Testing k-Monotonicity

Clément L. Canonne[1], Elena Grigorescu[2], Siyao Guo[3], Akash Kumar[4], and Karl Wimmer[5]

1 **Columbia University, New York, USA**
ccanonne@cs.columbia.edu
2 **Purdue University, West Lafayette, USA**
elena-g@purdue.edu
3 **New York University, New York, USA**
sg191@nyu.edu
4 **Purdue University, West Lafayette, USA**
akumar@purdue.edu
5 **Duquesne University, Pittsburgh USA**
wimmerk@duq.edu

Abstract

A Boolean *k-monotone* function defined over a finite poset domain $\mathcal{D}$ alternates between the values 0 and 1 at most k times on any ascending chain in $\mathcal{D}$. Therefore, k-monotone functions are natural generalizations of the classical *monotone* functions, which are the 1-*monotone* functions.

Motivated by the recent interest in k-monotone functions in the context of circuit complexity and learning theory, and by the central role that monotonicity testing plays in the context of property testing, we initiate a systematic study of k-monotone functions, in the property testing model. In this model, the goal is to distinguish functions that are k-monotone (or are close to being k-monotone) from functions that are far from being k-monotone.

Our results include the following:

1. We demonstrate a separation between testing k-monotonicity and testing monotonicity, on the hypercube domain $\{0,1\}^d$, for $k \geq 3$;
2. We demonstrate a separation between testing and learning on $\{0,1\}^d$, for $k = \omega(\log d)$: testing k-monotonicity can be performed with $2^{O(\sqrt{d}\cdot\log d\cdot\log 1/\varepsilon)}$ queries, while learning k-monotone functions requires $2^{\Omega(k\cdot\sqrt{d}\cdot 1/\varepsilon)}$ queries (Blais et al. (RANDOM 2015)).
3. We present a tolerant test for functions $f\colon [n]^d \to \{0,1\}$ with complexity independent of n, which makes progress on a problem left open by Berman et al. (STOC 2014).

Our techniques exploit the testing-by-learning paradigm, use novel applications of Fourier analysis on the grid $[n]^d$, and draw connections to distribution testing techniques.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Boolean Functions, Learning, Monotonicity, Property Testing

Digital Object Identifier 10.4230/LIPIcs.ITCS.2017.29

1 Introduction

A function $f\colon \mathcal{D} \to \{0,1\}$, defined over a finite partially ordered domain $(\mathcal{D}, \preceq)$ is said to be *k-monotone*, for some integer $k \geq 0$, if there does not exist $x_1 \preceq x_2 \preceq \ldots \preceq x_{k+1}$ in $\mathcal{D}$ such that $f(x_1) = 1$ and $f(x_i) \neq f(x_{i+1})$ for all $i \in [k]$. Note that 1-monotone functions are the classical *monotone* functions, satisfying $f(x_1) \leq f(x_2)$, whenever $x_1 \preceq x_2$.

Monotone functions have been well-studied on multiple fronts in computational complexity due to their natural structure. They have been celebrated for decades in the property

testing literature [25, 20, 24, 10, 15, 14, 16], where we have recently witnessed ultimate results [33, 18, 6], in the circuit complexity literature, where we now have strong lower bounds [43, 44], and in computational learning, where we now have learning algorithms in numerous learning models [11, 3, 32, 46, 40, 41].

The generalized notion of k-monotonicity has also been studied in the context of circuit lower bounds for more than 50 years. In particular, Markov [36] showed that any k-monotone function (even with multiple outputs) can be computed using circuits containing only $\log k$ negation gates. The presence of negation gates appears to be a challenge in proving circuit lower bounds: "the effect of such gates on circuit size remains to a large extent a mystery" [29]. The recent results of Blais *et al.* [9] on circuit lower bounds have prompted renewed interest in understanding k-monotone functions from multiple angles, including cryptography, circuit complexity, learning theory, and Fourier analysis ([45, 27, 26, 35]).

Motivated by the exponential lower bounds on PAC learning k-monotone functions due to [9], we initiate the study of k-monotonicity in the closely related *Property Testing* model. In this model, given query access to a function, one must decide if the function is k-monotone, or is far from being k-monotone, by querying the input only in a small number of places.

1.1 Our results

We focus on testing k-monotonicity of Boolean functions defined over the d-dimensional hypegrid $[n]^d$, and the hypercube $\{0,1\}^d$. We begin our presentation with the results for the hypercube, in order to build intuition into the difficulty of the problem, while comparing our results with the current literature on testing monotonicity. Our stronger results concern the hypegrid $[n]^d$.

1.1.1 Testing k-monotonicity on the hypercube $\{0,1\}^d$

In light of the recent results of [33] that provide a $O(\sqrt{d})$-query tester for monotonicity, we first show that testing k-monotonicity is strictly harder than testing monotonicity on $\{0,1\}^d$, for $k \geq 3$.

▶ **Theorem 1.** *For $1 \leq k \leq d^{1/4}/2$, any one-sided non-adaptive tester for k-monotonicity of functions $f\colon \{0,1\}^d \to \{0,1\}$ must make $\Omega\big(d/k^2\big)^{k/4}$ queries.*

Both Theorem 1 and its proof generalize the $\Omega(d^{1/2})$ lower bound for testing monotonicity, due to Fischer *et al.* [24].

On the upper bounds side, while the monotonicity testing problem is providing numerous potential techniques for approaching this new problem [25, 20, 14, 10, 19, 33], most common techniques appear to resist generalizations to k-monotonicity. However, our upper bounds demonstrate a separation between testing and PAC learning k-monotonicity, for large enough values of $k = \omega(\log d)$.

▶ **Theorem 2.** *There exists a one-sided non-adaptive tester for k-monotonicity of functions $f\colon \{0,1\}^d \to \{0,1\}$ with query complexity $q(d,\varepsilon,k) = 2^{O\left(\sqrt{d}\cdot\log d\cdot\log\frac{1}{\varepsilon}\right)}$.*

Indeed, in the related PAC learning model, [9] shows that learning k-monotone functions on the hypercube requires $2^{\Omega(k\cdot\sqrt{d}\cdot 1/\varepsilon)}$ many queries.

We further observe that the recent non-adaptive and adaptive 2-sided lower bounds of [18, 6], imply the same bounds for k-monotonicity, using black box reductions. We summarize the state of the art for testing k-monotonicity on the hypercube in Table 1.

Table 1 Testing k-monotonicity of a function $f\colon \{0,1\}^d \to \{0,1\}$

	upper bound	1.s.-n.a. lower bound	2.s.-n.a. lower bound	2.s.-a. lower bound
$k=1$	$O\left(\sqrt{d}\right)$ [33]	$\Omega\left(d^{1/2}\right)$ [24]	$\Omega\left(d^{1/2-o(1)}\right)$ [18]	$\tilde{\Omega}\left(d^{1/4}\right)$ [6]
$k \geq 2$	$d^{O(k\sqrt{d})}$ [9], $d^{O(\sqrt{d})}$ Thm 2	$\Omega\left(d/k^2\right)^{k/4}$ Thm 1 $(k = O(d^{1/4}))$	$\Omega\left(d^{1/2-o(1)}\right)$	$\tilde{\Omega}\left(d^{1/4}\right)$

Table 2 Summary of our results: testing k-monotonicity of a function $f\colon [n]^d \to \{0,1\}$ (first two columns). The last column contains known bounds on monotonicity testing and is provided for comparison.

