p_w(f)$ on w is roughly as follows. For any edge e in the Boolean lattice, let $p_w(e)$ denote the probability that a walk of length w (as defined above) passes through e. By the observation made previously, that a monotone function can have at most one influential edge in a given path, $p_w(f)$ is the sum of $p_w(e)$, taken over all edges e that are influential with respect to f. For our purposes it is important that $p_w(e)$ be roughly the same for almost all edges. Otherwise, different functions that have the same number of influential edges, and hence the same influence $I[f]$, but whose influential edges are distributed differently in the Boolean lattice, would give different values for $p_w(f)$. We show that for $w = O(\epsilon\sqrt{n/\log n})$, the value of $p_w(e)$ increases almost linearly with w for all but a negligible fraction of the influential edges (where 'negligible' is with respect to $I[f]$). This implies that $p_w(f)$ grows roughly linearly in w for $w = O(\epsilon\sqrt{n/\log n})$.

To demonstrate the benefit of taking walks of length $O(\sqrt{n})$, let us consider the classic example of the *Majority* function on n variables. Here, all influential edges are concentrated in the exact middle levels of the lattice (i.e, all of them are of the form $(x, x^{(\oplus i)})$ where the Hamming weight of x is $\lfloor n/2 \rfloor$ or $\lceil n/2 \rceil$). The probability, $p_w(e)$, of a walk of length w passing through an influential edge e is simply the probability of starting the walk at distance at most w above the threshold $n/2$. Thus, taking longer walks allows us, so to speak, to start our walk from a higher point in the lattice, and still hit an influential edge. Since the probability of a uniformly chosen point to fall in each one of the the first $\sqrt{n}$ levels above the middle is roughly the same, the probability of hitting an influential edge in that case indeed grows roughly linearly

that correspond to singleton sets $\{i\}$, $i \in \{1, \ldots, n\}$. Therefore, it is possible to approximate the influence of a function by approximating this sum, which equals $\frac{1}{2^n} \cdot \sum_{i=1}^{n} \left(\sum_{x \in \{0,1\}^n : x_i = 1} f(x) - \sum_{x \in \{0,1\}^n : x_i = 0} f(x) \right)$. However, the direct sampling approach for such an approximation again requires $\Omega(n/I[f])$ samples.

[5]That is, starting from a randomly selected point in $\{0,1\}^n$, at each step, if the current point is x, we uniformly select an index i such that $x_i = 1$ and continue the walk to $x^{(\oplus i)}$.

[6]For technical reasons we actually consider a slightly different measure than $p_w(f)$, but we ignore this technicality in the introduction.

in the size of the walk. Nevertheless, taking walks of length which significantly exceeds $O(\sqrt{n})$ (say, even $\Omega(\sqrt{n \cdot \log(n)})$) would add negligible contribution to that probability (as this contribution is equivalent to the probability of a uniformly chosen point to deviate $\Omega(\sqrt{n \cdot \log(n)})$ levels from the middle level) and thus the linear dependence on the length of the walk is no longer preserved.

2 Preliminaries

In the introduction we defined the influence of a function as the sum, over all its variables, of their individual influence. An equivalent definition is that the influence of a function f is the expected number of sensitive coordinates for a random input $x \in \{0,1\}^n$ (that is, those coordinates i for which $f(x) \neq f(x^{(\oplus i)})$).

It will occasionally be convenient to view f as a 2-coloring of the Boolean lattice. Under this setting, any "bi-chromatic" edge, i.e, an edge $(x, x^{(\oplus i)})$ such that $f(x) \neq f(x^{(\oplus i)})$, will be called an *influential edge*. The number of influential edges of a Boolean function f is $2^{n-1} \cdot I[f]$.[7]

We consider the standard partial order '$\prec$' over the (n-dimensional) Boolean lattice. Namely, for $x = (x_1, ..., x_n), y = (y_1, ..., y_n) \in \{0,1\}^n$, we use the notation $x \prec y$ to mean that $x_i \leq y_i$ for every $1 \leq i \leq n$, and $x_i < y_i$ for some $1 \leq i \leq n$. A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is said to be *monotone* if $f(x) \leq f(y)$ for all $x \prec y$. A well known isoperimetric inequality implies that any monotone Boolean function satisfies $I[f] = O(\sqrt{n})$ (see [16] for a proof). This bound is tight for the notable *Majority* function.

In this paper we deal mainly with monotone Boolean functions that have at least constant Influence (i.e, $I[f] \geq c$, for some $c \geq 0$), since the computational problem we study arises more naturally when the function has some significant sensitivity. As shown in [21], the influence of a function is lower bounded by $4 \cdot \Pr[f = 1] \cdot \Pr[f = 0]$, and so our analysis holds in particular for functions that are not too biased (relatively balanced).

Notations. We use the notation $f(n) = \tilde{O}(g(n))$ if $f(n) = O(g(n)\text{polylog}(g(n)))$. Similarly, $f(n) = \tilde{\Omega}(g(n))$ if $f(n) = \Omega(g(n)/\text{polylog}(g(n)))$.

