Electronic Colloquium on Computational Complexity, Report No. 131 (2015)

Smooth Boolean functions are easy: efficient algorithms for low-sensitivity functions

Parikshit Gopalan
Microsoft Research
parik@microsoft.com

Noam Nisan
Microsoft Research &
The Hebrew University
noam.nisan@gmail.com

Rocco A. Servedio
Columbia University
rocco@cs.columbia.edu

Kunal Talwar
Google Research
ktalwar@gmail.com

Avi Wigderson
IAS
avi@ias.edu

August 10, 2015

Abstract

A natural measure of smoothness of a Boolean function is its *sensitivity* (the largest number of Hamming neighbors of a point which differ from it in function value). The structure of smooth or equivalently low-sensitivity functions is still a mystery. A well-known conjecture states that every such Boolean function can be computed by a shallow decision tree. While this conjecture implies that smooth functions are easy to compute in the *simplest* computational model, to date no non-trivial upper bounds were known for such functions in *any* computational model, including unrestricted Boolean circuits. Even a bound on the description length of such functions better than the trivial 2^n does not seem to have been known.

In this work, we establish the first computational upper bounds on smooth Boolean functions:

- We show that every sensitivity s function is uniquely specified by its values on a Hamming ball of radius $2s$. We use this to show that such functions can be computed by circuits of size $n^{O(s)}$.
- We show that sensitivity s functions satisfy a strong *pointwise* noise-stability guarantee for random noise of rate $O(1/s)$. We use this to show that these functions have formulas of depth $O(s \log n)$.
- We show that sensitivity s functions can be (locally) self-corrected from worst-case noise of rate $\exp(-O(s))$.

All our results are simple, and follow rather directly from (variants of) the basic fact that the function value at few points in small neighborhoods of a given point determine its function value via a majority vote. Our results confirm various consequences of the conjecture. They may be viewed as providing a new form of evidence towards its validity, as well as new directions towards attacking it.

ISSN 1433-8092

1 Introduction

1.1 Background and motivation

The smoothness of a continuous function captures how gradually it changes locally (according to the metric of the underlying space). For Boolean functions on the Hamming cube, a natural analog is *sensitivity*, capturing how many neighbors of a point have different function values. More formally, the *sensitivity* of a Boolean function $f : \{0,1\}^n \to \{0,1\}$ at input $x \in \{0,1\}^n$, written $s(f,x)$, is the number of neighbors y of x in the Hamming cube such that $f(y) \neq f(x)$. The *max sensitivity* of f, written $s(f)$ and often referred to simply as the "sensitivity of f", is defined as $s(f) = \max_{x\in\{0,1\}^n} s(f,x)$. So, $0 \leq s(f) \leq n$, and while not crucial, it may be good for the reader to consider this parameter as "low" when e.g. either $s(f) \leq (\log n)^{O(1)}$ or $s(f) \leq n^{o(1)}$ (note that both upper bounds are closed under taking polynomials).

To see why low-sensitivity functions might be considered smooth, let $\delta(\cdot,\cdot)$ denote the normalized Hamming metric on $\{0,1\}^n$. A simple application of the triangle inequality gives

$$\mathbf{E}_{y:\delta(x,y)=\delta_0} |f(x) - f(y)| \leq \delta_0 s(f).$$

Thus $s(f)$ might be viewed as being somewhat analogous to the Lipschitz constant of f.

A well known conjecture states that every smooth Boolean function is computed by a shallow decision tree, specifically of depth polynomial in the sensitivity. This conjecture was first posed in the form of a question by Nisan [Nis91] and Nisan and Szegedy [NS94] but is now (we feel) widely believed to be true:

Conjecture 1. *[Nis91, NS94] There exists a constant c such that every Boolean function f has a decision tree of depth $s(f)^c$.*

The converse is trivial, since every Boolean function computable by a depth d decision tree has sensitivity at most d. However, the best known upper bound on decision tree depth in terms of sensitivity is exponential (see Section 1.3).

A remarkable series of developments, starting with Nisan's paper [Nis91], showed that decision tree depth is an extremely robust complexity parameter, in being polynomially related to many other, quite diverse complexity measures for Boolean functions, including PRAM complexity, block sensitivity, certificate complexity, randomized decision tree depth, quantum decision tree depth, real polynomial degree, and approximating polynomial degree. Arguably the one natural complexity measure that has defied inclusion in this equivalence class is sensitivity. Thus, there are many equivalent formulations of Conjecture 1; indeed, Nisan originally posed the question in terms of sensitivity versus block sensitivity [Nis91]. See the extensive survey [HKP11] for much more information about the conjecture and [BdW02] for background on various Boolean function complexity measures.

Conjecture 1 is typically viewed as a *combinatorial* statement about the Boolean hypercube. However, the conjecture also makes a strong assertion about *computation*, stating that smooth functions have very low complexity; indeed, the conjecture posits that they are easy to compute in arguably the simplest computational model — deterministic decision trees. This implies that smooth functions easy for many other "low-level" computational models via the following chain of inclusions:

$$\begin{aligned}\mathsf{DecTree\text{-}depth}(\mathrm{poly}(s)) &\subseteq \mathsf{DNF\text{-}width}(\mathrm{poly}(s)) \subseteq \mathsf{AC_0\text{-}size}(n^{\mathrm{poly}(s)}) \\ &\subseteq \mathsf{Formula\text{-}depth}(\mathrm{poly}(s)\log(n)) \subseteq \mathsf{Circuit\text{-}size}(n^{\mathrm{poly}(s)}).\end{aligned}$$

Given these inclusions, and the widespread interest that Conjecture 1 has attracted in the study of Boolean functions, it is perhaps surprising that no non-trivial upper bounds were previously known on

low sensitivity functions in *any* computational model, including unrestricted Boolean circuits. Indeed, a pre-requisite for a family of functions to have small circuits is an upper bound on the number of functions in the family, or equivalently on the description length of such functions; even such bounds were not previously known for low sensitivity functions. This gap in our understanding of low sensitivity functions helped motivate the present work.

An equivalent formulation of Conjecture 1 is that every sensitivity s function is computed by a real polynomial whose degree is upper bounded by some polynomial in s. This is equivalent to saying that the *Fourier expansion* of the function has degree $\mathrm{poly}(s)$:

Conjecture 2. *[Nis91, NS94] (Equivalent to Conjecture 1) There exist a constant c such that every Boolean function is computed by a real polynomial of degree $s(f)^c$.*

Given the analogy between sensitivity and the Lipschitz constant, this form of the conjecture gives a natural discrete analog of continuous approximations of smooth Lipschitz functions by low-degree polynomials, first obtained for univariate case by Weierstrass [Wei85], which has had a huge influence on the development of modern analysis. This lead to a large body of work in approximation theory, and we mention here the sharp quantitative version of the theorem [Jac30] and its extension to the multivariate case [NS64].

This formulation of the conjecture is also interesting because of the rich structure of low-degree polynomials that low sensitivity functions are believed to share. For instance, low-degree real polynomials on the Boolean cube are easy to interpolate from relatively few values (say over a Hamming ball). The interpolation procedure can be made tolerant to noise, and local (these follow from the fact that low-degree real polynomials also have low degree over $\mathbb{F}_2$). Again, our understanding of the structure of low sensitivity functions was insufficient to establish such properties for them prior to this work.

Finally, to every Boolean function f one can associate the bipartite graph G_f which has left and right vertex sets $f^{-1}(0)$ and $f^{-1}(1)$, and which has an edge (x, y) if the Hamming distance $d(x, y)$ is 1 and $f(x) \neq f(y)$. A function has max sensitivity s if and only the graph G_f has maximum degree at most s. From this perspective one can view Conjectures 1 and 2 as a step towards understanding the graph-theoretic structure of Boolean functions and relating it to their computational and analytic structure (as captured by the Fourier expansion). In this paper, we propose proving various implications of the conjecture both as a necessary first step towards the conjecture, and as a means to better understanding low sensitivity functions from a computational perspective.

1.2 Our Results

Let $\mathcal{F}(s, n)$ denote the set of Boolean functions on n variables such that $s(f) \leq s$. We sometimes refer to this class simply as "sensitivity s functions" (n will be implicit).

The starting point for our results is an upper bound stating that low-sensitivity functions can be interpolated from Hamming balls. This parallels the fact that a degree d polynomial can be interpolated from its values on a Hamming ball of radius d.

Theorem 3. *Every sensitivity s function on n variables is uniquely specified by its values on any Hamming ball of radius $2s$ in $\{0, 1\}^n$.*

The simple insight here is that knowing the values of f at any set of $2s + 1$ neighbors of a point x uniquely specifies the value of f at x: it is the majority value over the $2s + 1$ neighbors (else the point x would be too sensitive). This implies the following upper bound on the number of sensitivity s functions:

$$|\mathcal{F}(s, n)| \leq 2^{\binom{n}{\leq 2s}}.$$

Our proof of Theorem 3 is algorithmic (but inefficient). We build on it to give efficient algorithms that compute f at any point $x \in \{0,1\}^n$, given the values of f on a Hamming ball as advice.

Our first algorithm takes a bottom-up approach. We know the values of f on a ball of small radius around the origin, and wish to infer its value at some arbitrary point x. Imagine moving the center of the ball from the origin to x along a shortest path. The key observation is that after shifting the ball by Hamming distance 1, we can recompute the values of f on the shift using a simple Majority vote.

Our second algorithm uses a top-down approach, reducing computing f at x to computing f at $O(s)$ neighboring points of Hamming weight one less than x. We repeat this till we reach points of weight $O(s)$ (whose values we know from the advice). By carefully choosing the set of $O(s)$ neighbors, we ensure that no more than $n^{O(s)}$ values need to be computed in total:

Theorem 4. *Every sensitivity s function is computed by a Boolean circuit of size $O(sn^{2s+1})$ and depth $O(n^s)$.*

Simon has shown that every sensitivity s function depends on at most $2^{O(s)}$ variables [Sim82]. Thus, the circuit we construct has size at most $2^{O(s^2)}$.