	General k	$k=2$	$k=1$ (monotonicity)
$d=1$	$\Theta\left(\frac{k}{\varepsilon}\right)$ 1.s.-n.a., $\tilde{O}\left(\frac{1}{\varepsilon^7}\right)$ 2.s.-n.a.	$O\left(\frac{1}{\varepsilon}\right)$ 1.s.-n.a.	$\Theta\left(\frac{1}{\varepsilon}\right)$ 1.s.-n.a.
$d=2$	$\tilde{O}\left(\frac{k^2}{\varepsilon^3}\right)$ 2.s.-n.a. (from below)	$\Theta\left(\frac{1}{\varepsilon}\right)$ 2.s.-a.	$\Theta\left(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\right)$ 1.s.-n.a., $\Theta\left(\frac{1}{\varepsilon}\right)$ 1.s.-a.
$d \geq 3$	$\tilde{O}\left(\frac{1}{\varepsilon^2}\left(\frac{5kd}{\varepsilon}\right)^d\right)$ 2.s.-n.a., $2^{\tilde{O}(k\sqrt{d}/\varepsilon^2)}$ 2.s.-n.a.	$\tilde{O}\left(\frac{1}{\varepsilon^2}\left(\frac{10d}{\varepsilon}\right)^d\right)$ 2.s.-n.a., $2^{\tilde{O}(\sqrt{d}/\varepsilon^2)}$ 2.s.-n.a.	$O\left(\frac{d}{\varepsilon}\log\frac{d}{\varepsilon}\right)$ 1.s.-n.a.

1.1.2 Testing k-monotonicity on the hypergrid $[n]^d$

The remainder of the paper focuses on functions defined over the d-dimensional hypergrid domain $[n]^d$, where we denote by $(i_1, i_2, \ldots, i_d) \preceq (j_1, j_2, \ldots, j_d)$ the partial order in which $i_1 \leq j_1, i_2 \leq j_2, \ldots, i_d \leq j_d$. Testing monotonicity has received a lot of attention over the d-dimensional hypergrids [25, 21, 23, 5, 1, 28, 8, 15, 14, 16, 7], where the problem is well-understood, and we refer the reader the appendix of the full version for a detailed review on the state of the art in the area. We summarize our results on testing k-monotonicity over $[n]^d$ in Table 2.

1.1.2.1 Testing k-monotonicity on the line and the 2-dimensional grid

We begin with a study of functions $f\colon [n] \to \{0,1\}$. As before, note that 1-sided tests should always accept k-monotone functions, and so, they must accept unless they discover a violation to k-monotonicity in the form of a sequence $x_1 \preceq x_2 \preceq \ldots \preceq x_{k+1}$ in $[n]^d$, such that $f(x_1) = 1$ and $f(x_i) \neq f(x_{i+1})$. Therefore, lower bounds for 1-sided k-monotonicity testing must grow at least linearly with k. We show that this is indeed the case for both adaptive and non-adaptive tests, and moreover, we give a tight non-adaptive algorithm. Consequently, our results demonstrate that adaptivity does not help in testing k-monotonicity with one-sided error on the line domain.

▶ **Theorem 3.** *Any one-sided (possibly adaptive) tester for k-monotonicity of functions $f\colon [n] \to \{0,1\}$ must have query complexity $\Omega\left(\frac{k}{\varepsilon}\right)$.*

The upper bound generalizes the $O(1/\varepsilon)$ tester for monotonicity on the line.

▶ **Theorem 4.** *There exists a one-sided non-adaptive tester for k-monotonicity of functions $f\colon [n] \to \{0,1\}$ with query complexity $q(n, \varepsilon, k) = O\left(\frac{k}{\varepsilon}\right)$.*

Testing with 2-sided error, however, does not require a dependence on k. In fact the problem has been well-studied in the machine learning literature in the context of testing/learning "union of intervals" [31, 4], and in testing geometric properties, in the

context of testing surface area [34, 38],[1] resulting in an $O(1/\varepsilon^{7/2})$-query algorithm. Namely, the starting point of [4] (later improved by [34]) is a "Buffon Needle's"-type argument, where the crucial quantity to analyze is the noise sensitivity of the function, that is the probability that a randomly chosen pair of nearby points cross a "boundary" – i.e., have different values. (Moreover, the algorithm of [4] works in the *active testing* setting: it only requires a weaker access model that the standard query model).

We provide an alternate proof of a poly$(1/\varepsilon)$ bound (albeit with a worse exponent) that reveals a surprising connection with *distribution testing*, namely with the problem of estimating the support size of a distribution.

▶ **Theorem 5.** *There exists a two-sided non-adaptive tester for k-monotonicity of functions $f\colon [n] \to \{0,1\}$ with query complexity $q(n,\varepsilon,k) = \tilde{O}\big(1/\varepsilon^7\big)$,* independent of k.

An immediate implication of Theorem 5 is that one can test even $n^{1-\alpha}$-monotonicity of $f\colon [n] \to \{0,1\}$, for every $\alpha > 0$, with a constant number of queries. Hence, there is a separation between 1-sided and 2-sided testing, for $k = \omega(1)$.

Turning to the 2-dimensional grid, we show that 2-monotone functions can be tested with the minimum number of queries one could hope for:

▶ **Theorem 6.** *There exists a two-sided adaptive tester for 2-monotonicity of functions $f\colon [n]^2 \to \{0,1\}$ with query complexity $q(n,\varepsilon) = O\big(\frac{1}{\varepsilon}\big)$.*

We also discuss possible generalizations of Theorem 6 to general k or d in the full version.

1.1.2.2 Testing k-monotonicity on $[n]^d$, tolerant testing, and distance approximation

Moving to the general grid domain $[n]^d$, we show that k-monotonicity is testable with poly$(1/\varepsilon, k)$ queries in constant-dimension grids.

▶ **Theorem 7.** *There exists a non-adaptive tester for k-monotonicity of functions $f\colon [n]^d \to \{0,1\}$ with query complexity $q(n,d,\varepsilon,k) = \min(\tilde{O}\Big(\frac{1}{\varepsilon^2}\left(\frac{5kd}{\varepsilon}\right)^d\Big), 2^{\tilde{O}\left(k\sqrt{d}/\varepsilon^2\right)})$.*

In fact, we obtain more general testing algorithms than in Theorem 7, namely our results hold for *tolerant* testers (as we define next).

The notion of tolerant testing was first introduced in [42] to account for the possibility of noisy data. In this notion, a test should accept inputs that are ε_1-close to the property, and reject inputs that are ε_2-far from the property, where ε_1 and ε_2 are given parameters. Tolerant testing is intimately connected to the notion of distance approximation: given tolerant testers for every $(\varepsilon_1, \varepsilon_2)$, there exists an algorithm that estimates the distance to the property within any (additive) ε, while incurring only a $\tilde{O}\big(\log\frac{1}{\varepsilon}\big)$ factor blow up in the number of queries. Furthermore, [42] shows that both tolerant testing and distance approximation are no harder than agnostic learning. We prove the following general result.

▶ **Theorem 8.** *There exist*

- *a non-adaptive (fully) tolerant tester for k-monotonicity of functions $f\colon [n]^d \to \{0,1\}$ with query complexity $q(n,d,\varepsilon_1,\varepsilon_2,k) = \tilde{O}\left(\frac{1}{(\varepsilon_2-\varepsilon_1)^2}\left(\frac{5kd}{\varepsilon_2-\varepsilon_1}\right)^d\right)$;*
- *a non-adaptive tolerant tester for k-monotonicity of functions $f\colon [n]^d \to \{0,1\}$ with query complexity $q(n,d,\varepsilon_1,\varepsilon_2,k) = 2^{\tilde{O}\left(k\sqrt{d}/(\varepsilon_2-3\varepsilon_1)^2\right)}$, under the restriction that $\varepsilon_2 > 3\varepsilon_1$.*

[1] We thank Eric Blais for mentioning the connection, and pointing us to these works.