3 The Algorithm

As noted in the introduction, we can easily get a $(1 \pm \epsilon)$-factor estimate of the influence with high constant probability by uniformly sampling $\Theta\left(\frac{n}{I[f]} \cdot \epsilon^{-2}\right)$ pairs $(x, x^{(\oplus i)})$ (edges in the Boolean lattice), querying the function on these pairs, and considering the fraction of influential edges observed in the sample. We refer to this as the *direct sampling approach*. However, since we are interested in an algorithm whose complexity is $\frac{\sqrt{n}}{I[f]} \cdot \text{poly}(1/\epsilon)$ we take a different approach. To be precise, the algorithm we describe works for ϵ that is above a certain threshold (of the order of $\sqrt{\log n/n}$). However, if ϵ is smaller, then $\frac{n}{I[f]} \cdot \epsilon^{-2}$ is upper bounded by $\frac{\sqrt{n}}{I[f]} \cdot \text{poly}(1/\epsilon)$, and we can take the direct sampling approach. Thus we assume from this point on that $\epsilon = \omega(\sqrt{\log n/n})$.

As discussed in the introduction, instead of considering neighboring pairs, $(x, x^{(\oplus i)})$, we consider pairs (v, u) such that $v \succ u$ and there is a path down the lattice of length roughly $\epsilon\sqrt{n}$ between v and u. Observe that since the function f is monotone, if the path (down the lattice) from v to u contains an influential edge,

[7]To verify this, observe that when partitioning the Boolean lattice into two sets with respect to a coordinate i, we end up with 2^{n-1} vertices in each set. The individual influence of variable i, $I_i[f]$, is the fraction of the "bi-chromatic" edges among all edges crossing the cut. Since $I[f] = \sum_{i=1}^{n} I_i[f]$ we get that the total number of influential edges is $2^{n-1} \cdot I[f]$.

then $f(v) \neq f(u)$, and furthermore, any such path can contain at most one influential edge. The intuition is that since we "can't afford" to detect influential edges directly, we raise our probability of detecting edges by considering longer paths.

In our analysis we show that this intuition can be formalized so as to establish the correctness of the algorithm. We stress that when considering a path, the algorithm only queries its endpoints, so that it "doesn't pay" for the length of the path. The precise details of the algorithm are given in Figure 1. When we say that we take a walk of a certain length w down the Boolean lattice *with a cut-off* at a certain level ℓ, we mean that we stop the walk (before taking all w steps) if we reach a point in level ℓ (i.e., with Hamming weight ℓ).

Note that m, the number of walks taken, is a random variable. Namely, the algorithm continues taking new walks until the number of "successful" walks (that is, walks that pass through an influential edge) reaches a certain threshold, which is denoted by t. The reason for doing this, rather than deterministically setting the number of walks and considering the random variable which is the number of successful walks, is that the latter approach requires to know a lower bound on the influence of f. While it is possible to search for such a lower bound (by working iteratively in phases and decreasing the lower bound on the influence between phases) our approach yields a somewhat simpler algorithm.

```
Algorithm 1: Approximating the Influence (given $\epsilon, \delta$ and oracle access to $f$)

  1. Set $\tilde{\epsilon} = \epsilon/4$, $w = \frac{\tilde{\epsilon}\sqrt{n}}{16\sqrt{2\log(\frac{2n}{\tilde{\epsilon}})}}$, $s^* = \frac{1}{2}\sqrt{2n\log(\frac{2n}{\tilde{\epsilon}})}$, and $t = \frac{96\ln(\frac{2}{\delta})}{\epsilon^2}$.
  2. Initialize $\alpha \leftarrow 0$, $m \leftarrow 0$, and $\hat{I} \leftarrow 0$.
  3. Repeat the following until $\alpha = t$:
     (a) Perform a random walk of length $w$ down the $\{0,1\}^n$ lattice from a uniformly chosen point
         $v$ with a cut-off at $n/2 - s^* - 1$, and let $u$ denote the endpoint of the walk.
     (b) If $f(u) \neq f(v)$ then $\alpha \longleftarrow \alpha + 1$.
     (c) $m \leftarrow m + 1$
  4. $\hat{I} \leftarrow \frac{n}{w} \cdot \frac{t}{m}$
  5. Return $\hat{I}$.
```

Figure 1: The algorithm for approximating the influence of a function f.

In what follows we assume for simplicity that $I[f] \geq 1$. As we discuss subsequently, this assumption can be easily replaced by $I[f] \geq c$ for any constant $c > 0$, or even $I[f] \geq n^{-c}$, by performing a slight modification in the setting of the parameters of the algorithm.

Theorem 3.1 *For every monotone function $f : \{0,1\}^n \to \{0,1\}$ such that $I[f] \geq 1$, and for every $\delta > 0$ and $\epsilon = \omega(\sqrt{\log n/n})$, with probability at least $1 - \delta$, the output, $\hat{I}$, of Algorithm 1 satisfies:*

$$(1-\epsilon) \cdot I[f] \leq \hat{I} \leq (1+\epsilon) \cdot I[f] \; .$$

Furthermore, with probability at least $1 - \delta$, the number of queries performed by the algorithm is $O\left(\frac{\log(1/\delta)}{\epsilon^3} \cdot \frac{\sqrt{n}\log(n/\epsilon)}{I[f]}\right)$.

We note that the (probabilistic) bound on the number of queries performed by the algorithm implies that the expected query complexity of the algorithm is $O\left(\frac{\log(1/\delta)}{\epsilon^3} \cdot \frac{\sqrt{n}\log(n/\epsilon)}{I[f]}\right)$. Furthermore, the probability that the algorithm performs a number of queries that is more than k times the expected value decreases exponentially with k.

The next definition is central to our analysis.

Definition 1 *For a (monotone) Boolean function f and integers w and s^*, let $p_{w,s^*}(f)$ denote the probability that a random walk of length w down the Boolean lattice, from a uniformly selected point and with a cut-off at $n/2 - s^* - 1$, starts from $f(v) = 1$ and reaches $f(u) = 0$.*

Given the definition of $p_{w,s^*}(f)$, we next state and prove the main lemma on which the proof of Theorem 3.1 is based.