A natural next step would be to parallelize this algorithm. Towards this goal, we show that low sensitivity functions satisfy a very strong noise-stability guarantee: Start at any point $x \in \{0,1\}^n$ and take a random walk of length $n/10s$ to reach a point y. Then $f(x) = f(y)$ with probability 0.9, where the probability is only over the coin tosses of the walk and not over the starting point x. Intutitively, this says that the value of f at most points in a ball of radius $n/10s$ around x equals the value at x (note that in contrast, Theorem 3 only uses the fact that most points in a ball of radius 1 agree with the center). We use this structural property to get a small depth *formula* that computes f:

Theorem 5. *Every sensitivity s function is computed by a Boolean formula of depth $O(s \log n)$ and size $n^{O(s)}$.*

(By [Sim82], these formulas have depth at most $O(s^2)$ and size at most $2^{O(s^2)}$ as before.) At a high level, we again use the the values on a Hamming ball as advice. Starting from some arbitrary input x, we use a variant of the noise-stability guarantee (which holds for "downward" random walks that only flip 1-coordinates to 0) to reduce the computation of f at x to computing f on $O(1)$ many points whose weight is less than that of x by a factor of roughly $(1 - 1/(10s))$ (a majority vote on these serves to amplify the success probability). Repeating this for each of these new points, recursively, for $O(s \log(n))$ times, we reduce computing f at x to computing f at various points in a small Hamming ball around the origin, which we know from the advice.

We also show that low-sensitivity functions admit local self-correction. The setup here is that we are given oracle access to an unknown function $r : \{0,1\}^n \to \{0,1\}$ that is promised to be close to a low sensitivity function. Formally, there exists a sensitivity s function $f : \{0,1\}^n \to \{0,1\}$ such that

$$\delta(r, f) := \Pr_{x \in \{0,1\}^n}[r(x) \neq f(x)] \leq 2^{-ds}$$

for some constant d. We are then given an arbitrary $x \in \{0,1\}^n$ as an input, and our goal is to return $f(x)$ correctly with high probability for every x, where the probability is over the coin tosses of the (randomized) algorithm. We show that there is a self-corrector for f with the following guarantee:

Theorem 6. *There exist a constant d such that the following holds. Let $r : \{0,1\}^n \to \{0,1\}$ be such that $\delta(r, f) \leq 2^{-ds}$ for some sensitivity s function f. There is an algorithm which, when given an oracle for r and $x \in \{0,1\}^n$ as input, queries the oracle for r at $(n/s)^{O(s)}$ points, runs in $(n/s)^{O(s)}$ time, and returns the correct value of $f(x)$ with probability 0.99.*

Our self-corrector is similar in spirit to our formula construction: our estimate for $f(x)$ is obtained by taking the majority over a random sample of points in a ball of radius $n/10s$. Rather than querying these points directly (since they might all be incorrect for an adversarial choice of x and r), we use recursion. We show that $O(s \log(n))$ levels of recursion guarantee that we compute $f(x)$ with good probability. The analysis uses Bonami's hypercontractive inequality [O'D14].

Our results imply that low-degree functions and low sensitivity functions can each be reconstructed from their value on small Hamming balls using simple but dissimilar looking "propagation rules". We show how degree and sensitivity can be chracterized by the convergence of these respective propagation rules, and use this to present a reformulaion of Conjecture 1.

1.3 Related Work

The study of sensitivity originated from work on PRAMs [CDR86, Sim82]. As mentioned earlier, the question of relating sensitivity to other complexity measures such as block sensitivity was posed in [NS94]. There has been a large body of work on Conjecture 1 and its equivalent formulations, and recent years have witnessed significant interest in this problem (see the survey [HKP11] and the papers cited below). To date, the biggest gap known between sensitivity and other measures such as block-sensitivity, degree and decision tree depth is at most quadratic [Rub95, AS11]. Upper bounds on other measures such as block sensitivity and certificate complexity in terms of sensitivity are given in [KK04, ABG+14, AP14, APV15] (see also [AV15]). Very recently, a novel approach to this conjecture via a communication game was proposed in the work of Gilmer *et al.* [GKS15].

1.4 Preliminaries

We define the 0-sensitivity, 1-sensitivity and the max sensitivity of an n-variable function f as

$$s_0(f) = \max_{x \in f^{-1}(0)} s(f,x), \quad s_1(f) = \max_{x \in f^{-1}(1)} s(f,x), \quad s(f) = \max_{x \in \{0,1\}^n} s(f,x) = \max(s_0(f), s_1(f)).$$

We denote the real polynomial degree of a function by $\deg(f)$ and its $\mathbb{F}_2$ degree by $\deg_2(f)$. We write $\mathrm{wt}(x)$ for $x \in \{0,1\}^n$ to denote the Hamming weight of x (number of ones). We write $\delta(f,g)$ for $f, g : \{0,1\}^n \to \{0,1\}$ to denote $\Pr_{x \in \{0,1\}^n}[f(x) \neq g(x)]$.

For $x \in \{0,1\}^n$, let $\mathcal{B}(x,r) \subset \{0,1\}^n$ denote the Hamming ball consisting of all points at distance at most r from x. Let $\mathcal{S}(x,r)$ denote the Hamming sphere consisting of all points at distance exactly r from x. Let $N(x)$ denote the set of Hamming neighbors of x (so $N(x)$ is shorthand for $\mathcal{S}(x,1)$), and let $N_r(x)$ denote the set of neighbors of Hamming weight r (points with exactly r ones).

The following upper bound on sensitivity in terms of degree is due to Nisan and Szegedy.

Theorem 7. *[NS94] For every function $f : \{0,1\}^n \to \{0,1\}$, we have $\mathrm{s}(f) \leq 4(\deg(f))^2$.*

We record Simon's upper bound on the number of relevant variables in a low-sensitivity function:

Theorem 8. *[Sim82] For every function $f : \{0,1\}^n \to \{0,1\}$, the number of relevant variables n' is bounded by $n' \leq s(f)4^{s(f)}$.*

2 Structural properties of low sensitivity functions

2.1 Bounding the description length

We show that functions with low sensitivity have concise descriptions, so consequently the number of such functions is small. Indeed, we show that knowing the values on a Hamming ball of radius $2s+1$ suffices.

2.1.1 Reconstruction from Hamming balls and spheres.

The following simple but key observation will be used repeatedly:

Lemma 9. *Let $S \subseteq N(x)$ where $|S| \geq 2s+1$. Then $f(x) = \mathrm{Maj}_{y \in S}(f(y))$.*

Proof: Let $b \in \{0,1\}$ denote the majority value of f over S and let $S^b \subset S$ be the subset of S over which f takes the value b. Note that $|S^b| \geq \lceil |S|/2 \rceil \geq s+1$ since $|S| \geq 2s+1$. If $f(x) \neq b$, then every vertex in S^b represents a sensitive neighbor of x, and thus $s(f,x) \geq s+1$ which is a contradiction. ∎

Theorem 10. *Every sensitivity s function is uniquely specified by its values on a ball of radius $2s$.*

Proof: Suppose that we know the values of f on $\mathcal{B}(x, 2s)$. We may assume by relabeling that $x = 0^n$ is the origin. Note that $\mathcal{B}(0^n, 2s)$ is just the set of points of Hamming weight at most $2s$.

We will prove that f is uniquely specified on points x where $\mathrm{wt}(x) \geq 2s$ by induction on $r = \mathrm{wt}(x)$. The base case $r = 2s$ is trivial. For the induction step, assume we know f for all points of weight up to r for some $r \geq 2s$. Consider a point x with $\mathrm{wt}(x) = r+1$. The set $N_r(x)$ of weight-r neighbors of x has size $r+1 \geq 2s+1$. Hence

$$f(x) = \operatorname*{Maj}_{y \in N_r(x)} (f(y)). \tag{1}$$

by Lemma 9. ∎

Note that by Equation 1, we only need to know f on the sphere of radius r rather than the entire ball to compute f on inputs of weight $r+1$. This observation leads to the following sharpening for $s \leq n/4$.

Corollary 11. *Let $s \leq n/4$. Every sensitivity s function is uniquely specified by its values on a sphere of radius $2s$.*

Proof: As before we may assume that $x = 0^n$. By Equation 1, the values of f on $\mathcal{S}(0^n, r)$ fix the values at $\mathcal{S}(0^n, r+1)$. Hence knowing f on $\mathcal{S}(0^n, 2s)$ suffices to compute f at points of weight $2s+1$ and beyond. In particular, the value of f is fixed at all points of weight $n/2$ through n (since $2s \leq n/2$). Hence the value of f is fixed at all points of the ball $\mathcal{B}(1^n, 2s)$, and now Theorem 10 finishes the proof. ∎

2.1.2 Upper and lower bounds on $\mathcal{F}(s,n)$.

Recall that $|\mathcal{F}(s,n)|$ denotes the number of distinct Boolean functions on n variables with sensitivity at most s. We use the notation $\binom{n}{\leq k}$ to denote $\sum_{i=0}^{k} \binom{n}{i}$, the cardinality of a Hamming ball of radius k.

As an immediate corollary of Theorem 10, we have the following upper bound:

Corollary 12. *For all $s \leq n$, we have $|\mathcal{F}(s,n)| \leq 2^{\binom{n}{\leq 2s}}$.*

We have the following lower bounds:

Lemma 13. *For all $s \leq n$, we have $|\mathcal{F}(s,n)| \geq \max\left(\binom{n}{s}2^{2^s-1}, (n-s+1)^{2^{s-1}}\right)$.*

Proof: The first bound comes from considering s-juntas. We claim that there are at least 2^{2^s-1} functions on s variables that depend on all s variables. For any function $f : \{0,1\}^s \to \{0,1\}$ on s variables, either f or $f' = f \oplus \prod_{i=1}^{s} x_i$ is sensitive to all s variables. This is because $f \oplus f' = \prod_{i=1}^{s} x_i$, hence one of them has full degree as a polynomial over $\mathbb{F}_2$, and hence must depend on all n variables. The bound now follows by considering all subsets of n variables.