To the best of our knowledge, the only previous results for tolerant testing for monotonicity on $[n]^d$ are due to Fattal and Ron [22]. They give both additive and multiplicative distance approximations algorithms, and obtain $O(d)$-multiplicative and ε-additive approximations with query complexity poly$(\frac{1}{\varepsilon})$. While very efficient, there results only give fully tolerant testers for dimensions $d = 1$ and $d = 2$. Our results generalize the work of [22] showing existence of tolerant testers for k-monotonicity (and hence for monotonicity) for any dimension $d \geq 1$, and any $k \geq 1$, but paying the price in the query complexity.

As a consequence to Theorem 8, we make progress on an open problem of Berman *et al.* [7], as explained next.

1.1.2.3 Testing k-monotonicity under L_p distance

The property of being a monotone Boolean function has a natural extension to real-valued functions. Indeed, a real-valued function defined over a finite domain D is monotone if $f(x) \leq f(y)$ whenever $x \preceq y$. For real-valued functions the more natural notion of distance is L_p distance, rather than Hamming distance. The study of monotonicity has been extended to real-valued functions in a recent work by Berman *et al.* [7]. They give tolerant testers for grids of dimension $d = 1$ and $d = 2$, and leave open the problem of extending the results to general d, as asked explicitly at the recent Sublinear Algorithms Workshop 2016 [47].

We make progress towards solving this open problem, by combining our Theorem 8 with a reduction from L_p testing to Hamming testing inspired by [7]. This reduction relates L_1-distance of a function $f\colon [n]^d \to [0,1]$ to monotonicity to *Hamming* distance to monotonicity of a "rounded" function $\tilde{f}\colon [n]^d \times [m] \to \{0,1\}$, essentially trading the range for an extra dimension (where m is a rounding parameter to be suitably chosen). Moreover, simulating query access to $\tilde{f}$ can be performed efficiently given query access to f.

▶ **Theorem 9.** *There exists a non-adaptive tolerant L_1-tester for monotonicity of functions $f\colon [n]^d \to [0,1]$ with query complexity*

- $\tilde{O}\left(\frac{1}{(\varepsilon_2-\varepsilon_1)^2}\left(\frac{5d}{\varepsilon_2-\varepsilon_1}\right)^d\right)$*, for any* $0 \leq \varepsilon_1 < \varepsilon_2 \leq 1$*;*
- $2^{\tilde{O}\left(\sqrt{d}/(\varepsilon_2-3\varepsilon_1)^2\right)}$*, for any* $0 \leq 3\varepsilon_1 < \varepsilon_2 \leq 1$*.*

1.2 Proofs overview and technical contribution

Structural properties and the separation between testing and learning on $\{0,1\}^d$

We first observe that basic structural properties, such as *extendability* (i.e. the feature that a function that is monotone on a sub-poset of $[n]^d$ can be extended into a monotone function on the entire poset domain), and properties of the *violation graph* (i.e., the graph whose edges encode the violations to monotonicity), extend easily to k-monotonicity (see the full version of the paper). These properties help us to argue the separation between testing and learning (Theorem 2). However, unlike the case of monotonicity testing, these properties do not seem to be enough for showing upper bounds that grow polynomially in d.

Grid coarsening and testing by implicit/explicit learning

One pervading technique, which underlies all the hypergrid upper bounds in this work, is that of *gridding*: i.e., partitioning the domain into "blocks" whose size no longer depends on the main parameter of the problem, n. This technique generalizes the approach of [22] who performed a similar gridding for dimension $d = 2$. By simulating query access to the "coarsened" version of the unknown function (with regard to these blocks), we are able to

leverage methods such as testing-by-learning (either fully or partially learning the function), or reduce our testing problem to a (related) question on these nicer "coarsenings." (The main challenge here lies in providing efficient and consistent oracle access to the said coarsenings.)

At a high-level, the key aspect of k-monotonicity which makes this general approach possible is reminiscent of the concept of *heredity* in property testing. Specifically, we rely upon the fact that "gridding preserves k-monotonicity:" if f *is* k-monotone, then so will be its coarsening g – but now g is much simpler to handle. This allows us to trade the domain $[n]^d$ for what is effectively $[m]^d$, with $m \ll n$. We point out that this differs from the usual paradigm of *dimension reduction*: indeed, the latter would reduce the study of a property of functions on $[n]^d$ to that of functions on $[n]^{d'}$ for $d' \ll d$ (usually even $d' = 1$) by projecting f on a lower-dimensional domain. In contrast, we do not take the dimension down, but instead reduce the size of the *alphabet*.Moreover, it is worth noting that this gridding technique is also orthogonal to that of *range reduction*, as used e.g. in [20]. Indeed, the latter is a reduction of the range of the function from $[R]$ to $\{0, 1\}$, while gridding is only concerned about the domain size.

Estimating the support of distributions

Our proof of the poly$(1/\varepsilon)$ upper bound for testing k-monotonicity on the line (Theorem 5) rests upon an unexpected connection to *distribution testing*, namely to the question of support size estimation of a probability distribution. In more detail, we describe how to reduce k-monotonicity testing to the support size estimation problem in (a slight modification of) the *Dual access model* introduced by Canonne and Rubinfeld [13], where the tester is granted samples from an unknown distribution as well as query access to its probability mass function.

For our reduction to go through, we first describe how any function $f\colon [n] \to \{0, 1\}$ determines a probability distribution D_f (on $[n]$), whose effective support size is directly related to the k-monotonicity of f. We then show how to implement dual access to this D_f from queries to f: in order to avoid any dependence on k and n in this step, we resort both to the gridding approach outlined above (allowing us to remove n from the picture) and to a careful argument to "cap" the values of D_f returned by our simulated oracle. Indeed, obtaining the exact value of $D_f(x)$ for arbitrary x may require $\Omega(k)$ queries to f, which we cannot afford; instead, we argue that only returning $D_f(x)$ whenever this value is "small enough" is sufficient. Finally, we show that implementing this "capped" dual access oracle is possible with no dependence on k whatsoever, and we can now invoke the support size estimation algorithm of [13] to conclude.

Fourier analysis on the hypergrid

We give an algorithm for fully tolerantly testing k-monotonicity whose query complexity in exponential in d. We also describe an alternate tester (with a slightly worse tolerance guarantee) whose query complexity is instead exponential in $\widetilde{O}(k\sqrt{d})$ for constant distance parameters. As mentioned above, we use our gridding approach combined with tools from learning theory. Specifically, we employ an agnostic learning algorithm of [30] using polynomial regression. Our coarsening methods allow us to treat the domain as if it were $[m]^d$ for some m that is independent of n. To prove that this agnostic learning algorithm will succeed, we turn to Fourier analysis over $[m]^d$. We extend the bound on average sensitivity of k-monotone functions over the Boolean hypercube from [9] to the hypergrid, and we show that this result implies that the Fourier coefficients are concentrated on "simple" functions.

1.3 Discussion and open problems

This is the first work to study k-monotonicity, a natural and well-motivated generalization of monotonicity. Hence this work opens up many intriguing questions in the area of property testing, with potential applications to learning theory, circuit complexity and cryptography.

As previously mentioned, the main open problem prompted by our work is the following:

Can k-monotonicity on the hypercube $\{0,1\}^d$ be tested with $\mathrm{poly}(d^k)$ queries?

A natural 1-sided tester for k-monotonicity is a *chain tester*: it queries points along a random chain, and rejects only if it finds a violation to k-monotonicity, in the form of a sequence $x_1 \preceq x_2 \preceq \ldots \preceq x_{k+1}$ in $\{0,1\}^d$, such that $f(x_1) = 1$ and $f(x_i) \neq f(x_{i+1})$. In particular, the testers in [25, 14, 19, 33] all directly imply a chain tester. We conjecture that there exists a chain tester for k-monotonicity that succeeds with probability $d^{-O(k)}$.