Lemma 3.2 *Let f satisfy $I[f] \geq 1$, let $\epsilon > 0$ satisfy $\epsilon > \frac{8\sqrt{2\log(\frac{8n}{\epsilon})}}{\sqrt{n}}$, and denote $\tilde{\epsilon} = \epsilon/4$. For any $w \leq \frac{\tilde{\epsilon}\sqrt{n}}{16\sqrt{2\log(\frac{2n}{\tilde{\epsilon}I[f]})}}$ and for $s^* = \frac{1}{2}\sqrt{n} \cdot \sqrt{2\log(\frac{2n}{\tilde{\epsilon}})}$ we have that*

$$(1-\epsilon/2) \cdot \frac{w}{n} \cdot I[f] \;\leq\; p_{w,s^*}(f) \;\leq\; (1+\epsilon/2) \cdot \frac{w}{n} \cdot I[f]\;.$$

Proof: For a point $y \in \{0,1\}^n$, let $h(y)$ denote its Hamming weight (which we also refer to as the *level* in the Boolean lattice that it belongs to). By the choice of $s^* = \frac{1}{2}\sqrt{n}\sqrt{2\log(\frac{2n}{\tilde{\epsilon}})}$, and since $I[f] \geq 1$, the number of points y for which $h(y) \geq n/2 + s^*$ or $h(y) \leq n/2 - s^*$, is upper bounded by $2^n \cdot \frac{\tilde{\epsilon}I[f]}{n}$. Each such point y is incident to n edges, and each edge has two endpoints. It follows that there are at most $2^{n-1} \cdot \tilde{\epsilon}I[f]$ edges (y,x) for which $h(y) \geq n/2 + s^*$ or $h(y) \leq n/2 - s^*$. Recall that an influential edge (y,x) for $h(y) = h(x)+1$, is an edge that satisfies $f(y) = 1$ and $f(x) = 0$. Let $e_{s^*}(f)$ denote the number of influential edges (y,x) such that $n/2 - s^* \leq h(x), h(y) \leq n/2 + s^*$. Since the total number of influential edges is $2^{n-1}I[f]$, we have that

$$(1-\tilde{\epsilon}) \cdot 2^{n-1}I[f] \;\leq\; e_{s^*}(f) \;\leq\; 2^{n-1}I[f]\;. \tag{1}$$

Consider any influential edge (y,x) where $h(y) = \ell$ and $\ell \geq n/2 - s^*$. We are interested in obtaining bounds on the probability that a random walk of length w (where $w \leq \frac{\tilde{\epsilon}\sqrt{n}}{16\sqrt{2\log(\frac{2n}{\tilde{\epsilon}I[f]})}}$) down the lattice, starting from a uniformly selected point $v \in \{0,1\}^n$, and with a cut-off at $n/2 - s^* - 1$, passes through (y,x). First, there is the event that $v = y$ and the edge (y,x) was selected in the first step of the walk. This event occurs with probability $2^{-n} \cdot \frac{1}{\ell}$. Next there is the event that v is at distance 1 from y (and above it, that is, $h(v) = h(y) + 1 = \ell + 1$), and the edges (v,y) and (y,x) are selected. This occurs with probability $2^{-n} \cdot (n-\ell) \cdot \frac{1}{\ell+1} \cdot \frac{1}{\ell}$. In general, for every $1 \leq i \leq w-1$ we have $(n-\ell)\cdots(n-\ell-i+1)$ pairs (v,P) where $v \succ y$ and $w(v) = \ell + i$, and where P is a path down the lattice from v to y. The probability of selecting v as the starting vertex is 2^{-n} and the probability of taking the path P from v is $\frac{1}{(\ell+i)\cdots(\ell+1)}$. Therefore, the probability that the random walk passes through (y,x) is:

$$2^{-n} \cdot \frac{1}{\ell} \cdot \left(1 + \sum_{i=1}^{w-1} \frac{(n-\ell)\cdots(n-\ell-i+1)}{(\ell+i)\cdots(\ell+1)}\right) = 2^{-n} \cdot \frac{1}{\ell}\left(1 + \sum_{i=1}^{w-1}\prod_{j=0}^{i-1} \frac{n-\ell-j}{\ell+i-j}\right)\;. \tag{2}$$

Let $\ell = n/2 + s$ (where s may be negative), and denote $\tau(\ell, i, j) \overset{\text{def}}{=} \frac{n-\ell-j}{\ell+i-j}$. Then

$$\tau(\ell, i, j) = \frac{n/2 - s - j}{n/2 + s + i - j} = 1 - \frac{2s + i}{n/2 + s + i - j} \ . \tag{3}$$

Consider first the case that $\ell \geq n/2$, i.e $\ell = n/2 + s$ ($s \geq 0$). In that case it is clear that $\tau(\ell, i, j) \leq 1$ (since $j \leq i$), so $\prod_{j=0}^{i-1} \tau(\ell, i, j)$ is upper bounded by 1. In order to lower bound $\prod_{j=0}^{i-1} \tau(\ell, i, j)$, we note that

$$\tau(\ell, i, j) \geq 1 - \frac{2s + w}{n/2} = 1 - \frac{2(2s + w)}{n} \ . \tag{4}$$