The second bound comes from the addressing functions. Divide the variables into $s-1$ address variables $y_1, \ldots, y_{s-1}$ and $n-s+1$ output variables $x_1, \ldots, x_{n-s+1}$. Consider the addressing function computed by a decision tree with nodes at the first $s-1$ levels labelled by $y_1, \ldots, y_{s-1}$ and each leaf labelled by some x_i (the same x_i can be repeated at multiple leaves). It is easy to check that this defines a family of sensitivity s functions, that all the functions in the family are distinct, and that the cardinality is as claimed. ∎

In the setting when $s = o(n)$, the gap between our upper and lower bounds is roughly 2^{n^s} versus n^{2^s}. The setting where $s = O(\log(n))$ is particularly intriguing.

Problem 14. *Is $|\mathcal{F}(2\log(n), n)| = 2^{n^{\omega(1)}}$?*

2.2 Noise Stability

We start by showing that functions with small sensitivity satisfy a strong noise-stability guarantee.

For a point $x \in \{0,1\}^n$ and $\delta \in [0,1]$, let $\mathrm{N}_{1-2\delta}(x)$ denote the δ-noisy version of x, i.e. a draw of $y \sim \mathrm{N}_{1-2\delta}(x)$ is obtained by independently setting each bit y_i to be x_i with probability $1-2\delta$ and uniformly random with probability 2δ. The noise sensitivity of f at x at noise rate δ, denoted $\mathrm{NS}_\delta[f](x)$, is defined as

$$\mathrm{NS}_\delta[f](x) = \Pr_{y \sim \mathrm{N}_{1-2\delta}(x)}[f(x) \neq f(y)].$$

The noise sensitivity of f at noise rate δ, denoted $\mathrm{NS}_\delta[f]$, is then defined as

$$\mathrm{NS}_\delta[f] = \mathop{\mathbb{E}}_{x \sim \{0,1\}^n}[\mathrm{NS}_\delta[f](x)] = \Pr_{x \sim \{0,1\}^n, y \sim \mathrm{N}_{1-2\delta}(x)}[f(x) \neq f(y)].$$

The next lemma shows that low-sensitivity functions are noise-stable at every point $x \in \{0,1\}^n$:

Lemma 15. *Let $f : \{0,1\}^n \to \{0,1\}$ have sensitivity s. For every $x \in \{0,1\}^n$ and $0 \leq \delta \leq 1/2$, we have $\mathrm{NS}_\delta[f](x) < 2\delta s$.*

Proof: Let $t \in [n]$. Consider a random process that starts at x and then flips a uniformly random subset $T \subseteq [n]$ of coordinates of cardinality t, which takes it from x to $y \in \{0,1\}^n$. We claim that $\Pr_T[f(x) \neq f(y)] \leq \frac{st}{n-t+1}$. To see this, we can view going from x to y as a walk where at each step, we pick the next coordinate to walk along uniformly from the set of coordinates that have not been selected so far. Let $x = x_0, x_1, \ldots, x_t = y$ denote the sequence of vertices visited by this walk. At x_i, we choose the next coordinate to flip uniformly from a set of size $n-i$. Since x_i has at most s sensitive coordinates, we have $\Pr[f(x_i) \neq f(x_{i+1})] \leq \frac{s}{n-i}$. Hence by a union bound we get

$$\Pr[f(x_0) \neq f(x_t)] \leq \sum_{i=0}^{t-1} \Pr[f(x_i) \neq f(x_{i+1})] \leq \sum_{i=0}^{t-1} \frac{s}{n-i} \leq \frac{st}{n-t+1}$$

as claimed.

Now we turn to noise sensitivity. We can view a draw of $y \sim \mathrm{N}_{1-2\delta}(x)$ as first choosing the number t of coordinates of x to flip according to the binomial distribution $\mathrm{Bin}(n,\delta)$, and then flipping a random set $T \subseteq [n]$ of size t. From above, we have $\Pr[f(y) \neq f(x) \mid |T| = t] \leq \frac{st}{n-t+1}$. Hence

$$\begin{aligned}
\Pr[f(x) \neq f(y)] &\leq \mathop{\mathbb{E}}_{t\sim\mathrm{Bin}(n,\delta)}\left[\frac{st}{n-t+1}\right] = s\sum_{t=1}^{n}\delta^t(1-\delta)^{n-t}\binom{n}{t}\cdot\frac{t}{n-t+1} \\
&= s\sum_{t=1}^{n}\delta^t(1-\delta)^{n-t}\binom{n}{t-1} \\
&= \frac{s\delta}{1-\delta}\sum_{t'=0}^{n-1}\delta^{t'}(1-\delta)^{n-t'}\binom{n}{t'} \\
&= \frac{s\delta}{1-\delta}(1-\delta^n)
\end{aligned}$$

which is less than $2\delta s$ for $\delta \leq 1/2$. ∎

We can restrict the noise distribution and get similar bounds. The setting that we now describe, where we only allow walks in the lower shadow of a vertex, will be useful later when we construct shallow formulas for a low sensitivity function f.

Let $D(x,t)$ denote the points in the lower shadow of x at distance t from it (so a point in $D(x,t)$ is obtained by flipping exactly t of the bits of x from 1 to 0). We show that a random point in $D(x,t)$ is likely to agree with x (for $t \leq \mathrm{wt}(x)/2s$).

Lemma 16. *Let* $\mathrm{wt}(x) = d \geq s$. *Then if* $s(f) \leq s$, *we have* $\Pr_{y\in D(x,t)}[f(x) \neq f(y)] \leq \frac{st}{d-t}$.

Proof: We consider a family of random walks that we call *downward walks*. In such a walk, at each step we pick a random index that is currently 1 and set it to 0. Consider a downward walk of length t and let $x = x_0, x_1, \ldots, x_t = y$ denote the sequence of vertices that are visited by the walk. We claim that $\Pr[f(x_i) \neq f(x_{i+1})] \leq \frac{s}{d-i}$. To see this observe that out of the $d-i$ possible 1 indices in x_i that could be flipped to 0, at most s are sensitive. Hence we have

$$\Pr[f(x_0) \neq f(x_t)] \leq \sum_{i=0}^{t-1}\Pr[f(x_i) \neq f(x_{i+1})] \leq \sum_{i=0}^{t-1}\frac{s}{d-i} \leq \frac{st}{d-t}$$

Since $y = x_t$ is a random point in $D(x,t)$, the proof is complete. ∎

Corollary 17. *Let* $\mathrm{wt}(x) = d$ *and* $t \leq d/(10s+1)$. *Then* $\Pr_{y\in D(x,t)}[f(y) \neq f(x)] \leq 1/10$.

2.3 Bias and Interpolation

It is known that low sensitivity functions cannot be highly biased. For $f : \{0,1\}^n \to \{0,1\}$, let

$$\mu_0(f) = \Pr_{x\in\{0,1\}^n}[f(x) = 0],\ \mu_1(f) = \Pr_{x\in\{0,1\}^n}[f(x) = 1],$$
$$\mu(f) = \min(\mu_0(f), \mu_1(f))$$

Lemma 18. *For $f : \{0,1\}^n \to \{0,1\}$ we have*

$$s_0(f) \geq \log_2\left(\frac{1}{\mu_0(f)}\right) \text{ if } \mu_0(f) > 0$$
$$s_1(f) \geq \log_2\left(\frac{1}{\mu_1(f)}\right) \text{ if } \mu_1(f) > 0.$$

Equality holds iff the set $f^{-1}(b)$ is a subcube.

We note that these bounds are implied by the classical isoperimetric inequality, which in fact shows that $\mathbf{E}_{x \in f^{-1}(b)}[s(f,x)] \geq \log(1/\mu_b(f))$ for $b = 0, 1$. We present a simple inductive proof of the max-sensitivity bounds given by Lemma 18 in the appendix.

We say that a set $K \subseteq \{0,1\}^n$ *hits* a set of functions $\mathcal{F}$ if for every $f \in \mathcal{F}$, there exists $x \in K$ such that $f(x) \neq 0$. We say that K *interpolates* $\mathcal{F}$ if for every $f_1 \neq f_2 \in \mathcal{F}$, there exists $x \in K$ such that $f_1(x) \neq f_2(x)$.

Corollary 19. *Let $k \geq C2^{2s}\binom{n}{\leq 4s}$, and let S be a random subset of $\{0,1\}^n$ obtained by taking k points drawn uniformly from $\{0,1\}^n$ with replacement. The set S interpolates $\mathcal{F}(s,n)$ with probability $1 - \exp(-\binom{n}{\leq 4s})$ (over the choice of S).*

Proof: We first show that large sets hit $\mathcal{F}(t,n)$ with very high probability. Fix $f \in \mathcal{F}(t,n)$. Since we have $\mu_1(f) \geq 2^{-t}$ by Lemma 18, the probability that k random points all miss $f^{-1}(1)$ is bounded by $(1-2^{-t})^k \leq \exp(-k/2^t)$. By Corollary 12 we have $\mathcal{F}(t,n) \leq 2^{\binom{n}{\leq 2t}}$, so by the union bound, the probability that S does not hit every function in this set is at most $2^{\binom{n}{\leq 2t}}\exp(-k/2^t)$, which is $\exp(-\binom{n}{\leq 2t})$ for $k \geq C2^t\binom{n}{\leq 2t}$.

Next, we claim that if S hits $\mathcal{F}(2s,n)$ then it interpolates $\mathcal{F}(s,n)$. Given functions $f_1, f_2 \in \mathcal{F}(s,n)$, let $g = f_1 \oplus f_2$. It is easy to see that $g \in \mathcal{F}(2s,n)$. and that $g^{-1}(1)$ is the set of points x where $f_1(x) \neq f_2(x)$, so indeed if S hits $\mathcal{F}(2s,n)$ then it interpolates $\mathcal{F}(s,n)$. Given this, the corollary follows from our lower bound on k, taking $t = 2s$. ∎

3 Efficient algorithms for computing low sensitivity functions

3.1 Small circuits

In this subsection, we will prove Theorem 4. Recall that the proof of Theorem 10 gives an algorithm to compute the truth table of f from an advice string which tells us the values on some Hamming ball of radius $2s+1$. In this section we present two algorithms which, given this advice, can (relatively) efficiently compute any entry of the truth table without computing the truth-table in its entirety. This is equivalent to a small circuit computing f. We first give a (non-uniform) "bottom-up" algorithm for computing f at a given input point $x \in \{0,1\}^n$. In the appendix we describe a "top-down" algorithm with a similar performance bound.