Another important open question concerns the hypergrid domain, and in particular it pushes for a significant strengthening of Theorem 7 and Theorem 9:

Can k-monotonicity on the hypergrid $[n]^d$ be (tolerantly) tested with $2^{o_k(\sqrt{d})}$ queries?

Answering this question would imply further progress on the L_1-testing question for monotonicity, left open in [7, 47].

There also remains the question of establishing two-sided lower bounds that would go beyond those of monotonicity. Specifically:

Is there an $d^{\Omega(k)}$-query two-sided *lower bound for k-monotonicity on the hypercube $\{0,1\}^d$?*

In this work we also show surprising connections to distribution testing (e.g. in the proof of Theorem 5), and to testing union of intervals and testing surface area. An intriguing direction is to generalize this connection to union of intervals and surface area in higher dimensions, to leverage or gain insight on k-monotonicity on the d-dimensional hypergrid.

Finally, while we only stated here a few directions, we emphasize that every question that is relevant to monotonicity is also relevant and interesting in the case of k-monotonicity.

1.4 Related work

As mentioned, k-monotonicity has deep connections with the notion of *negation complexity* of functions, which is the minimum number of negation gates needed in a circuit to compute a given function. The power of negation gates is intriguing and far from being understood in the context of circuit lower bounds. Quoting from Jukna's book [29], *the main difficulty in proving nontrivial lower bounds on the size of circuits using AND, OR, and NOT is the presence of NOT gates: we already know how to prove even exponential lower bounds for monotone functions if no NOT gates are allowed. The effect of such gates on circuit size remains to a large extent a mystery.*

This gap has motivated the study of circuits with *few* negations. Two notable works successfully extend lower bounds in the monotone setting to negation-limited setting: in [2], Amano and Maruoka show superpolynomial circuit lower bounds for $(1/6)\log\log n$ negations using the Clique function; and recently the breakthrough work of Rossman [45] establishes circuit lower bounds for NC^1 with roughly $\frac{1}{2}\log n$ negations by drawing upon his lower bound for monotone NC^1.

The divide between the understanding of monotone and non-monotone computation exists in general: while we usually have a fairly good understanding of the monotone case, many things get murky or fail to hold even when a single negation gate is allowed. In order to get a better grasp on negation-limited circuits, a body of recent work has been considering this model in various contexts: Blais *et al.* [9] study negation-limited circuits from a computational learning viewpoint, Guo *et al.* [27] study the possibility of implementing cryptographic primitives using few negations, and Lin and Zhang [35] are interested in verifying whether some classic Boolean function conjectures hold for the subset of functions computed by negation-limited circuits.

Many of these results implicitly or explicitly rely on a simple but powerful tool: the decomposition of negation-limited circuits into a composition of some "nice" function with monotone components. Doing so enables one to apply results on separate monotone components, and finally to carefully combine the outcomes (e.g., [26]). Though these techniques can yield results for as many as $O(\log n)$ negations, they also leave open surprisingly basic questions:

- [9] Can we have an efficient weak learning algorithm for functions computed by circuits with a *single* negation?
- [27] Can we obtain pseudorandom generators when allowing only a *single* negation?

In contexts where the circuit size is not the quantity of interest, the equivalent notion of 2-monotone functions is more natural than that of circuits allowing only one negation. Albeit seemingly simple, even the class of 2-monotone functions remains largely a mystery: as exemplified above, many basic yet non-trivial questions, ranging from the structure of their Fourier spectrum to their expressive power of k-monotone functions, remain open.

1.5 Organization of the paper

After recalling some notations and definitions in section 2, we consider the case of functions on the line in section 3, focusing on the proof of the two-sided upper bound of Theorem 3.

In section 4 we present our general algorithms for k-monotonicity on the hypergrid $[n]^d$, for arbitrary k and d. We prove Theorem 8 in two parts. We establish its first item (general tolerant testing algorithm with exponential dependence in d) in subsection 4.1 (Proposition 22). The second item (with query complexity exponential in $k\sqrt{d}$) is proven in subsection 4.2, where we analyze the Fourier-based tolerant tester of Proposition 31.

Our results on the Boolean hypercube, the two-dimensional grid, as well as some structural results and applications to tolerant L_1-testing of monotonicity have been left out of this short version, and are deferred to the full version of the paper [12].

2 Preliminaries

We denote by log the binary logarithm, and use $\tilde{O}(\cdot)$ to hide polylogarithmic factors in the argument (so that $\tilde{O}(f) = O(f \log^c f)$ for some $c \geq 0$).

Given two functions $f, g\colon \mathcal{X} \to \mathcal{Y}$ on a finite domain $\mathcal{X}$, we write $\mathrm{dist}(f,g)$ for the (normalized) Hamming distance between them, i.e.

$$\mathrm{dist}(f,g) = \frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}} \mathbb{1}_{\{f(x)\neq g(x)\}} = \Pr_{x\sim\mathcal{X}}[\,f(x)\neq g(x)\,]$$

where $x \sim \mathcal{X}$ refers to x being drawn from the uniform distribution on $\mathcal{X}$. A *property* of functions from $\mathcal{X}$ to $\mathcal{Y}$ is a subset $\mathcal{P} \subseteq \mathcal{X}^{\mathcal{Y}}$ of these functions; we define the distance of a function f to $\mathcal{P}$ as the minimum distance of f to any $g \in \mathcal{P}$:

$$\mathrm{dist}(f,\mathcal{P}) = \inf_{g\in\mathcal{P}} \mathrm{dist}(f,g)\,.$$

For some of our applications, we will also use another notion of distance specific to real-valued functions, the L_1 distance (as introduced in the context of property testing in [7]). For $f, g\colon \mathcal{X} \to [0,1]$, we write

$$L_1(f,g) = \frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}} |f(x)-g(x)| = \mathbb{E}_{x\sim\mathcal{X}}[|f(x)-g(x)|] \in [0,1]$$

and extend the definition to $L_1(f,\mathcal{P})$, for $\mathcal{P} \subseteq \mathcal{X}^{[0,1]}$, as before.

Property testing

We recall the standard definition of testing algorithms, as well as some terminology:

▶ **Definition 10.** Let $\mathcal{P}$ be a property of functions from $\mathcal{X}$ to $\mathcal{Y}$. A *q-query testing algorithm for $\mathcal{P}$* is a randomized algorithm $\mathcal{T}$ which takes as input $\varepsilon \in (0,1]$ as well as query access to a function $f\colon \mathcal{X} \to \mathcal{Y}$. After making at most $q(\varepsilon)$ queries to the function, $\mathcal{T}$ either outputs ACCEPT or REJECT, such that the following holds:

- if $f \in \mathcal{P}$, then $\mathcal{T}$ outputs ACCEPT with probability at least 2/3; (Completeness)
- if $\mathrm{dist}(f,\mathcal{P}) \geq \varepsilon$, then $\mathcal{T}$ outputs REJECT with probability at least 2/3; (Soundness)

where the probability is taken over the algorithm's randomness. If the algorithm only errs in the second case but accepts any function $f \in \mathcal{P}$ with probability 1, it is said to be a *one-sided* tester; otherwise, it is said to be *two-sided*. Moreover, if the queries made to the function can only depend on the internal randomness of the algorithm, but not on the values obtained during previous queries, it is said to be *non-adaptive*; otherwise, it is *adaptive*.