Thus, for $s \leq s^*$ we have

$$\begin{aligned}
\prod_{j=0}^{i-1} \tau(\ell, i, j) &\geq \prod_{j=0}^{i-1} \left(1 - \frac{2(2s+w)}{n}\right) \\
&\geq \left(1 - \frac{2(2s+w)}{n}\right)^w && \text{(since } i \leq w\text{)} \\
&\geq 1 - \frac{2(2s+w)w}{n} \\
&\geq 1 - \frac{6s^*w}{n} && (2s + w \geq 3s^* \text{ since } s \leq s^* \text{ and } w \leq s^*) \\
&= 1 - \frac{3\tilde{\epsilon}}{16} && \text{(by the definitions of } s^* \text{ and } w\text{)} \\
&\geq 1 - \tilde{\epsilon}/2 \ .
\end{aligned} \tag{5}$$

Therefore, we have that for $n/2 \leq \ell \leq n/2 + s^*$,

$$1 - \tilde{\epsilon}/2 \ \leq \ \prod_{j=0}^{i-1} \frac{n - \ell - j}{\ell + i - j} \ \leq \ 1 \ , \tag{6}$$

and for $\ell > n/2 + s^*$ it holds that

$$\prod_{j=0}^{i-1} \frac{n - \ell - j}{\ell + i - j} \leq 1 \ . \tag{7}$$

We turn to the case where $n/2 - s^* \leq \ell < n/2$. Here we have

$$\tau(\ell, i, j) \ = \ 1 + \frac{2s - i}{n/2 - s + i - j} \ \geq \ 1 - \frac{2w}{n - 2w} \ \geq \ 1 - \frac{4w}{n} \tag{8}$$

where the last inequality follows from the fact that $w < n/4$. Thus,

$$\prod_{j=0}^{i-1} \tau(\ell, i, j) \ \geq \ \left(1 - \frac{4w}{n}\right)^w \ \geq \ 1 - \frac{4w^2}{n} \ = \ 1 - \frac{4}{n} \cdot \left(\frac{\tilde{\epsilon}\sqrt{n}}{16\sqrt{2\log(\frac{2n}{\tilde{\epsilon}I[f]})}}\right)^2 \ > \ 1 - \tilde{\epsilon}^2/2 \ > \ 1 - \tilde{\epsilon}/2 \ . \tag{9}$$

On the other hand,

$$\tau(\ell,i,j) \;=\; 1+\frac{2s-i}{n/2-s+i-j} \;\le\; 1+\frac{2s}{n/2-s} \;\le\; 1+\frac{8s^*}{n}\;, \tag{10}$$

where the last inequality holds since $n \ge 2s$. Thus, we have

$$\prod_{j=0}^{i-1}\tau(\ell,i,j) \;\le\; \left(1+\frac{8s^*}{n}\right)^{w} \;\le\; 1+\frac{16s^*w}{n} \;=\; 1+\tilde{\epsilon}/2\;. \tag{11}$$

where the second inequality follows from the inequality $(1+\alpha)^k \le 1+2\alpha k$ which holds for $\alpha < 1/(2k)$; Indeed, in our case $8s^*/n \le 1/(2w)$ (this is equivalent to $w \le n/16s^*$ which holds given our setting of s^* and the upper bound on w).
We therefore have that for $n/2-s^* \le \ell < n/2$,

$$1-\tilde{\epsilon}/2 \;\le\; \prod_{j=0}^{i-1}\frac{n-\ell-j}{\ell+i-j} \;\le\; 1+\tilde{\epsilon}/2\;. \tag{12}$$

Combining Equations (6) and (12), we have that for $n/2-s^* \le \ell \le n/2+s^*$,

$$1-\tilde{\epsilon}/2 \;\le\; \prod_{j=0}^{i-1}\frac{n-\ell-j}{\ell+i-j} \;\le\; 1+\tilde{\epsilon}/2\;. \tag{13}$$

Now, we are interested in summing up the probability, over all random walks, that the walk passes through an influential edge. Since the function is monotone, every random walk passes through at most one influential edge, so the sets of random walks that correspond to different influential edges are disjoint (that is, the event that a walk passes through an influential edge (y,x) is disjoint from the event that it passes through another influential edge (y',x')). Since the edges that contribute to $p_{w,s^*}(f)$ are all from levels $\ell \ge n/2-s^*$ (and since there are $2^{n-1}I[f]$ influential edges in total), by Equations (2), (7) and (13) we have

$$\begin{aligned}
p_{w,s^*}(f) &\le 2^{n-1}I[f]2^{-n}\cdot\frac{1}{n/2-s^*}\left(1+\sum_{i=1}^{w-1}(1+\tilde{\epsilon}/2)\right) && (14)\\
&\le \frac{1}{2}I[f]\cdot\frac{1}{n/2-s^*}\cdot w(1+\tilde{\epsilon}/2) && (15)\\
&\le \frac{1}{2}I[f]\cdot\frac{2}{n}(1+\tilde{\epsilon})\cdot w(1+\tilde{\epsilon}/2) && (16)\\
&\le \frac{I[f]\cdot w}{n}\cdot(1+2\tilde{\epsilon}) && (17)\\
&= \frac{I[f]\cdot w}{n}(1+\epsilon/2)\;, && (18)
\end{aligned}$$

where Equation (16) follows from the definition of s^*, the premise of the lemma that $\epsilon > \frac{8\sqrt{2\log(\frac{8n}{\epsilon})}}{\sqrt{n}}$ and $\tilde{\epsilon} = \epsilon/4$.