3.1.1 A Bottom-Up Algorithm

The algorithm takes as advice the values of f on $\mathcal{B}(0^n, 2s)$. It then shifts the center of the ball along a shortest path from 0^n to x, computing the values of f on the shifted ball at each step. This computation is

made possible by a lemma showing that when we shift a Hamming ball $\mathcal{B}$ by a unit vector to get a new ball $\mathcal{B}'$, points in $\mathcal{B}'$ either lie in $\mathcal{B}$ or are adjacent to many points in $\mathcal{B}$, which lets us apply Lemma 9.

Let $\mathbb{1}(S)$ denote the indicator of $S \subseteq [n]$ and $S \Delta T$ denote the symmetric difference of the sets S, T. For $B \subseteq \{0,1\}^n$ we write $B \oplus e_i$ to denote the pointwise shift of B by the unit vector e_i.

Lemma 20. *For any $y \in \mathcal{B}(x \oplus e_i, r) \setminus \mathcal{B}(x, r)$, we have $|N(y) \cap \mathcal{B}(x, r)| = r + 1$.*

Proof: Fix any such y. Since $\mathcal{B}(x \oplus e_i, r) = \mathcal{B}(x, r) \oplus e_i$, we have that

$$\begin{aligned} y &= x' \oplus e_i \text{ for some } x' \in \mathcal{B}(x, r), \text{ where} \\ x' &= x \oplus \mathbb{1}(S) \text{ for some } S \subseteq [n], |S| \leq r, \text{ and hence} \\ y &= x \oplus \mathbb{1}(S \Delta \{i\}). \end{aligned}$$

If $i \in S$ or $|S| \leq r - 1$, then $|S \Delta \{i\}| \leq r$; but this means that $y \in \mathcal{B}(x, r)$, which is in contradiction to our assumption that $y \in \mathcal{B}(x \oplus e_i, r) \setminus \mathcal{B}(x, r)$. Hence $i \notin S$ and $|S| = r$. But then we have $y \oplus e_j \in \mathcal{B}(x, r)$ for precisely those j that belong to $S \cup \{i\}$, which gives the claim. ∎

Corollary 21. *Knowing the values of f on $\mathcal{B}(x, 2s)$ lets us compute f on $\mathcal{B}(x \oplus e_i, 2s)$.*

Proof: Either $y \in \mathcal{B}(x \oplus e_i, 2s)$ lies in $\mathcal{B}(x, 2s)$ so we know $f(y)$ already, or by the previous lemma y has $2s + 1$ neighbors in $\mathcal{B}(x, 2s)$, in which case Lemma 9 gives $f(y) = \mathrm{Maj}_{y' \in N(y) \cap \mathcal{B}(x, 2s)}(f(y'))$. ∎

Now we can give our algorithm for computing $f(x)$ at an arbitrary input $x \in \{0,1\}^n$.

Bottom-Up
Advice: The value of f at all points in $\mathcal{B}(0^n, 2s)$.
Input: $x \in \{0,1\}^n$.

1. Let $0^n = x_0, x_1, \ldots, x_d = x$ be a shortest path from 0^n to x.
2. For $i \in \{1, \ldots, d\}$ compute f on $\mathcal{B}(x_i, 2s)$ using the values at points in $\mathcal{B}(x_{i-1}, 2s)$.
3. Output $f(x_d)$.

Theorem 22. *The algorithm* **Bottom-Up** *computes $f(x)$ for any input x in time $O(sn^{2s+1})$ using space $O(n^{2s})$.*

Proof: The values at $\mathcal{B}(0^n, 2s)$ are known as advice. Corollary 21 tells us how to compute the values at $\mathcal{B}(x_i, 2s)$ using the values on $\mathcal{B}(x_{i-1}, 2s)$. If we store the values at $\mathcal{B}(x_{i-1}, 2s)$ in an array indexed by subsets of size $2s$, the value at any point $y \in \mathcal{B}(x_i, 2s)$ can be computed in time $O(s)$, by performing $2s + 1$ array lookups and then taking the majority. Thus computing the values over the entire ball takes time $O(sn^{2s})$, and we repeat this $d \leq n$ times. Finally, at stage i we only need to store the values of f on the latest shift, $\mathcal{B}(x_{i-1}, 2s)$, so the total space required is $O(n^{2s})$. ∎

3.2 Small-depth Formulas

Theorem 22 established that any n-variable sensitivity-s function f is computed by a circuit of size $O(sn^{2s+1})$, but of relatively large depth $O(n^{2s})$. In this section we improve this depth by showing that *shallow* circuits of essentially the same size (equivalently, formulas of small depth) can compute low-sensitivity functions.

For $\mu < 1/2$, let $B(c,\mu)$ denote the product distribution over $y \in \{0,1\}^c$ where $\Pr[y_i = 1] = \mu$ for each $i \in [c]$. For constants $1/2 > \mu > \delta > 0$, let $c = c(\mu,\delta) \in \mathbb{Z}$ be the smallest integer constant such that

$$\Pr_{y \sim B(c,\mu)}[\mathrm{Maj}_{i\in[c]}(y_i) = 1] \leq \delta.$$

We now present a randomized parallel algorithm for computing $f(x)$.

Parallel-Algorithm
Advice: f at all points in $\mathcal{B}(0^n, 10s)$.
Input: $x \in \{0,1\}^n$.
Let $d = \mathrm{wt}(x)$, $t = \lfloor d/(10s+1) \rfloor$, $c = c(1/5, 1/20)$.

1. If $d \leq 10s$, return $A(x) = f(x)$.
2. Else sample $y_1, \ldots, y_c$ randomly from $D(x,t)$. Recursively run **Parallel-Algorithm** to compute $A(y_i)$ in parallel for all $i \in [c]$.
3. Return $A(x) = \mathrm{Maj}_{i\in[c]}(A(y_i))$.

For brevity we use A to denote the algorithm above and $A(x) \in \{0,1\}$ to denote the random variable which is its output on input x. For $d \geq 10s+1$, the random choices of A in computing $A(x)$ are described by a c-regular tree. The tree's root is labeled by x and its children are labeled by $y_1, \ldots, y_c$; its leaves are labeled by strings that each have Hamming weight at most $10s$. Further, the various subtrees rooted at each level are independent of each other.

Theorem 23. *The algorithm runs in parallel time $O(s \log n)$ using $n^{O(s)}$ processors. For any $x \in \{0,1\}^n$, we have $\Pr_A[A(x) \neq f(x)] \leq \frac{1}{20}$, where $\Pr_A$ denotes that the probability is over the random coin tosses of the algorithm.*

Proof: We first prove the correctness of the algorithm by induction on $\mathrm{wt}(x) = d$. When $d \leq 10s$, the claim follows trivially. Assume that the claim is true for $\mathrm{wt}(x) \leq d-1$, and consider an input x of weight d. Note that every $y \in D(x,t)$ has $\mathrm{wt}(y) = d - t \leq d-1$, hence the inductive hypothesis applies to it. For each $i \in [c]$, we independently have

$$\Pr_A[A(y) \neq f(x)] \leq \Pr_{A,y_i}[A(y_i) \neq f(y_i)] + \Pr_{y_i \in D(x,t)}[f(y_i) \neq f(x)] \leq \frac{1}{10} + \frac{1}{20} < \frac{1}{5}.$$

where the $1/10$ bound is by Corollary 17 and the $1/20$ is by the inductive hypothesis. The algorithm samples c independent points $y_i \in D(x,t)$, computes $A(y_i)$ for each of them using independent randomness, and then returns the majority of $A(y_i)$ over those $i \in [c]$. Hence, by our choice of $c = c(1/5, 1/20)$, we have that $\Pr_A[\mathrm{Maj}_{i\in[c]}(A(y_i)) \neq f(x)] \leq \frac{1}{20}$.

To bound the running time, we observe that for $d \geq 10s+1$,

$$t = \left\lfloor \frac{d}{10s+1} \right\rfloor \geq \frac{d}{25s}, \quad \text{so} \quad d - t \leq d\left(1 - \frac{1}{25s}\right).$$

But this implies that in $k = O(s \log d)$ steps, the weight reduces below $10s+1$. The number of processors required is bounded by $c^k = n^{O(s)}$. ∎

By hardwiring the random bits and the advice bits, we can conclude that functions with low sensitivity have small-depth formulas, thus proving Theorem 5.

4 Self-correction

In this section we show that functions with low sensitivity admit self-correctors. Recall that for Boolean functions, $f, g : \{0,1\}^n \to \{0,1\}$ we write $\delta(f,g)$ to denote $\Pr_{x \in \{0,1\}^n}[f(x) \neq g(x)]$.

Our self-corrector is given a function $r : \{0,1\}^n \to \{0,1\}$ such that there exists $f \in \mathcal{F}(s,n)$ satisfying $\delta(r,f) \leq 2^{-cs}$ for some constant $c > 2$ to be specified later. By Lemma 18, it follows that any two sensitivity s functions differ in at least 2^{-2s} fraction of points, so if such a function f exists, it must be unique. We consider two settings (in analogy with coding theory): in the global setting, the self-corrector is given the truth-table of r as input and is required to produce the truth-table of f as output. In the local setting, the algorithm has black-box oracle access to r. It is given $x \in \{0,1\}^n$ as input, and the desired output is $f(x)$.

At a high level, our self-corrector relies on the fact that small-sensitivity sets are noise-stable at noise rate $\delta \approx 1/s$, by Lemma 15, whereas small sets of density $\mu \leq c^{-s}$ tend to be noise sensitive. The analysis uses the hypercontractivity of the $T_{1-2\delta}(\cdot)$ operator.