Additionally, we will also be interested in *tolerant* testers – roughly, algorithms robust to a relaxation of the first item above:

▶ **Definition 11.** Let $\mathcal{P}$, $\mathcal{X}$, and $\mathcal{Y}$ be as above. A *q-query tolerant testing algorithm for $\mathcal{P}$* is a randomized algorithm $\mathcal{T}$ which takes as input $0 \leq \varepsilon_1 < \varepsilon_2 \leq 1$, as well as query access to a function $f\colon \mathcal{X} \to \mathcal{Y}$. After making at most $q(\varepsilon_1,\varepsilon_2)$ calls to the oracle, $\mathcal{T}$ outputs either ACCEPT or REJECT, such that the following holds:

- if $\mathrm{dist}(f,\mathcal{P}) \leq \varepsilon_1$, then $\mathcal{T}$ outputs ACCEPT with probability at least 2/3; (Completeness)
- if $\mathrm{dist}(f,\mathcal{P}) \geq \varepsilon_2$, then $\mathcal{T}$ outputs REJECT with probability at least 2/3; (Soundness)

where the probability is taken over the algorithm's randomness. The notions of one-sidedness and adaptivity of Theorem 10 extend to tolerant testing algorithms as well.

Note that as stated, in both cases the algorithm "knows" $\mathcal{X}, \mathcal{Y}$, and $\mathcal{P}$; so that the query complexity q can be parameterized by these quantities. More specifically, when considering $\mathcal{X} = [n]^d$ and the property $\mathcal{P}$ of k-monotonicity, we will allow q to depend on n, d, and k. Finally, we shall sometimes require a probability of success $1-\delta$ instead of the (arbitrary) constant 2/3; by standard techniques, this can be obtained at the cost of a multiplicative $O(\log(1/\delta))$ in the query complexity.

PAC and agnostic learning [48]

A learning algorithm $\mathcal{A}$ for a *concept class* $\mathcal{C}$ of functions $f\colon \mathcal{X} \to \mathcal{Y}$ (under the uniform distribution) is given parameters $\varepsilon, \delta > 0$ and sample access to some target function $f \in \mathcal{C}$ *via* labeled samples $\langle x, f(x)\rangle$, where x is drawn uniformly at random from $\mathcal{X}$. The algorithm should output a hypothesis $h\colon \mathcal{X} \to \mathcal{Y}$ such that $\operatorname{dist}(h, f) \leq \varepsilon$ with probability at least $1-\delta$. The algorithm is *efficient* if it runs in time $\operatorname{poly}(n, 1/\varepsilon, 1/\delta)$. If $\mathcal{A}$ must output $h \in \mathcal{C}$ we say it is a *proper learning algorithm*, otherwise, we say it is an *improper learning* one.

Moreover, if $\mathcal{A}$ still succeeds when f does not actually belong to $\mathcal{C}$, we say it is an *agnostic learning algorithm*. Specifically, the hypothesis function h that it outputs must satisfy $\operatorname{dist}(f, g) \leq \textsc{opt}_f + \varepsilon$ with probability at least $1-\delta$, where $\textsc{opt}_f = \min_{g\in\mathcal{C}} \operatorname{dist}(f, g)$.

3 On the line

In this section we focus on testing k-monotonicity on the line, that is of functions $f\colon [n] \to \{0,1\}$. Our results include Theorem 4, which establishes that this can be done non-adaptively with one-sided error, with only $O(k/\varepsilon)$ queries; complemented by Theorem 3, which shows that this is the best one can hope for if we insist on one-sidedness. Due to space constraints, we only prove here Theorem 5 (restated below), which shows that – perhaps unexpectedly – *two-sided* algorithms, even non-adaptive, can break this barrier and test k-mononicity with *no* dependence on k. The proofs of Theorem 4 and Theorem 3 can be found in the full version.

▶ **Theorem 5.** *There exists a two-sided non-adaptive tester for k-monotonicity of functions $f\colon [n] \to \{0,1\}$ with query complexity $q(n, \varepsilon, k) = \tilde{O}\big(1/\varepsilon^7\big)$, independent of k.*

In what follows, we assume that $k > 20/\varepsilon$, as otherwise we can use for instance the $O(k/\varepsilon)$-query (non-adaptive, one-sided) tester of Theorem 4 to obtain an $O\big(1/\varepsilon^2\big)$ query complexity.

3.1 Testing k-monotonicity over $[Ck]$

In this section, we give a $\operatorname{poly}(C/\varepsilon)$-query tester for k-monotonicity over the domain $[Ck]$, where C is a parameter to be chosen (for our applications, we will eventually set $C = \operatorname{poly}(1/\varepsilon)$).

▶ **Lemma 12.** *There exists a two-sided non-adaptive tester for k-monotonicity of functions $f\colon [Ck] \to \{0,1\}$ with query complexity $O\Big(\frac{C^3}{\varepsilon^3}\Big)$.*

The tester proceeds by reducing to support size estimation and using (a slight variant of) an algorithm of Canonne and Rubinfeld [13]. Let $f\colon [Ck] \to \{0,1\}$, and suppose f is s-monotone but not $(s-1)$-monotone. Then there is a unique partition of $[Ck]$ into $s+1$ disjoint intervals $I_1, I_2, \ldots, I_{s+1}$ such that f is constant on each interval; note that this constant value alternates in consecutive intervals. We can then define a distribution D_f over $[s+1]$ such that $D_f(i) = |I_i|/(Ck)$.

Our next claims, Claim 13 and Claim 14, provide the basis for the reduction (from testing k-monotonicity of f to support size estimation of D_f).

▶ **Claim 13.** *If f is ε-far from k-monotone, then it is not $(1+\frac{\varepsilon}{4})k$-monotone, and in particular $|\operatorname{supp}(D_f)| > (1+\frac{\varepsilon}{4})k + 1$.*

▶ **Claim 14.** *To ε-test k-monotonicity of f, it suffices to estimate $|\operatorname{supp}(D_f)|$ to within $\frac{\varepsilon k}{10}$.*

The proofs of above claims are relatively straightforward. So we defer these proofs to the end of this section and now proceed to use algorithm of [13] to do support size estimation. The algorithm of [13] uses "dual access" to D; an oracle that provides a random sample from D, and an oracle that given an element of D, returns the probability mass assigned to this element by D.

▶ **Theorem 15** ([13, Theorem 14 (rephrased)]). *In the access model described above, there exists an algorithm that, on input a threshold $n \in \mathbb{N}^*$ and a parameter $\varepsilon > 0$, and given access to a distribution D (over an arbitrary set) satisfying $\min_{x \in \mathrm{supp}(D)} D(x) \geq \frac{1}{n}$ estimates the support size $|\mathrm{supp}(D)|$ up to an additive εn, with query complexity $O\left(\frac{1}{\varepsilon^2}\right)$.*

Note however that we only have access to D_f through query access to f, and thus have to manage to simulate (efficiently) access to the former. One difficulty is that, to access $D_f(i)$, we need to determine where I_i lies in f. For example, finding $D_f(k/2)$ requires finding $I_{k/2}$, which might require a large number of queries to f. We circumvent this by weakening the "dual access" model in two ways, arguing for each of these two relaxations that the algorithm of [13] can still be applied:

- we rewrite the support size as in [13], as $|\mathrm{supp}(D_f)| = \mathbb{E}_{x \sim D_f}[1/D_f(x)]$. We want to estimate it to within $\pm O(\varepsilon k)$ which we can do by random sampling;
- the quantity inside the expectation depends on $D_f(x)$ but not x itself, so "labels" are unnecessary for our random sampling. Thus, it will be sufficient to be able to compute $D_f(x)$ (and thus $1/D_f(x)$) for a random $x \sim D_f$, even if not actually knowing x itself;
- actually, even calculating $D_f(x)$ may possibly too expensive, so instead we will estimate $\mathbb{E}_{x \sim D_f}[1/\tilde{D}_f(x)]$ where $\tilde{D}_f(x) = \min(\frac{20}{\varepsilon k}, D_f(x))$. Note that $\tilde{D}_f$ might no longer define a probability distribution; but this expectation is only off by at most $\frac{\varepsilon k}{20}$, since $1/D'_f(x) = \max(\varepsilon k/20, 1/D_f(x))$ and $1/D_f(x)$ is positive.