For lower bounding $p_{w,s^*}(f)$, we will consider only the contribution of the influential edges that belong to levels $\ell \le n/2 + s^*$. Consequently, Equations (1), (2) and (13) give in total

$$\begin{aligned} p_{w,s^*}(f) &\ge 2^{n-1}(1-\tilde{\epsilon})I[f]2^{-n} \cdot \frac{1}{n/2+s^*}\left(1+\sum_{i=1}^{w-1}(1-\tilde{\epsilon}/2)\right) && (19)\\ &\ge \frac{1}{2}I[f](1-\tilde{\epsilon})w(1-\tilde{\epsilon/2}) \cdot \frac{1}{n/2+s^*} && (20)\\ &\ge \frac{1}{2}I[f]\cdot w(1-\tilde{\epsilon})(1-\tilde{\epsilon/2}) \cdot \frac{2}{n}(1-\tilde{\epsilon}) && (21)\\ &\ge \frac{I[f]\cdot w}{n}(1-2\tilde{\epsilon}) && (22)\\ &= \frac{I[f]\cdot w}{n}(1-\epsilon/2)\ , && (23) \end{aligned}$$

where Equation (21) follows from the definition of s^*, the premise of the lemma that $\epsilon > \frac{8\sqrt{2\log(\frac{8n}{\epsilon})}}{\sqrt{n}}$ and $\tilde{\epsilon} = \epsilon/4$.

Equations (18) and (23) give

$$(1-\epsilon/2)\cdot\frac{w}{n}\cdot I[f] \le p_{w,s^*}(f) \le (1+\epsilon/2)\cdot\frac{w}{n}\cdot I[f]\ , \tag{24}$$

as claimed in the Lemma. ■

Proof of Theorem 3.1: For w and s^* as set by the algorithm, let $p_{w,s^*}(f)$ be as in Definition 1, where we shall use the shorthand $p(f)$. Recall that m is a random variable denoting the number of iterations performed by the algorithm until it stops (once $\alpha = t$). Let $\tilde{m} = \frac{t}{p(f)}$, $\tilde{m}_1 = \frac{\tilde{m}}{(1+\epsilon/4)}$, and $\tilde{m}_2 = \frac{\tilde{m}}{(1-\epsilon/4)}$. We say that an iteration of the algorithm is *successful* if the walk taken in that iteration passes through an influential edge (so that the value of α is increased by 1). Let $\hat{p}(f) = \frac{t}{m}$ denote the fraction of successful iterations.

Suppose that $\tilde{m}_1 \le m \le \tilde{m}_2$. In such a case,

$$(1-\epsilon/4)\cdot p(f) \ \le\ \hat{p}(f) \ \le\ (1+\epsilon/4)p(f) \tag{25}$$

since $\hat{p}(f) = \frac{t}{m} = \frac{p(f)\cdot\tilde{m}}{m}$. By the definition of the algorithm, $\hat{I} = \frac{n}{w}\cdot\frac{t}{M} = \frac{n}{w}\cdot\hat{p}(f)$ so by Lemma 3.2 (recall that by the premise of the theorem, $\epsilon = \omega(\sqrt{\log n/n})$) we have

$$(1-\epsilon)I[f] \ \le\ (1-\epsilon/2)(1-\epsilon/4)I[f] \ \le\ \hat{I} \ \le\ (1+\epsilon/4)(1+\epsilon/2)I[f] \ \le\ (1+\epsilon)I[f] \tag{26}$$

and thus (assuming $\tilde{m}_1 \le m \le \tilde{m}_2$), the output of the algorithm provides the estimation we are looking for.

It remains to prove that $\tilde{m}_1 \le m \le \tilde{m}_2$ with probability at least $1-\delta$. Let X_i denote the indicator random variable whose value is 1 if and only if the i^{th} iteration of the algorithm was successful, and let $X = \sum_{i=1}^{\tilde{m}_1} X_i$. By the definition of X_i, we have that $\mathrm{E}[X_i] = p(f)$, and so (by the definition of $\tilde{m}_1$ and $\tilde{m}$) we have that $\mathrm{E}[X] = \tilde{m}_1\cdot p(f) = \frac{t}{1+\epsilon/4}$ Hence, by applying the multiplicative Chernoff bound,

$$\Pr[m < \tilde{m}_1] = \Pr[X > t] = \Pr[X > (1+\epsilon/4)\mathrm{E}[X]] \le \exp\left(-\frac{1}{3}\left(\frac{\epsilon}{4}\right)^2\frac{t}{1+\epsilon/4}\right) \le \exp\left(-\frac{\epsilon^2 t}{96}\right) \tag{27}$$

Thus, for $t = \frac{96\ln(\frac{2}{\delta})}{\epsilon^2}$ we have that $\Pr[m < \tilde{m}_1] \le \frac{\delta}{2}$. By an analogous argument we get that $\Pr[m > \tilde{m}_2] \le \frac{\delta}{2}$, and so $\tilde{m}_1 \le m \le \tilde{m}_2$ with probability at least $1-\delta$, as desired.

Since we have shown that $m \le \tilde{m}_2$ with probability at least $1-\delta$, and the query complexity of the algorithm is $O(m)$, we have that, with probability at least $1-\delta$, the query complexity is upper bounded by

$$O(\tilde{m}_2) \;=\; O\left(\frac{t}{p(f)}\right) \;=\; O\left(\frac{t\cdot n}{w\cdot I[f]}\right) \;=\; O\left(\frac{\log(1/\delta)}{\epsilon^3}\cdot\frac{\sqrt{n}\log(n/\epsilon)}{I[f]}\right)\ , \tag{28}$$

as required. ■

Remark. We assumed that $I[f] \ge 1$ only for the sake of technical simplicity. This assumption can be replaced with $I[f] \ge \frac{1}{n^c}$ for any constant $c \ge 0$, and the only modifications needed in the algorithm and its analysis are the following. The level of the cutoff s^* should be set to $s^* = \sqrt{n/2}\cdot\sqrt{\log(\frac{2n}{\tilde{\epsilon}n^{-c}})} = \frac{1}{2}\sqrt{n}\sqrt{2c\log(2n)+\log(1/\tilde{\epsilon})}$ (which is a constant factor larger than the current setting), and the length w of the walks in the algorithm should be set to $w = \frac{\tilde{\epsilon}\sqrt{n}}{16\sqrt{2\log(\frac{2n}{\tilde{\epsilon}n^{-c}})}}$ (which is a constant factor smaller than the current setting).