Following [O'D14], for $f : \{0,1\}^n \to \mathbb{R}$, we define

$$T_{1-2\delta} f(x) = \mathop{\mathbb{E}}_{y \sim \mathrm{N}_{1-2\delta}(x)}[f(y)],$$

where recall that a draw of $y \sim \mathrm{N}_{1-2\delta}(x)$ is obtained by independently setting each bit y_i to be x_i with probability $1-2\delta$ and uniformly random with probability 2δ. We can view (x,y) where $x \sim \{0,1\}^n$ and $y \sim \mathrm{N}_{1-2\delta}(x)$ as defining a distribution on the edges of the complete graph on the vertex set $\{0,1\}^n$. We refer to this weighted graph as the *δ-noisy hypercube*. The $(2,q)$-Hypercontractivity Theorem quantifies the expansion of the noisy hypercube:

Theorem 24. *($(2,q)$-Hypercontractivity.) Let $f : \{0,1\}^n \to \mathbb{R}$. Then*

$$\|T_{1-2\delta} f\|_q \leq \|f\|_2 \text{ for } 2 \leq q \leq 1 + \frac{1}{(1-2\delta)^2}.$$

We need the following consequence, which says that for any small set S, most points do not have too many neighbors in the noisy hypercube that lie within S. For $S \subseteq \{0,1\}^n$, let us define the set $\Lambda_{\delta,\theta}(S)$ of those points for which a θ fraction of neighbors in the δ-noisy hypercube lie in S. Formally,

$$\Lambda_{\delta,\theta}(S) = \{x \in \{0,1\}^n \; s.t. \mathop{\Pr}_{y \sim \mathrm{N}_{1-2\delta}(x)}[y \in S] \geq \theta\}.$$

Abusing the notation from Section 2.3, for $S \subseteq \{0,1\}^n$ we write $\mu(S)$ to denote $\Pr_{x \in \{0,1\}^n}[x \in S]$.

Lemma 25. *We have*

$$\mu(\Lambda_{\delta,\theta}(S)) \leq \left(\frac{\mu(S)}{\theta^2}\right)^{1+2\delta}.$$

Proof: Let $f(x) = \mathbb{1}(x \in S)$. Then

$$T_{1-2\delta}f(x) = \Pr_{y \in \mathrm{N}_{1-2\delta}(x)}[y \in S].$$

Hence $\Lambda_{\delta,\theta}(S)$ is the set of those x such that $T_{1-2\delta}f(x) \geq \theta$.

Let $q = 2(1+2\delta)$. It is easy to see that q satsfies the hypothesis of Theorem 24. Hence we can bound the q^{th} moment of $T_{1-2\delta}f$ as

$$\mathop{\mathbb{E}}_{x \in \{0,1\}^n}[(T_{1-2\delta}f(x))^q] \leq \|f\|_2^q = \mu(S)^{q/2}.$$

Hence by Markov's inequality,

$$\Pr_{x \in \{0,1\}^n}[T_{1-2\delta}f(x) \geq \theta] \leq \frac{\mu(S)^{q/2}}{\theta^q}.$$

The claim follows from our choice of q. ∎

Corollary 26. *If $\mu(S) \leq \theta^{4+2/\delta}$, then $\mu(\Lambda_{\delta,\theta}(S)) \leq \mu(S)^{1+\delta}$.*

Proof: By Lemma 25, it suffices that $\left(\frac{\mu(S)}{\theta^2}\right)^{1+2\delta} \leq \mu(S)^{1+\delta}$, and it is easy to check that this condition holds for our choice of $\mu(S)$. ∎

4.1 Global Self-correction

Our global self-corrector is given a function $r : \{0,1\}^n \to \{0,1\}$ such that there exists $f \in \mathcal{F}(s,n)$ satisfying $\delta(r,f) \leq 2^{-c_1 s}$ for some constant $c_1 > 2$ to be specified later. By Lemma 18, it follows that any two sensitivity s functions differ in at least 2^{-2s} fraction of points, so such a function f if it exists must be unique. Our self-corrector defines a sequence of functions $f_0, \ldots, f_T$ such that $f_0 = r$ and $f_T = f$ (with high probability).

Global Self-corrector
Input: $r : \{0,1\}^n \to \{0,1\}^n$ such that $\delta(r,f) \leq 2^{-c_1 s}$ for some $f \in \mathcal{F}(s,n)$.
Output: The sensitivity-s function f.

Let $f_0 = r$, $k = c_2 s \log(n/s)$, $\delta = 1/(20s)$.
For $t = 1, \ldots, k$,
  For every $x \in \{0,1\}^n$,
    Let $f_t(x) = \mathrm{Maj}_{y \sim \mathrm{N}_{1-2\delta}(x)}(f_{t-1}(y))$.
Return f_k.

The algorithm runs in time $2^{O(n)}$, which is polynomial in the length of its output (which is a truth table of size 2^n). To analyze the algorithm, let us define the sets S_t for $t \in \{0, \ldots, T\}$ as

$$S_t = \{x \in \{0,1\}^n \text{ such that } f_t(x) \neq f(x).\}$$

The following is the key lemma for the analysis.

Lemma 27. *We have* $S_t \subseteq \Lambda_{\delta,2/5}(S_{t-1})$.

Proof: For $x \in S_t$,

$$f(x) \neq \operatorname*{Maj}_{y \sim \mathrm{N}_{1-2\delta}(x)}(f_{t-1}(y)),$$

hence

$$\Pr_{y \sim \mathrm{N}_{1-2\delta}(x)}[f(x) \neq f_{t-1}(y)] \geq \frac{1}{2}.$$

We can upper bound this probability by

$$\Pr_{y \sim \mathrm{N}_{1-2\delta}(x)}[f(x) \neq f_{t-1}(y)] \leq \Pr_{y \sim \mathrm{N}_{1-2\delta}(x)}[f(x) \neq f(y)] + \Pr_{y \sim \mathrm{N}_{1-2\delta}(x)}[f(y) \neq f_{t-1}(y)].$$

Since the distributions $\mathrm{Noise}_\delta(x)$ and $\mathrm{N}_{1-2\delta}(x)$ are identical, we can bound the first term by Lemma 15, which gives

$$\Pr_{y \sim \mathrm{N}_{1-2\delta}(x)}[f(x) \neq f(y)] \leq 2s\delta = \frac{1}{10}.$$

Hence

$$\Pr_{y \sim \mathrm{N}_{1-2\delta}(x)}[f(y) \neq f_{t-1}(y)] \geq \frac{1}{2} - \frac{1}{10} = \frac{2}{5}.$$

But $f(y) \neq f_{t-1}(y)$ implies $y \in S_{t-1}$, hence by definition of $\Lambda_{\delta,\theta}(S)$ we have $x \in \Lambda_{\delta,2/5}(S_{t-1})$. ∎

We can now analyze our global self-corrector.

Theorem 28. *There exist constants* c_1, c_2 *such that if* $\delta(r, f) \leq 2^{-c_1 s}$ *for some* $f \in \mathcal{F}(s, n)$*, then for* $k \geq c_2 s \log(n/s)$*, we have* $f_k = f$.

Proof: Let $\delta = 1/(20s)$. Assume that there exists $f \in \mathcal{F}(s, n)$ such that

$$\delta(f, s) = \mu(S_0) \leq 2^{-c_1 s} \leq (2/5)^{4+40s}.$$

By Lemma 27 and Corollary 26, we have

$$\mu(S_t) \leq \mu(\Lambda_{\delta,2/5}(S_{t-1})) \leq \mu(S_{t-1})^{1+\delta} \leq \mu(S_0)^{(1+\delta)^t}.$$

For $t \geq c_2' \ln(n/s)/\delta = c_2 s \log(n/s)$, we have

$$\mu(S_t) \leq \mu(S_0)^{(1+\delta)^t} < 2^{-n}.$$

But since $S_t \subseteq \{0,1\}^n$, it must be the empty set, and this implies that $f_t = f$. ∎

4.2 Local Self-Correction

Recall that in the local self-correction problem, the algorithm is given $x \in \{0,1\}^n$ as input and oracle access to $r : \{0,1\}^n \to \{0,1\}$ such that $\delta(r, f) \leq 2^{-d_1 s}$ for some constant $d_1 > 2$ to be specified later. The goal is to compute $f(x)$. Our local algorithm can be viewed as derived from the global algorithm, where we replace the Majority computation with sampling, and only compute the parts of the truth tables that are essential to computing $f_T(x)$.

We define a distribution $\mathcal{T}_k(x)$ over c-regular trees of depth k rooted at x, where each tree node is labelled with a point in $\{0,1\}^n$. To sample a tree $T_1(x)$ from $\mathcal{T}_1(x)$, we place x at the root, then sample c independent points from $\mathrm{N}_{1-2\delta}(x)$, and place them at the leaves. To sample a tree $T_k(x)$ from $\mathcal{T}_k(x)$, we first sample $T_{k-1}(x) \sim \mathcal{T}_{k-1}(x)$ and then for every leaf $x_i \in T_{k-1}(x)$, we sample c independent points according to $\mathrm{N}_{1-2\delta}(x_i)$, and make them the children of x_i. (Note the close similarity between these trees and the trees discussed in Section 3.2. The difference is that the trees of Section 3.2 correspond to random walks that are constructed to go downward while now the random walks corresponding to the noise process $\mathrm{N}_{1-2\delta}(\cdot)$ do not have this constraint.)

Given oracle access to $r : \{0,1\}^n \to \{0,1\}$, we use the tree $T_k(x)$ to compute a guess for the value of $f(x)$, by querying r at the leaves and then using Recursive Majority. In more detail, we define functions $\tilde{r}_0, \tilde{r}_1, \ldots, \tilde{r}_k$ which collectively assign a *guess* for every node in T_k. (In more detail, each $\tilde{r}_i$ is a function from $L(T_k(x), i)$ to $\{0,1\}$, where $L(T_k(x), i)$ is the set of points in $\{0,1\}^n$ that are at the nodes at depth $k-i$ in $T_k(x)$.) For each leaf y, we let $\tilde{r}_0(y) = r(y)$. Once $\tilde{r}_{k-t}$ has been defined for nodes at depth t in $T_k(x)$, given y at depth $t-1$ in $T_k(x)$, we set $\tilde{r}_{k-t+1}(y)$ to be the majority of $\tilde{r}_{k-t}$ at its children. We output $\tilde{r}_k(x)$ as our estimate for $f(x)$.