More details follow.

First, we note as discussed above that the algorithm does not require knowing the "label" of any element in the support of the distribution: the only access required is being able to randomly sample elements according to D_f, and evaluate the probability mass on the sampled points. This, in turn, can be done, as the following two lemmas explain:

▶ **Lemma 16** (Sampling from D_f). *Let $i \in [n]$ be chosen uniformly at random, and let j be such that $i \in I_j$. Then, the distribution of j is exactly D_f.*

▶ **Lemma 17** (Evaluating $D_f(j)$). *Suppose $I_j = \{a, a+1, \ldots, b\}$. Given i such that $i \in I_j$, we can find I_j by querying $f(i+1) = f(i+2) = \cdots = f(b)$ and $f(b+1) \neq f(b)$, as well as $f(i-1) = f(i-2) = \ldots = f(a)$ and $f(a-1) \neq f(a)$. The number of queries to f is $b - a + 3 = |I_j| + 3$.*

Here comes the second difficulty: if we straightforwardly use these approaches to emulate the required oracles to estimate the support size of D_f, the number of queries is potentially very large. For instance, if we attempt to query $D_f(j)$ where $|I_j| = \Omega(k)$, we will need $\Omega(k)$ queries to f. This is where comes the second relaxation: specifically, we shall argue that it will be enough for us to "cap" the size of the interval (as per our next lemma).

▶ **Lemma 18** (Evaluating $D_f(j)$ with a cap). *Given i such that $i \in I_j$, we will query f on every point in $[i - 20C/\varepsilon, i + 20C/\varepsilon]$. If $|I_j| \leq 20C/\varepsilon$, then I_j will be determined by these queries. If these queries do not determine I_j, we know $|I_j| > 20C/\varepsilon$. Beyond querying i, this requires $40C/\varepsilon$ (nonadaptive) queries.*

We now can put all the above pieces together and give the proof for Lemma 12:

Proof of Theorem 12: As previously discussed, we use the algorithm of [13] for estimating support size. Inspecting their algorithm, we see that our cap of $20C/\varepsilon$ for interval length (and therefore $20/(\varepsilon k)$ for maximum probability reported) might result in further error of the estimate. The algorithm interacts with the unknown function by estimating the expected value of $1/D_f(j)$ over random choices of j with respect to D_f. Our cap can only decrease this expectation by at most $(\varepsilon k)/20$. Indeed, the algorithm works by estimating the quantity $\mathbb{E}_{x\sim D_f}[\frac{1}{D_f(x)}\mathbb{1}_{\{D_f(x)>\tau\}}]$, for some suitable parameter $\tau > 0$. By capping the value of $1/D_f(x)$ to $20/(\varepsilon k)$, we can therefore only decrease the estimate, and by at most $20/(\varepsilon k) \cdot D_f(\{\ x :\ D_f(x) > (\varepsilon k)/20\ \}) \le 20/(\varepsilon k)$.

The condition for their algorithm to estimate support size to within $\pm\varepsilon m$ is that all elements in the support have a probability mass of at least $1/m$. Since each nonempty interval has length at least 1, we have $\min_j D_f(j) \ge (1/Ck)$. In order for their algorithm to report an estimate within $\pm\varepsilon k/20$ of support size, we set $\varepsilon' = (\varepsilon/20C)$ in their algorithm.

The total error in support size is at most $\varepsilon k/20 + \varepsilon k/20 = \varepsilon k/10$. By Claim 14, this suffices to test ε-test k-monotonicity of f.

Using the algorithm of [13], we need $O(1/{\varepsilon'}^2) = O\big((C/\varepsilon)^2\big)$ queries to D_f. For every query to D_f, we need to make $O(C/\varepsilon)$ queries to f, so the overall query complexity is $O\big(C^3/\varepsilon^3\big)$. ◀

Proof of Claim 13. The last part of the statement is immediate from the first, so it suffices to prove the first implication. We show the contrapositive: assuming f is $(1+\frac{\varepsilon}{4})k$-monotone, we will "fix" it into a k-monotone function by changing at most εn points. In what follows, we assume $\frac{\varepsilon k}{4} \ge 1$, as otherwise the statement is trivial (any function that is ε-far from k-monotone is *a fortiori* not k-monotone).

Let as before ℓ^* be the minimum integer ℓ for which f is ℓ-monotone: we can assume $k < \ell^* \le (1+\frac{\varepsilon}{4})k$ (as if $\ell^* \le k$ we are done.) Consider as above the maximal consecutive monochromatic intervals $I_1, \ldots, I_{\ell^*}$, and let i be the index of the shortest one. In particular, it must be the case that $|I_i| \le \frac{n}{\ell^*+1}$. Flipping the value of f on I_i therefore has "cost" at most $\frac{n}{\ell^*+1}$, and the resulting function f' is now exactly (ℓ^*-2)-monotone if $1 < i < \ell^*$, and (ℓ^*-1)-monotone if $i \in \{1, \ell^*\}$. This means in particular that repeating the above $\frac{\varepsilon}{4}k$ times is enough to obtain a k-monotone function, and the total cost is upperbounded by

$$\sum_{j=0}^{\varepsilon k/4} \frac{n}{\ell^*+1-2j} \le \sum_{j=0}^{\varepsilon k/4} \frac{n}{k+1-2j} = \sum_{j=k(1-\frac{\varepsilon}{2})+1}^{k+1} \frac{n}{j} \le n\frac{\frac{\varepsilon}{2}k+1}{(1-\frac{\varepsilon}{4})k+1} \le n\frac{\frac{3\varepsilon}{4}k}{(1-\frac{\varepsilon}{4})k}$$

where for the last inequality (for the numerator) we used that $1 \le \frac{\varepsilon k}{4}$. But this last RHS is upperbounded by εn (as $\frac{3}{4}x \le x(1-\frac{1}{4}x)$ for $x \in [0,1]$), showing that therefore, f was ε-close to k-monotone to begin with, which is a contradiction. ◀

Proof of Claim 14. If f is ε-far from k-monotone, then $|\mathrm{supp}(D_f)| > (1+\frac{\varepsilon}{4})k = k + \frac{\varepsilon}{4}k$, and if f is k-monotone, then $|\mathrm{supp}(D_f)| \le k+1$. The fact that $k > 20/\varepsilon$ then allows us to conclude. ◀

3.2 Reducing $[n] \to \{0,1\}$ to $[Ck] \to \{0,1\}$.

Now we show how to reduce ε-testing k-monotonicity of $f\colon [n] \to \{0,1\}$ to ε'-testing k-monotonicity of a function $g\colon [Ck] \to \{0,1\}$ for $C = \mathrm{poly}(1/\varepsilon)$ and $\varepsilon' = \mathrm{poly}(\varepsilon)$, resulting in a $\mathrm{poly}(1/\varepsilon)$-query algorithm for ε-testing k-monotonicity.

The first step is (as before) to divide $[n]$ into blocks (disjoint intervals) of size $\frac{\varepsilon n}{4k}$ if $\varepsilon > \frac{8k}{n}$ (again assuming without loss of generality that $\frac{\varepsilon n}{4k}$ is an integer), and blocks of size 1 otherwise (in which case $n \leq \frac{8k}{\varepsilon}$ and we can directly apply the result of Theorem 12, with $C = n/k \leq 8/\varepsilon$). Let $m = 4k/\varepsilon$ be the number of resulting blocks, and define $f_m \colon [n] \to \{0,1\}$ as the *m-block-coarsening* of f: namely, for any $j \in B_i$, we set

$$f_m(j) = \text{argmax}_{b \in \{0,1\}} \Pr_{k \in B_i} [f_m(k) = b] \qquad \text{(majority vote)}$$

Ordering the blocks $B_1, B_2, \ldots, B_m$, we also define $g \colon [m] \to \{0,1\}$ such that $g(i) = \min_{a \in B_i} f_m(a)$.