The first modification follows from the fact that the number of points y whose Hamming weight $h(y)$ is at least $n/2 + r\cdot\sqrt{n/2}$ or at most $n/2 - r\cdot\sqrt{n/2}$ is upper bounded by $2^n \cdot 2e^{-r^2}$. This implies that the number of edges (y,x) (where $h(y) = h(x)+1$) such that $h(y) \ge n/2+r\cdot\sqrt{n/2}$ or $h(y) \le n/2-r\cdot\sqrt{n/2}$ is upper bounded by $n\cdot 2^n\cdot 2^{-r^2}$. Requiring that the latter is no more than $\tilde{\epsilon}\cdot I[f]2^{n-1} \ge \tilde{\epsilon}\cdot n^{-c}2^{n-1}$ (i.e, $\tilde{\epsilon}$-fraction of the total number of influential edges), yields the desired r, where $s^* = r\sqrt{n/2}$. The second modification, i.e, in the length of the walk, is governed by the choice of s^*, since, by the analysis, their product should be bounded by $O(\tilde{\epsilon}n)$. Since in both expressions $1/I[f] = n^c$ appears only inside a $\log$ term, this translates only to constant factor increase.

We note that the lower bound we give in Section 4 applies only to functions with (at least) constant influence, and so in the above case where $I[f] = 1/\mathrm{poly}(n)$, the tightness of the algorithm (in terms of query complexity) is not guaranteed.

4 A Lower Bound

In this section we prove a lower bound of $\Omega\left(\frac{\sqrt{n}}{I[f]\cdot\log n}\right)$ on the query complexity of approximating the influence of monotone functions. Following it we explain how a related construction gives a lower bound of $\Omega\left(\frac{n}{I[f]}\right)$ on approximating the influence of *general* functions. The idea for the first lower bound is the following. We show that any algorithm that performs $o\left(\frac{\sqrt{n}}{I[f]\cdot\log n}\right)$ queries cannot distinguish with constant success probability between that following: (1) A certain threshold function (over a relatively small number of variables), and (2) A function selected uniformly at random from a certain family of functions that have significantly higher influence than the threshold function. The functions in this family can be viewed as "hiding their influence behind the threshold function". More precise details follow.

We first introduce one more notation. For any integer $1 \le k \le n$ and $0 \le t \le k$, let $\tau_k^t : \{0,1\}^n \to \{0,1\}$ be the *t-threshold function* over $x_1,\ldots,x_k$. That is, $\tau_k^t(x) = 1$ if and only if $\sum_{i=1}^k x_i \ge t$. Observe that (since for every $1 \le i \le k$ we have that $I_i[\tau_k^t] = 2^{-k}\cdot 2\cdot\binom{k-1}{t-1}$ while for $i > k$ we have that $I_i[\tau_k^t] = 0$), $I[\tau_k^t] = k\cdot 2^{-(k-1)}\cdot\binom{k-1}{t-1}$.

The above observation implies that for every sufficiently large k ($k \geq 2\log n$ suffices), there exists a setting of $t < k/2$, which we denote by $t(k,1)$, such that $I[\tau_k^{t(k,1)}] = 1 - o(1)$ (where the $o(1)$ is with respect to k). This setting satisfies $\binom{k-1}{t(k,1)-1} = \Theta(2^k/k)$ (so that $t(k,1) = k/2 - \Theta(\sqrt{k\log k})$).

Theorem 4.1 *For every I^* such that $2 \leq I^* \leq \sqrt{n}/\log n$, there exists a family of monotone functions F_{I*} such that $I[f] \geq I^*$ for every $f \in F_{I^*}$, but any algorithm that distinguishes with probability at least $2/3$ between a uniformly selected function in F_{I^*} and $\tau_k^{t(k,1)}$ for $k = 2\log n$, must perform $\Omega\left(\frac{\sqrt{n}}{I^*\cdot\log n}\right)$ queries.*

In particular, considering $I^* = c$ for any constant $c \geq 2$, we get that every algorithm for approximating the influence to within a multiplicative factor of $\sqrt{c}$ must perform $\tilde{\Omega}(\sqrt{n})$ queries. If we increase the lower bound on the influence, then the lower bound on the complexity of the algorithm decreases, but the approximation factor (for which the lower bound holds), increases. We note that the functions for which the lower bound construction hold are not balanced, but we can easily make them very close to balanced without any substantial change in the argument (by "ORing" $\tau_k^{t(k,1)}$ as well as every function in F_{I^*} with x_1). We also note that for $I^* = \Omega(\sqrt{\log n})$ we can slightly improve the lower bound on approximating the influence to $\Omega\left(\frac{\sqrt{n}}{I^*\cdot\sqrt{\log(\sqrt{n}/I^*)}}\right)$ (for a slightly smaller approximation factor). We address this issue following the proof.