Local Self-corrector
Input: $x \in \{0,1\}^n$, oracle for $r : \{0,1\}^n \to \{0,1\}$ such that $\delta(r,f) \leq 2^{-d_1 s}$ for some $f \in \mathcal{F}(s,n)$.
Output: $b \in \{0,1\}$ which equals $f(x)$ with probability $1-\epsilon$.

Let $\delta = 1/(20s)$, $c = c(1/4, \epsilon)$, $k \in \mathbb{Z}$.
Sample $T_k \sim \mathcal{T}(k,x)$.
For each leaf $y \in T_k$, query $r(y)$.
For $i = 0$ to k, compute $\tilde{r}_i : L(T_k(x), i) \to \{0,1\}$ as described above.
Output $\tilde{r}_k(x)$.

To analyze the algorithm, for $k \in \mathbb{Z}$ define

$$S_k = \left\{ x \in \{0,1\}^n \text{ such that } \Pr_{T_k(x) \sim \mathcal{T}_k(x)} [\tilde{r}_k(x) \neq f(x)] > \epsilon \right\}.$$

The following is analogous to Lemma 27:

Lemma 29. *For $k \geq 1$ and $\epsilon < 1/25$, we have $S_k \subseteq \Lambda_{\delta,1/10}(S_{k-1})$.*

Proof: We have $\tilde{r}_k(x) = \mathrm{Maj}_{1 \leq i \leq c}(b_i)$ where each b_i is drawn independently according to $\tilde{r}_{k-1}(\mathrm{N}_{1-2\delta}(x))$. If $x \in S_k$, then by our choice of $c = c(1/4, \epsilon)$,

$$\Pr_{\substack{y \sim \mathrm{N}_{1-2\delta}(x) \\ T_{k-1}(y) \sim \mathcal{T}_{k-1}(y)}} [\tilde{r}_{k-1}(y) \neq f(x)] > \frac{1}{4}.$$

On the other hand, we also have

$$\Pr_{\substack{y \sim \mathrm{N}_{1-2\delta}(x) \\ T_{k-1}(y) \sim \mathcal{T}_{k-1}(y)}} [\tilde{r}_{k-1}(y) \neq f(x)] \leq \Pr_{\substack{y \sim \mathrm{N}_{1-2\delta}(x) \\ T_{k-1}(y) \sim \mathcal{T}_{k-1}(y)}} [f(y) \neq f(x)] + \Pr_{\substack{y \sim \mathrm{N}_{1-2\delta}(x) \\ T_{k-1}(y) \sim \mathcal{T}_{k-1}(y)}} [\tilde{r}_{k-1}(y) \neq f(y)].$$

The first term on the LHS is bounded by $1/10$ by Lemma 15. Hence we have

$$\Pr_{\substack{y\sim N_\delta(x)\\ T_{k-1}(y)\sim\mathcal{T}_{k-1}(y)}}[\tilde{r}_{k-1}(y)\neq f(y)]\geq\frac{1}{4}-\frac{1}{10}>\frac{1}{7}.$$

But by the definition of S_{k-1},

$$\begin{aligned}\Pr_{\substack{y\sim \mathrm{N}_{1-2\delta}(x)\\ T_{k-1}(y)\sim\mathcal{T}_{k-1}(y)}}[\tilde{r}_{k-1}(y)\neq f(y)]&\leq\epsilon\cdot\Pr_{y\sim\mathrm{N}_{1-2\delta}(x)}[y\notin S_{k-1}]+\Pr_{y\sim\mathrm{N}_{1-2\delta}(x)}[y\in S_{k-1}]\\&\leq\epsilon+\Pr_{y\sim\mathrm{N}_{1-2\delta}(x)}[y\in S_{k-1}].\end{aligned}$$

Hence for $\epsilon<1/25$,

$$\Pr_{y\sim\mathrm{N}_{1-2\delta}(x)}[y\in S_{k-1}]\geq\frac{1}{7}-\epsilon\geq\frac{1}{10},$$

so by the definition of $\Lambda_{\delta,\theta}(S)$ we have $x\in\Lambda_{\delta,1/10}(S_{k-1})$. ∎

We can now analyze our local self-corrector.

Theorem 30. *There exist constants d_1,d_2 such that if $\delta(r,f)\leq 2^{-d_1 s}$ for some $f\in\mathcal{F}(s,n)$, then for $k\geq d_2 s\log(n/s)$ we have that $\tilde{r}_k(x)=f(x)$ with probability 0.99. The algorithm queries the oracle for r at $(n/s)^{O(s)}$ points.*

Proof: Let $\delta=1/(20s)$. Let $d_1>0$ be such that

$$2^{-d_1 s}<(0.1)^{4+60s}.$$

Assume there exists $f\in\mathcal{F}(s,n)$ such that

$$\delta(r,f)\leq 2^{-d_1 s}.$$

Observe that $\tilde{r}_0(x)=r(x)$, so consequently $\mu(S_0)=\delta(r,f)$. By Lemma 29 and Corollary 26, we have

$$\mu(S_k)\leq\mu(\Lambda_{\delta,1/10}(S_{k-1}))\leq\mu(S_{k-1})^{1+\delta}\leq\mu(S_0)^{(1+\delta)^k}.$$

For $k\geq d_2'\ln(n/s)/\delta=d_2 s\log(n/s)$, we have $\mu(S_t)<2^{-n}$, so S_t must be the empty set. But this implies that $\tilde{r}_k(x)=f(x)$ except with probability ϵ.

The number of queries to the oracle is bounded by the number of leaves in the tree, which is c^k. Setting $\epsilon=1/100$, since $c(1/4,1/100)=O(1)$, this is at most $c^k=(n/s)^{O(s)}$. We can amplify the success probability to $1-\epsilon$ using $c(1/100,\epsilon)=O(\log(1/\epsilon))$ independent repetitions. ∎

Discussion. Every real polynomial of degree d computing a Boolean function is also a degree d polynomial over $\mathbb{F}_2$. Hence, it has a natural self-corrector which queries the value at a random affine subspace of dimension $d+1$ containing x, and then outputs the XOR of those values. Conjecture 2 implies that this self-corrector also works for low sensitivity functions. The parameters one would get are incomparable to Theorem 6; we find it interesting that this natural self-corrector is very different from the algorithm of Theorem 6.

We further remark that every Boolean function with real polynomial degree $\deg(f)\leq d$ satsifies $s(f)\leq O(d^2)$ (recall Theorem 7). Thus, Theorem 6 gives a self-corrector for functions with $\deg(f)\leq d$ that has query complexity $n^{O(d^2)}$. It is interesting to note (by considering the example of parity), that this performance guarantee does not extend to all functions with $\mathbb{F}_2$ degree $\deg_2(f)\leq d$.

5 Propagation rules

We have seen that low-degree functions and low-sensitivity functions share the property that they are uniquely specified by their values on small-radius Hamming balls. In either case, we can use these values over a small Hamming ball to infer the values at other points in $\{0,1\}^n$ using simple "local propagation" rules. The propagation rules for the two types of functions are quite different, but Conjecture 2 and its converse given by Theorem 7 together imply that the two rules must converge beyond a certain radius. In this section, we discuss this as a possible approach to Conjecture 2 and some questions that arise from it.

5.1 Low sensitivity functions: the Majority Rule

If $f : \{0,1\}^n \to \{0,1\}$ has sensitivity s, Theorem 10 implies that given the values of f on a ball of radius $2s$, we can recover f at points at distance $r+1 \geq 2s+1$ from the center by taking the Majority value over its neighbors at distance r (see Equation (1)). It is worth noting that as r gets large, the Majority is increasingly lopsided: at most s out of r points are in the minority. We refer to the process of inferring f's values everywhere from its values on a ball via the Majority rule, increasing the distance from the center by one at a time, as "applying the Majority rule".

For concreteness, let us conder the ball centered at 0^n. If there exists a sensitivity s function $f : \{0,1\}^n \to \{0,1\}$ such that the points in $\mathcal{B}(0^n, 2s)$ are labelled according to f, then applying the Majority rule recovers f. However, not every labeling of $\mathcal{B}(0^n, 2s)$ will extend to a low sensitivity function on $\{0,1\}^n$ via the Majority Rule. It is an interesting question to characterize such labelings; progress here will likely lead to progress on Question 14. An obvious necessary condition is that every point in $\mathcal{B}(0^n, 2s)$ should have sensitivity at most s, but this is not sufficient. This can be seen by considering the DNF version of the "tribes" function, where there are n/s disjoint tribes, each tribe is of size s, and $n > s^2$. (So this function f is an (n/s)-way OR of s-way ANDs over disjoint sets of variables.) Every $x \in \mathcal{B}(0^n, 2s)$ has $s(f,x) \leq s$ — in fact, this is true for every $x \in \mathcal{B}(0^n, s(s-1))$ — but it can be verified that applying the Majority rule starting from the ball of radius $2s$ does recover the Tribes function, which has sensitivity $n/s > s$. Another natural question is whether there is a nice characterization of the class of functions that can be obtained by applying the majority rule to a labeling of $\mathcal{B}(0^n, 2s)$.

5.2 Low degree functions: the Parity Rule

It is well known that all functions $f : \{0,1\}^n \to \mathbb{R}$ with $\deg(f) \leq d$ are uniquely specified by their values on a ball of radius d. This follows from the Möbius inversion formula. Again, let us fix the center to be 0^n for concreteness. Letting $\mathbb{1}(T)$ denote the indicator vector of the set T, the formula (see e.g. Section 2.1 of [Juk12]) states that

$$f(x) = \sum_{S \subseteq [n]} c_S \prod_{i \in S} x_i \text{ where } c_S = \sum_{T \subseteq S} (-1)^{|S|-|T|} f(\mathbb{1}(T)). \tag{2}$$

From this it can be inferred that if $\deg(f) \leq d$, then for $|S| \geq d+1$, we have

$$f(\mathbb{1}(S)) = \sum_{T \subset S} (-1)^{|S|-|T|+1} f(\mathbb{1}(T)). \tag{3}$$

We will refer to Equation (3) as the "Parity rule", since it states that f is uncorrelated with the parity of the variables in S on the subcube given by $\{\mathbb{1}(T) : T \subseteq S\}$. We refer to the process of inferring f's values

everywhere from its values on a ball of radius d via the Parity rule, increasing the distance from the center by one at a time, as "applying the Parity rule".