It is easy to see that if f is k-monotone, then f has at most k non-constant blocks, and f_m is k-monotone. Because the function f only changes values k times; for a block to be non-constant, the block must contain a pair of points with a value change. We call a block *variable* if the minority points comprise at least an $\varepsilon/100$-fraction of the block; formally, B is variable if $\min_{b \in \{0,1\}} \Pr_{j \in B}[f(j) = b] \geq \varepsilon/100$.

We need following claims (their proofs are at the end of the section) to prove Theorem 5.

▶ **Claim 19.** *Suppose f has s variable blocks. Then* $\text{dist}(f, f_m) \leq s/m + \varepsilon/100$.

▶ **Claim 20.** *Suppose f is promised to be either (i) k-monotone or (ii) such that f_m has more than $\frac{5}{4}k$ variable blocks. Then we can determine which with $O\big(\frac{1}{\varepsilon^2} \log \frac{1}{\varepsilon}\big)$ queries, and probability* 9/10.

Proof of Theorem 5. We use the estimation/test from the previous claim as the first part of our tester. Assuming f passes, we can assume that f_m has less than $\frac{5}{4}k$ variable blocks. By Claim 19, $\text{dist}(f, f_m) \leq \frac{5k}{4}/m + \frac{\varepsilon}{100} = \frac{5\varepsilon}{32} + \frac{\varepsilon}{100} \leq \frac{\varepsilon}{3}$. This part takes $O\big(\frac{1}{\varepsilon^2} \log \frac{1}{\varepsilon}\big)$ queries.

Now, we apply the tester of Theorem 12 (with probability of success amplified to 9/10 by standard arguments) to $(\varepsilon/6)$-test k-monotonicity of $g \colon [m] \to \{0,1\}$, where $g(i)$ is the constant value of f_m on B_i, and $m = (4k)/\varepsilon$. Let q be the query complexity of the tester, and set $\delta = 1/(10q)$; to query $g(i)$, we randomly query f on $O\big(\frac{1}{\varepsilon} \log \frac{1}{\delta}\big)$ points in B_i and take the majority vote. With probability at least $1 - \delta$, we get the correct value of $g(i)$, and by a union bound all q simulated queries have the correct value with probability at least 9/10.

Therefore, to get a single query to g, we use $O((\log q)/\varepsilon)$ queries. In the context of our previous section, we have $C = 4/\varepsilon$, so $q = O(C^3/\varepsilon^3) = O\big(1/\varepsilon^6\big)$ and the overall query complexity of this part is $O((q \log q)/\varepsilon) = O\big(\frac{1}{\varepsilon^7} \log \frac{1}{\varepsilon}\big)$. This dominates the query complexity of the other part of the tester, from Claim 20, which is $O\big(\frac{1}{\varepsilon^2} \log \frac{1}{\varepsilon}\big)$. By a union bound over the part from Claim 20, the simulation of g, and the call to the tester of Theorem 12, the algorithm is correct with probability at least $1 - 3/10 > 2/3$. ◀

Proof of Claim 19. We will estimate the error of f_m in computing f on variable blocks and non-variable blocks separately. Each non-variable block B can contribute error on at most $\varepsilon |B| /100$ points. Each variable block B can contribute error on at most $|B| = n/m$ points. The total number of errors is at most $\varepsilon n/100 + s(n/m) = n(\varepsilon/100 + s/m)$, yielding the upper bound on $\text{dist}(f, f_m)$. ◀

Proof of Claim 20. We first note that given any fixed block B, it is easy to detect whether it is variable (with probability of failure at most δ) by making $O\big(\frac{1}{\varepsilon} \log \frac{1}{\delta}\big)$ uniformly distributed queries in B. Doing so, a variable block will be labelled as such with probability at least $1 - \delta$, while a constant block will never be marked as variable. (If a block is neither constant nor variable, then any answer will do.)

Letting s denote the number of variable blocks, we then want to non-adaptively distinguish between $s \geq \frac{5}{4}k = \frac{5\varepsilon}{16}m$ and $s \leq k = \frac{\varepsilon}{4}m$ (since if f were k-monotone, then f_m had at most k variable blocks). Doing so with probability at least 19/20 can be done by checking only $q = O\left(\frac{1}{\varepsilon}\right)$ blocks chosen uniformly at random: by the above, setting $\delta = \frac{1}{20q}$ all of the q checks will also yield the correct answer with probability no less than 9/10, so by a union bound we will distinguish (i) and (ii) with probability at least 9/10. We conclude by observing that all $O\left(q \cdot \frac{1}{\varepsilon}\log\frac{1}{q}\right) = O\left(\frac{1}{\varepsilon^2}\log\frac{1}{\varepsilon}\right)$ queries are indeed non-adaptive. ◀

4 On the high-dimensional grid

In this section, we give two algorithms for tolerant testing, that is testing whether a function $f\colon [n]^d \to \{0,1\}$ is ε_1-close to k-monotone vs. ε_2-far from k-monotone, establishing Theorem 8. The first has query complexity exponential in the dimension d and is *fully tolerant*, that is works for any setting of $0 \leq \varepsilon_1 < \varepsilon_2 \leq 1$. The second applies whenever $\varepsilon_2 > 3\varepsilon_1$, and has (incomparable) query complexity exponential in $\tilde{O}(k\sqrt{d}/(\varepsilon_2 - 3\varepsilon_1)^2)$. Both of these algorithms can be used for non-tolerant ("regular") testing by setting $\varepsilon_1 = 0$ and $\varepsilon_2 = \varepsilon$, which implies Theorem 7.

▶ **Theorem 8.** *There exist*

- *a non-adaptive (fully) tolerant tester for k-monotonicity of functions $f\colon [n]^d \to \{0,1\}$ with query complexity $q(n,d,\varepsilon_1,\varepsilon_2,k) = \tilde{O}\left(\frac{1}{(\varepsilon_2-\varepsilon_1)^2}\left(\frac{5kd}{\varepsilon_2-\varepsilon_1}\right)^d\right)$;*
- *a non-adaptive tolerant tester for k-monotonicity of functions $f\colon [n]^d \to \{0,1\}$ with query complexity $q(n,d,\varepsilon_1,\varepsilon_2,k) = 2^{\tilde{O}\left(k\sqrt{d}/(\varepsilon_2-3\varepsilon_1)^2\right)}$, under the restriction that $\varepsilon_2 > 3\varepsilon_1$.*

As a corollary, this implies Theorem 7, restated below:

▶ **Theorem 7.** *There exists a non-adaptive tester for k-monotonicity of functions $f\colon [n]^d \to \{0,1\}$ with query complexity $q(n,d,\varepsilon,k) = \min(\tilde{O}\left(\frac{1}{\varepsilon^2}\left(\frac{5kd}{\varepsilon}\right)^d\right), 2^{\tilde{O}\left(k\sqrt{d}/\varepsilon^2\right)})$.*

For convenience, we will view in this part of the paper the set $[n]$ as $[n] = \{0,1,\ldots,n-1\}$. Assuming that m divides n, we let $\mathcal{B}_{m,n}\colon [n]^d \to [m]^d$ be the mapping such that $\mathcal{B}_{m,n}(y)_i = \lfloor y_i/m \rfloor$ for $1 \leq i \leq m$. For $x \in [m]^d$, we define the set $\mathcal{B}^{-1}_{m,n}(x)$ to be the inverse image of x. Specifically, $\mathcal{B}^{-1}_{m,n}(x)$ is the set of points of the form $m \cdot x + [n/m]^d$, with standard definitions for scalar multiplication and coordinate-wise addition. That is, $\mathcal{B}^{-1}_{m,n}(x)$ is a "coset" of $[n/m]^d$ points in $[n]^d$. To keep on with the notations of the other sections, we will call these cosets *blocks*, and will say a function $h\colon [n]^d \to \{0,1\}$ is an *m-block function* if it is constant on each block. Moreover, for clarity of presentation, we will omit the subscripts on $\mathcal{B}$ and $\mathcal{B}^{-1}$ whenever they are not necessary.