Proof: For $k = 2\log n$ and for any $0 \leq t \leq k$, let $L_k^t \overset{\text{def}}{=} \{x \in \{0,1\}^k \;:\; \sum_{i=1}^k x_i = t\}$. We shall also use the shorthand $\tilde{t}$ for $t(k,1)$. Fixing a choice of I^*, each function in F_{I^*} is defined by a subset R of $L_k^{\tilde{t}}$ where $|R| = \beta(I^*)\cdot 2^k$ for $\beta(I^*)$ that is set subsequently. We denote the corresponding function by f_R and define it as follows: For every $x \in \{0,1\}^n$, if $x_1 \ldots x_k \notin R$, then $f_R(x) = \tau_k^{\tilde{t}}(x)$, and if $x_1 \ldots x_k \in R$, then $f_R(x) = \text{maj}'_{n-k}(x)$, where $\text{maj}'_{n-k}(x) = 1$ if and only if $\sum_{i=k+1}^n x_i > (n-k)/2$. By this definition, for every $f_R \in F_{I^*}$

$$I[f_R] \geq \beta(I^*)\cdot I[\text{maj}'_{n-k}]\,. \tag{29}$$

If we take $\beta(I^*)$ to be $\beta(I^*) = I^*/I[\text{maj}'_{n-k}] = cI^*/\sqrt{n-k}$ (for c that is roughly $\sqrt{\pi/2}$), then in F_{I^*} every function has influence at least I^*. Since $\beta(I^*)$ is upper bounded by $|L_k^{\tilde{t}}|/2^k$, which is of the order of $k/2^k = 2\log n/2^k$, this construction is applicable to $I^* = O(\sqrt{n}/\log n)$.

Consider an algorithm that needs to distinguish between $\tau_k^{\tilde{t}}$ and a uniformly selected $f_R \in F_{I^*}$. Clearly, as long as the algorithm doesn't perform a query on x such that $x_1 \ldots x_k \in R$, the value returned by f_R is the same as that of $\tau_k^{\tilde{t}}$. But since R is selected uniformly in $L_k^{\tilde{t}}$, as long as the algorithm performs less than $\frac{|L_k^{\tilde{t}}|}{c'\cdot\beta(I^*)\cdot 2^k}$ queries (where c' is some sufficiently large constant), with high constant probability (over the choice of R), it won't "hit" a point in R. Since $\frac{|L_k^{\tilde{t}}|}{c'\cdot\beta(I^*)\cdot 2^k} = \Theta\left(\frac{\sqrt{n}}{\log n\cdot I^*}\right)$, the theorem follows. ■

In order to get the aforementioned slightly higher lower bound for $I^* = \Omega(\sqrt{\log n})$, we modify the settings in the proof of Theorem 4.1 in the following manner. We set $k = \log(\sqrt{n}/I^*)$ and $t = k/2$ (so that the "low influence" function is simply a majority function over k variables, $\tau_k^{k/2}$). For the "high influence" function, we let R consist of a single point $\tilde{x}$ in $L_k^{k/2}$, where for each $R = \{\tilde{x}\}$ we have a different function in F_{I^*} (as defined in the proof of Theorem 4.1). It follows that for each such R, $I[f_R] = (1-o(1))\sqrt{k} + \frac{1}{2^k}\sqrt{n-k} \geq I^*$, while $I[\tau_k^{k/2}] \approx \sqrt{k} = O(\sqrt{\log n})$. By the same argument as in the proof of Theorem 4.1, if the algorithm preforms less than $\frac{c'|L_k^{k/2}|}{|R|} = \frac{2^k}{c'\sqrt{k}} = \frac{\sqrt{n}}{c'I^*\sqrt{\log(\frac{\sqrt{n}}{I^*})}}$ queries (for small enough c'), with high probability it won't "hit" $\tilde{x}$, and thus will not be able to distinguish between a randomly selected function $f \in f_R$ (where the randomness is over the choice of $\tilde{x} \in L_k^{k/2}$) and $\tau_k^{k/2}$.

A lower bound of $\Omega(n/I[f])$ for general functions. We note that for general (not necessarily monotone) functions, there is a lower bound of $\Omega(n/I[f])$ on estimating the influence, which implies that it is not possible in general to improve on the simple edge-sampling approach (in terms of the dependence on n and $I[f]$). Similarly to what we showed in the case of monotone functions, we show that for every $I^* \geq 2$, it is hard to distinguish between the dictatorship function $f(x) = x_1$ (for which $I[f] = 1$) and a uniformly selected function in a family F_{I^*} of functions, where every function in F_{I*} has influence at least I^*.

Similarly to the construction in the proof of Theorem 4.1, we consider the first k variables, where here $k = \log n$. Fixing I^* (where $I^* = o(n)$ or else the lower bound is trivial), each function in F_{I^*} is defined by a subset R of $\{0,1\}^k$ such that $|R| = I^*$. We denote the corresponding function by f_R and define it as follows: For every $x \in \{0,1\}^n$, if $x_1 \ldots x_k \notin R$ then $f_R(x) = x_1$, and if $x_1 \ldots x_k \in R$, we let $f_R(x) = \bigoplus_{i=k+1}^n x_i$. By this definition (since $2^k = n$), for every $f_R \in F_{I^*}$ $I[f_R] \geq (1 - 2I^*/n) + (I^*/n) \cdot (n-k) \geq I^*$. The argument for establishing that it is hard to distinguish between $f(x) = x_i$ and a uniformly selected function in F_{I^*} is essentially the same as in the proof of Theorem 4.1.

References

[1] K. Arrow. A difficulty in the theory of social welfare. *Journal of political economics*, 58:328–346, 1950.