Given a (partial) function $f : \mathcal{B}(0^n, d) \to \{0,1\}$, applying the Parity rule starting from the values of f on $\mathcal{B}(0^n, d)$ lets us extend f to all of $\{0,1\}^n$. Note that the resulting total function f is guaranteed to have degree at most d, but it is not guaranteed to be Boolean-valued everywhere on $\{0,1\}^n$. Indeed, an easy counting argument (see e.g. Lemma 31 of [MORS07]) shows that there are at most $2^{d^2 2^{2d}} \cdot \binom{n}{d2^d}$ degree-d functions over $\{0,1\}^n$, whereas the number of partial functions $f : \mathcal{B}(0^n, d) \to \{0,1\}$ is $2^{\binom{n}{\leq d}}$. It is an interesting question to characterize the set of partial functions $f : \mathcal{B}(0^n, d) \to \{0,1\}$ whose extension by the Parity rule is a Boolean function.

On the other hand, every partial function $f : \mathcal{B}(0^n, d) \to \{0,1\}$ can be uniquely extended to a total function $f : \{0,1\}^n \to \{0,1\}$ such that $\deg_2(f) = d$. This follows from the Mobius inversion formula for multilinear polynomials over $\mathbb{F}_2$:

$$f(x) = \sum_{S \subseteq [n]} c_S \prod_{i \in S} x_i \text{ where } c_S = \sum_{T \subseteq S} f(\mathbb{1}(T)) \tag{4}$$

where the sums are modulo 2. If $\deg_2(f) \leq d$, then $c_S = 0$ for all S where $|S| \geq d+1$. Hence by Equation (2), for $|S| \geq d+1$, we have the simple rule

$$f(\mathbb{1}(S)) = \sum_{T \subset S} f(\mathbb{1}(T)). \tag{5}$$

We can view this as a propagation rule for functions with $\deg_2(f) \leq d$, which extends a labeling of the ball $\mathcal{B}(0^n, d)$ to the entire cube $\{0,1\}^n$. If we start from a labeling of the ball which corresponds to a function $f : \{0,1\}^n \to \{0,1\}$ with $\deg(f) \leq d$, then Equation (5) above coincides with the Parity rule.

5.3 When do the rules work?

Given a partial function $g : \mathcal{B}(x_0, r) \to \{0,1\}$, we can extend it to a total function $g^{\text{Maj}} : \{0,1\}^n \to \{0,1\}$ by applying the Majority rule (if there is not a clear majority among the neighbors queried, the value is underetmined). We can also extend it to a total function $g^{\text{Par}} : \{0,1\}^n \to \mathbb{R}$ using the Parity rule. Given a function $f : \{0,1\}^n \to \{0,1\}$, and a center x_0, we define a series of partial functions $f|_{\mathcal{B}(x_0,r)}$ obtained by restricting f to the ball of radius r around x_0. We are interested in how large r needs to be for the Parity and Majority rules to return the function f for every choice of center x_0 . Formally, we define the following quantities.

Definition 31. *Let $r^{\text{Par}}(f)$ be the smallest r such that for every $x_0 \in \{0,1\}^n$, the Parity rule applied to $\mathcal{B}(x_0, r)$ returns the function f. Formally,*

$$r^{\text{Par}}(f) = \min\{r : \forall x_0 \in \{0,1\}^n,\ (f|_{\mathcal{B}(x_0,r)})^{\text{Par}} = f\}.$$

Similarly, let $r^{\text{Maj}}(f)$ be the smallest r such that for every $x_0 \in \{0,1\}^n$, the Majority rule applied to $\mathcal{B}(x_0, r)$ returns the function f. Formally,

$$r^{\text{Maj}}(f) = \min\{r : \forall x_0 \in \{0,1\}^n (f|_{\mathcal{B}(x_0,r)})^{\text{Maj}} = f\}.$$

It is easy to see that r^{Par} captures the real degree of f:

Lemma 32. *For all $f : \{0,1\}^n \to \{0,1\}$, we have $r^{\mathrm{Par}}(f) = \deg(f)$.*

Proof: The inequality $r^{\mathrm{Par}}(f) \leq \deg(f)$ follows from the fact that the Parity rule correctly extends degree d functions starting from any Hamming ball of radius d.

On the other hand for any center x_0, running the Parity rule on $f|_{\mathcal{B}(x_0,r)}$ for some $r < \deg(f)$ results in a function $(f|_{\mathcal{B}(x_0,r)})^{\mathrm{Par}}$ of degree at most r, since the Parity rule explicitly sets the coefficients of monomials of degree higher than r to 0. But then it follows that $(f|_{\mathcal{B}(x_0,r)})^{\mathrm{Par}} \neq f$, since their difference is a non-zero multilinear polynomial. ∎

The proof above shows that quantifying over x_0 is not necessary in the definition of $r^{\mathrm{Par}}(f)$, since for every $x_0 \in \{0,1\}^n$, we have

$$r^{\mathrm{Par}}(f) = \min\{r : \ (f|_{\mathcal{B}(x_0,r)})^{\mathrm{Par}} = f\}.$$

We now turn to the Majority rule.

Lemma 33. *For all $f : \{0,1\}^n \to \{0,1\}$, we have $r^{\mathrm{Maj}}(f) = \min(2s(f), n)$.*

Proof: We have $r^{\mathrm{Maj}}(f) \leq n$, since $\mathcal{B}(x_0, n)$ is the entire Hamming cube. The upper bound $r^{\mathrm{Maj}}(f) \leq 2s(f)$ follows from the definiton of the Majority rule and Theorem 10.

For the second part, we show that for every $r < \min(2s(f), n)$, there exists a center x_0 such that $(f|_{\mathcal{B}(x_0,r)})^{\mathrm{Maj}} \neq f$. Let x be a point with sensitivity $s(f)$, and let $S \subset [n]$ be the set of $s(f)$ sensitive coordinates at x. We will pick x_0 so that $d(x, x_0) = r + 1$ as follows. If $r + 1 \leq s(f)$, we obtain x_0 from x by flipping some $r + 1$ coordinates from S. If $r + 1 > s(f)$, then we obtained x_0 from f by flipping all the coordinates in S, and any $r + 1 - s(f)$ other coordinates $T \subseteq [n] \setminus S$. The condition $r + 1 \leq n$ guarantees that a subset of the desired size exists, while $r + 1 \leq 2s(f)$ enures that $|T| \leq |S|$.

Since $d(x, x_0) = r + 1$, the value at x is inferred using the Majority rule applied to the neighbors of x in $\mathcal{B}(x_0, r)$. These neighbors are obtained by either flipping coordinates in S or T (where T might be empty). The former disagree with $f(x)$ while the latter agree. Since $|S| \geq |T|$, the Majority rule either labels x wrongly, or leaves it undetermined (in the case when $r = 2s(f)$). This shows that $(f|_{\mathcal{B}(x_0,r)})^{\mathrm{Maj}} \neq f(x)$, hence $r^{\mathrm{Maj}} \geq \min(2s(f), n)$. ∎

In contrast with Lemma 32, quantifying over all centers x_0 in the defintion of r^{Par} is crucial for the lower bound in Lemma 33. This is seen by considering the n-variable OR function, where the sensitivity is n. Applying the Majority rule to a ball of radius 2 around 0^n returns the right function, but if we center the ball at 1^n, then the Majority rule cannot correctly infer the value at 0^n, so it needs to be part of the advice, hence $r^{\mathrm{Maj}}(\mathrm{OR}) = n$.

5.4 Agreement of the Majority and Parity Rule

Lemmas 32 and 33 can be viewed as alternate characterizations of the degree and sensitivity of a Boolean function. The degree versus sensitivity conjecture asserts that both these rules work well (meaning that they only require the values on a small ball as advice) for the same class of functions. Given that the rules are so simple, and seem so different from each other, we find this assertion surprising.

In particular, Conjecture 2 is equivalent to the following statement:

Conjecture 34. *There exists constants d_1, d_2 such that*

$$r^{\mathrm{Par}}(f) \leq d_1(r^{\mathrm{Maj}})^{d_2}. \tag{6}$$

Along similar lines, one can use Theorem 7, due to [NS94], to show that the Majority rule recovers low-degree Boolean functions:

$$r^{\mathrm{Maj}}(f) \leq 8(r^{\mathrm{Par}}(f))^2. \tag{7}$$

Their proof uses Markov's inequality from analysis. It might be interesting to find a different proof, which one could hope to extend to proving Equation (6) as well.

6 Conclusion (and more open problems)

We have presented the first upper bounds on the computational complexity of low sensitivity functions. We believe this might be a promising alternative approach to Conjecture 1 as opposed to getting improved bounds on specific low level measures like block sensitivity or decision tree depth [KK04, ABG+14, AS11].

Conjecture 1 implies much stronger upper bounds than are given by our results. We list some of the ones which might be more approachable given our results:

1. Every sensitivity s function has a TC_0 circuit of size $n^{O(s)}$.
2. Every sensitivity s function has a polynomial threshold function (PTF) of degree $\mathrm{poly}(s)$.
3. Every sensitivity s function f has $\deg_2(f) \leq s^c$ for some constant c.