[2] M. Ben-Or and N. Linial. Collective coin flipping, robust voting schemes and minima of Banzhaf values. In *Proceedings of the Twenty-Sixth Annual Symposium on Foundations of Computer Science (FOCS)*, pages 408–416, 1985.

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

[4] E. Blais. Improved bounds for testing juntas. In *Proceedings of the Twelfth International Workshop on Randomization and Computation (RANDOM)*, pages 317–330, 2008.

[5] E. Blais. Testing juntas nearly optimally. In *Proceedings of the Fourty-First Annual ACM Symposium on the Theory of Computing (STOC)*, pages 151–158, 2009.

[6] R. Boppana. The average sensitivity of bounded depth circuits. *Information Processing Letters*, 63:257–261, 1997.

[7] J. Bourgain, J. Kahn, G. Kalai, Y. Katznelson, and N. Linial. The influence of variables in product spaces. *Israel Journal of Math*, 77:55–64, 1992.

[8] J. Bourgain and G. Kalai. Influences of variables and threshold intervals under group symmetries. *Geometric Functional Analaysis*, 7:438–461, 1997.

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

[10] I. Diakonikolas, P. Harsha, A. Klivans, R. Meka, P. Raghavendra, R. Servedio, and L.-Y. Tan. Bounding the average sensitivity and noise sensitivity of polynomial threshold functions. In *Proceedings of the Fourty-Second Annual ACM Symposium on the Theory of Computing (STOC)*, 2010. To appear.

[11] I. Dinur, E. Friedgut, G. Kindler, and R. O'Donnell. On the fourier tails of bounded functions over the discrete cube. To appear in *Israel Journal of Math*, 2010.

[12] I. Dinur and S. Safra. The importance of being biased. In *Proceedings of the Thirty-Fourth Annual ACM Symposium on the Theory of Computing (STOC)*, pages 33–42, 2002.

[13] E. Fischer, G. Kindler, D. Ron, S. Safra, and S. Samorodnitsky. Testing juntas. *Journal of Computer and System Sciences*, 68(4):753–787, 2004.

[14] E. Friedgut. Boolean functions with low average sensitivity depend on few coordinates. *Combinatorica*, 18(1):27–36, 1998.

[15] E. Friedgut. Influences in product spaces: KKL, BKKKL revisited. *Combinatorics, Probability and Computing*, 13:17–29, 2004.

[16] E. Friedgut and G. Kalai. Every monotone graph property has a sharp threshold. In *Proc. Amer. Math. Soc. 124*, pages 2993–3002, 1996.

[17] G.Kalai and S.Safra. Threshold phenomena and influence. *Computational Complexity and Statistical Physics, A.G. Percus, G. Istrate and C. Moore, eds. (Oxford University Press, New York, 2006)*, pages 25–60, 2006.

[18] T. Hancock and Y. Mansour. Learning monotone k-μ DNF formulas on product distributions. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theorey (COLT)*, pages 179–183, 1991.

[19] G. Kalai. A Fourier-theoretic perspective for the Condorcet Paradox and Arrow's Theorem. *Avances in Applied Mathematics*, 29:412–426, 2002.

[20] G. Kalai. Social indeterminacy. *Econometrica*, 72:1565–1581, 2004.

[21] G. Kalai, J. Kahn, and N. Linial. The influence of variables on Boolean functions. In *Proceedings of the Twenty-Ninth Annual Symposium on Foundations of Computer Science (FOCS)*, pages 68–80, 1988.

[22] S. Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the Thirty-Fourth Annual ACM Symposium on the Theory of Computing (STOC)*, pages 767–775, 2002.

[23] M. Krivelevich, B. Sudakov, and V. H. Vu. A sharp threshold for network reliability. *Combinatorics, Probability and Computing*, 11:465–474, 2002.

[24] E. Lehrer. An axiomatization of the Banzhaf value. *International Journal of Game Theory*, 17:89–99, 1988.

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

[26] K. Matulef, R. O'Donnell, R. Rubinfed, and R. A. Servedio. Testing $\{-1, +1\}$ halfspaces. In *Proceedings of the Thirteenth International Workshop on Randomization and Computation (RANDOM)*, pages 646–657, 2009.

[27] K. Matulef, R. Servedio, and K. Wimmer. Personal communication. 2009.

[28] R. O'Donnell, M. Saks, O. Schramm, and R. Servedio. Every decision tree has an influential variable. In *Proceedings of the Forty-Sixth Annual Symposium on Foundations of Computer Science (FOCS)*, pages 31–39, 2005.

[29] R. O'Donnell and R. Servedio. Learning monotone decision trees in polynomial time. *SIAM Journal on Computing*, 37(3):827–844, 2007.

[30] R. O'Donnell and R. Servedio. The Chow parameters problem. In *Proceedings of the Fourtieth Annual ACM Symposium on the Theory of Computing (STOC)*, pages 517–526, 2008.

[31] D. Ron and G. Tsur. Testing membership in width-2 OBDDs. In *Proceedings of the Thirteenth International Workshop on Randomization and Computation (RANDOM)*, pages 686–699, 2009.

[32] L. Russo. On the critical percolation probabilities. *Z. Wahrsch. Verw. Geb.*, 56:229–237, 1981.

[33] M. Talagrand. On Russo's approximate $0-1$ law. *Annals of Probability*, 22:1576–1587, 1994.

[34] M. Talagrand. How much are increasing sets correlated. *Combinatorica*, 16:243–258, 1996.

[35] M. Talagrand. On boundaries and influences. *Combinatorica*, 17:275–285, 1997.