References

[ABG+14] Andris Ambainis, Mohammad Bavarian, Yihan Gao, Jieming Mao, Xiaoming Sun, and Song Zuo. Tighter relations between sensitivity and other complexity measures. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014*, pages 101–113, 2014. 1.3, 6

[AP14] Andris Ambainis and Krisjanis Prusis. A tight lower bound on certificate complexity in terms of block sensitivity and sensitivity. In *MFCS*, pages 33–44, 2014. 1.3

[APV15] Andris Ambainis, Krisjanis Prusis, and Jevgenijs Vihrovs. Sensitivity versus certificate complexity of boolean functions. *CoRR*, abs/1503.07691, 2015. 1.3

[AS11] Andris Ambainis and Xiaoming Sun. New separation between $s(f)$ and $bs(f)$. *CoRR*, abs/1108.3494, 2011. 1.3, 6

[AV15] Andris Ambainis and Jevgenijs Vihrovs. Size of sets with small sensitivity: a generalization of simon's lemma. In *Theory and Applications of Models of Computation - 12th Annual Conference, TAMC 2015*, pages 122–133, 2015. 1.3

[BdW02] H. Buhrman and R. de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288(1):21–43, 2002. 1.1

[CDR86] Stephen A. Cook, Cynthia Dwork, and Rüdiger Reischuk. Upper and lower time bounds for parallel random access machines without simultaneous writes. *SIAM J. Comput.*, 15(1):87–97, 1986. 1.3

[GKS15] Justin Gilmer, Michal Koucký, and Michael E. Saks. A new approach to the sensitivity conjecture. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015*, pages 247–254, 2015. 1.3

[HKP11] Pooya Hatami, Raghav Kulkarni, and Denis Pankratov. *Variations on the Sensitivity Conjecture*. Number 4 in Graduate Surveys. Theory of Computing Library, 2011. 1.1, 1.3

[Jac30] Dunham Jackson. The theory of approximation. *New York*, 19:30, 1930. 1.1

[Juk12] S. Jukna. *Boolean Function Complexity: Advances and Frontiers*. Springer, 2012. 5.2

[KK04] Claire Kenyon and Samuel Kutin. Sensitivity, block sensitivity, and l-block sensitivity of Boolean functions. *Information and Computation*, pages 43–53, 2004. 1.3, 6

[MORS07] K. Matulef, R. O'Donnell, R. Rubinfeld, and R. Servedio. Testing Halfspaces. Technical Report 128, Electronic Colloquium in Computational Complexity, 2007. Full version in FOCS 2007. 5.2

[Nis91] Noam Nisan. Crew prams and decision trees. *SIAM Journal on Computing*, 20(6):999–1007, 1991. 1.1, 1, 1.1, 2

[NS64] DJ Newman and HS Shapiro. Jackson's theorem in higher dimensions. In *On Approximation Theory/Über Approximationstheorie*, pages 208–219. Springer, 1964. 1.1

[NS94] N. Nisan and M. Szegedy. On the degree of Boolean functions as real polynomials. *Comput. Complexity*, 4:301–313, 1994. 1.1, 1, 2, 1.3, 7, 5.4

[O'D14] Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014. 1.2, 4

[Rub95] David Rubinstein. Sensitivity vs. block sensitivity of Boolean functions. *Combinatorica*, pages 297–299, 1995. 1.3

[Sim82] Hans-Ulrich Simon. A tight omega(log log n)-bound on the time for parallel ram's to compute nondegenerated boolean functions. *Information and Control*, 55(1-3):102–106, 1982. 1.2, 1.2, 1.3, 8

[Wei85] Karl Weierstrass. Über die analytische darstellbarkeit sogenannter willkürlicher functionen einer reellen veränderlichen. *Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften zu Berlin*, 2:633–639, 1885. 1.1

A Omitted Proofs and Results

Proof of Lemma 18: It suffices to prove the bound for s_1. The proof is by induction on the dimension n. Observe that if $\mu_1(f) = 1$ then the claim is trivial, so we may assume $\mu_1 \in (0, 1)$. In the base case $n = 1$, we must have $\mu_1 = 1/2$, in which case $s(f) = 1$ so the claim holds.

For any $i \in [n]$, let $f_{i,1} = f|_{x_i=1}$ and $f_{i,0} = f|_{x_i=0}$ denote the restrictions of f to the subcubes defined by x_i. These are each functions on variables in $[n] \setminus \{i\}$. Then

$$\mu_1(f) = \frac{\mu_1(f_{i,1}) + \mu_1(f_{i,0})}{2}.$$

If there exists $b \in \{0,1\}$ such that $0 < \mu_1(f_{i,b}) \leq \mu_1(f)$ then we can apply the inductive claim to the restricted function $f_{i,b}$ to conclude that there exists a point $x \in f_{i,b}^{-1}(1)$ so that

$$s(f,x) \geq \log\left(\frac{1}{\mu_1(f_{i,b})}\right) \geq \log\left(\frac{1}{\mu_1(f)}\right).$$

If not, it must be that $f(x) = 1$ implies $x_i = b$ for some $b \in \{0,1\}$, so that

$$\mu_1(f_{i,b}) = 2\mu_1(f) \text{ and } \mu_1(f_{i,1-b}) = 0.$$

But then every point $x \in f^{-1}(1)$ is sensitive to x_i. Further, we can apply the inductive hypothesis to $f_{i,b}$, to conclude that there exists $x \in f_{i,b}^{-1}(1)$ such that x is sensitive to

$$\log\left(\frac{1}{\mu_1(f_{i,b})}\right) = \log\left(\frac{1}{2\mu_1(f)}\right) = \log\left(\frac{1}{\mu_1(f)}\right) - 1$$

coordinates from $[n] \setminus \{i\}$. Since x is also sensitive to i, we have

$$s_1(f,x) \geq \log\left(\frac{1}{\mu_1(f)}\right).$$

For the final claim, assume the above bound holds with equality. Then there do not exist $i \in [n], b \in \{0,1\}$ such that $0 < \mu_1(f_{i,b}) < \mu_1(f)$ (if they did exist then we would get a stronger bound). So for every i, either $\mu_1(f_{i,b}) = 0$ for some b, or $\mu_1(f_{i,0}) = \mu_1(f_{i,1})$. In the former case, the set $f^{-1}(1)$ is contained in the subcube $x_i = b$. In the latter case, by induction we may assume that $f^{-1}(1)$ restricted to both $x_i = 0$ and $x_1 = 1$ is a subcube of density exactly $\mu_1(f)$ in $\{0,1\}^{[n]\setminus\{i\}}$, so every point in these subcubes must have sensitivity $\log(1/\mu_1(f))$ We further claim that the two subcubes are identical as functions on $\{0,1\}^{[n]\setminus\{i\}}$. If they were not identical, then some point (in each subcube) would be sensitive to coordinate i, but then this point would have sensitivity at least $\log(1/\mu_1(f)) + 1$.

This implies that $f^{-1}(1)$ is a subcube defined by the equations $x_i = 1 - b$ for all pairs (i,b) such that $\mu_1(f_{i,b}) = 0$. □

A.1 A Top-Down Algorithm

Next we describe a "top-down" algorithm for computing $f(x)$ where f is a function of sensitivity s. This algorithm has a similar performance bound to our "bottom-up" algorithm described earlier.

Associate the bit string $x \in \{0,1\}^n$ with the integer $z(x) = \sum_{i=1}^n x_i 2^i$, and let $x < x'$ if $z(x) < z(x')$. We refer to this as the colex ordering on strings.

The top-down algorithm also takes the values of f on $\mathcal{B}(0^n, 2s)$ as advice. Given an input $x \in \{0,1\}^n$ where we wish to evaluate f, we recursively evaluate f at the first $2s+1$ neighbors of x of Hamming weight $\mathrm{wt}(x)-1$ in the colex order. The recursion bottoms out when we reach an input of weight $2s$. The restriction to small elements in the colex order ensures that the entire set of inputs on which we need to evaluate f is small. A detailed description of the algorithm follows:

Top-Down
Advice: f at all points in $\mathcal{B}(0^n, 2s)$.
Input: $x \in \{0,1\}^n$.

1. If $\mathrm{wt}(x) \leq 2s$ or if $f(x)$ has been computed before, return $f(x)$.
2. Otherwise, let $x_1, \ldots, x_{2s+1}$ be the $2s + 1$ smallest elements in $N(x)$ of weight $\mathrm{wt}(x) - 1$ in the colex order. If some $f(x_i)$ has not been computed yet, compute it recursively and store the value.
3. Return $f(x) = \mathrm{Maj}_{i \in [2s+1]}(f(x_i))$.

The key to the analysis is the following lemma.

Lemma 35. *Let* $\mathrm{wt}(x) = d$. *For* $2s \leq k \leq d$, *the number of weight* k *vectors* x' *for which* $f(x')$ *is computed by the* **Top-Down** *algorithm is bounded by*

$$\binom{d-k+2s}{d-k} \leq d^{2s}.$$

Proof: Given $x \in \{0,1\}^n$, for $t \leq \mathrm{wt}(x)$, let $R(t) \subseteq [n]$ denote the t largest indices $i \in [n]$ where $x_i = 1$. We claim that all vectors x' with $\mathrm{wt}(x') = k$ that are generated by the algorithm are obtained by setting $d - k$ indices in $R(d - k + 2s)$ to 0. This claim clearly implies the desired bound.

The claim is proved by induction on $d - k$. The case $d - k = 1$ is easy to see, since the $2s + 1$ smallest neighbors of x in the colex order (of weight one less than x) are each obtained by setting one of the indices in $R(2s + 1)$ to 0. For the inductive case, assume that $\mathrm{wt}(y) = k$, and that y is generated as a neighbor of y' with $\mathrm{wt}(y') = k + 1$. Inductively, y' is obtained from x by setting indices in $S \subset R(d - k - 1 + 2s)$ to 0, where $|S| = d - k - 1$, and hence leaving $2s$ of them 1. Thus the $2s$ smallest neighbors of y' are obtained by setting indices in $R(d - k - 1 + 2s) \setminus S$ to 0, and the $(2s + 1)$th neighbor is obtained by setting the $(d - k + 2s)$th 1 from the right to 0. In both cases, we get $d - k$ indices from $R(d - k + 2s)$ being set to 0. This completes the induction. ∎

Theorem 36. *The* **Top-Down** *algorithm computes* $f(x)$ *for any input* x *in time* $O(sn^{2s+1})$ *using space* $O(n^{2s+1})$.

Proof: By the preceding lemma, for an input x of weight d, the total number of x' for which $f(x')$ is computed and stored is bounded by

$$\sum_{k=2s}^{d} d^{2s} \leq d^{2s+1} \leq n^{2s+1}.$$

The cost of computing f at x given f's values at the relevant $2s + 1$ neighbors of x (see Step 3) is $O(s)$, so on average the amortized cost for computing $f(x)$ at each x is bounded by $O(s)$. Hence overall the running time and space are bounded by $O(sn^{2s+1})$ and $O(n^{2s+1})$ respectively. ∎

ECCC ISSN 1433-8092
http://eccc.hpi-web.de