We first establish a lemma that will be useful for the proofs of correctness of both algorithms.

▶ **Lemma 21.** *Suppose $f\colon [n]^d \to \{0,1\}$ is k-monotone. Then there is an m-block function $h\colon [n]^d \to \{0,1\}$ such that $\mathrm{dist}(f,h) < kd/m$.*

Proof. Fix any k-monotone function $f\colon [n]^d \to \{0,1\}$. We partition $[m]^d$ into chains of the form

$$C_x = \left\{ x + \ell \cdot 1^d \,:\, \ell \in \mathbb{N}, x \in [m]^d \text{ and } x_i = 0 \text{ for some } i \right\}.$$

There are $m^d - (m-1)^d \leq dm^{d-1}$ of these chains: we will show that f can only be nonconstant on at most k blocks of each chain.

Algorithm 1 Fully tolerant testing with $O(kd/(\varepsilon_2-\varepsilon_1)))^d$ queries.

Require: Query access to $f\colon [n]^d \to \{0,1\}$, $\varepsilon_2 > \varepsilon_1 \geq 0$, a positive integer k

$\alpha \leftarrow (\varepsilon_2 - \varepsilon_1), m \leftarrow \lceil 5kd/\alpha \rceil, t \leftarrow \lceil 25\ln(6m^d)/(2\alpha^2) \rceil$

▷ Define a distribution D over $[m]^d \times \{0,1\}$.

for $x \in [m]^d$ **do**

  Query f on t random points $T_x \subseteq \mathcal{B}^{-1}(x)$.

  $D(x,0) \leftarrow \Pr_{y\in T_x}[\,f(y)=0\,]/m^d$

  $D(x,1) \leftarrow \Pr_{y\in T_x}[\,f(y)=1\,]/m^d$

end for

▷ Define a distribution D' over $[n]^d \times \{0,1\}$ such that $D'(y,b) = D(\mathcal{B}(y),b)\cdot m^d/n^d$.

if there exists a k-monotone m-block function h such that $\Pr_{(y,b)\sim D'}[\,h(y)\neq b\,] \leq \varepsilon_1 + \frac{\alpha}{2}$ **then return** ACCEPT

end if

return REJECT

By contradiction, suppose there exists $x \in [m]^d$ such that f is nonconstant on $k+1$ different blocks $\mathcal{B}^{-1}(z^{(i)})$, where $z^{(1)} \prec z^{(2)} \prec \ldots \prec z^{(k)} \prec z^{(k+1)}$, and each $z^{(i)} \in C_x$. By construction, we have $\mathcal{B}^{-1}(z^{(i)}) \prec \mathcal{B}^{-1}(z^{(j)})$ for $i<j$. For each $1 \leq i \leq k+1$, there are two points $v_*^{(i)}, v_{**}^{(i)} \in \mathcal{B}^{-1}(z_i)$ such that $v_*^{(i)} \prec v_{**}^{(i)}$ and $f(v_*^{(i)}) \neq f(v_{**}^{(i)})$. By construction $v_*^{(1)} \prec v_{**}^{(1)} \prec v_*^{(2)} \prec v_{**}^{(2)} \prec v_*^{(3)} \prec v_{**}^{(3)} \prec \ldots \prec v_*^{(k+1)} \prec v_{**}^{(k+1)}$, and there must be at least $k+1$ pairs of consecutive points with differing function values. Out of these $2k+2$ many points, there is a chain of points $\bar{v}^{(1)} \prec \bar{v}^{(2)} \prec \ldots \prec \bar{v}^{(k+1)}$ where $f(\bar{v}^{(i)}) \neq f(\bar{v}^{(i+1)})$ for $1 \leq i \leq k$, which is a violation of the k-monotonicity of f.

Thus, in each of the dm^{d-1} many chains of blocks, there can only be k nonconstant blocks. It follows that there are at most kdm^{d-1} nonconstant blocks in total. We now define $h(y)$ to be equal to $f(y)$ if f is constant on $\mathcal{B}(y)$, and arbitrarily set $h(y)=0$ otherwise. Each set $\mathcal{B}^{-1}(y)$ contains $(n/m)^d = n^d \cdot m^{-d}$ many points, and f is not constant on at most kdm^{d-1} of these. It follows that $\mathrm{dist}(f,h) \leq kdm^{d-1}\cdot m^{-d} = kd/m$. ◀

4.1 Fully tolerant testing with $O(kd/(\varepsilon_2-\varepsilon_1)))^d$ queries

Our first algorithm (Algorithm 1) then proceeds by essentially brute-force learning an m-block function close to the unknown function, and establishes the first item of Theorem 8.

▶ **Proposition 22.** *Algorithm 1 accepts all functions ε_1-close to k-monotone functions, and rejects all functions ε_2-far from k-monotone (with probability at least $2/3$). Its query complexity is $O\left(\frac{d}{(\varepsilon_2-\varepsilon_1)^2}\left(\frac{5kd}{\varepsilon_2-\varepsilon_1}+1\right)^d \log\frac{kd}{\varepsilon_2-\varepsilon_1}\right)$.*

Proof. The algorithm first estimates $\Pr_{y\in\mathcal{B}^{-1}(x)}[\,f(y)=b\,]$ for every $x \in [m]^d$ and $b \in \{0,1\}$ to within $\pm\frac{\alpha}{5}$. We use $t = 25\ln(6m^d)/2\alpha^2$ points in each block to ensure (by an additive Chernoff bound) that each estimate is correct except with probability at most $m^{-d}/3$. By a union bound, the probability that all estimates are correct is at least $2/3$, and we hereafter condition on this. By construction, $\mathbb{E}_{(x,b)\sim D}[\Pr_{y\in\mathcal{B}^{-1}(x)}[\,f(y)\neq b\,]] = \Pr_{(y,b)\sim D'}[\,f(y)\neq b\,] \leq \frac{\alpha}{5}$. In this probability experiment, the marginal distribution of D' on y is uniform over $[n]^d$.

Let $f^*\colon [n]^d \to \{0,1\}$ be a k-monotone function minimizing $\Pr[\,f(y)\neq f^*(y)\,]$. Theorem 21 ensures that there is a k-monotone m-block function $h\colon [n]^d \to \{0,1\}$ such that $\mathrm{dist}(f^*,h) < kd/m \leq \alpha/5$. Let $h^*\colon [n]^d \to \{0,1\}$ be a k-monotone m-block function minimizing $\mathrm{dist}(f^*,h^*)$.

Completeness

Suppose $\mathrm{dist}(f, f^*) \le \varepsilon_1$. Then by the triangle inequality,

$$\begin{aligned}\Pr_{(y,b)\sim D'}[\,h^*(y) \neq b\,] &\le \Pr_{(y,b)\sim D'}[\,h^*(y) \neq f^*(y)\,] + \Pr_{(y,b)\sim D'}[\,f^*(y) \neq f(y)\,] \\ &\quad + \Pr_{(y,b)\sim D'}[\,f(y) \neq b\,] \\ &\le \varepsilon_1 + \frac{2\alpha}{5}.\end{aligned}$$

where to bound the first term $\Pr_{(y,b)\sim D'}[\,h^*(y) \neq f^*(y)\,]$ by $\mathrm{dist}(f^*, h^*) \le \alpha/5$ we used the fact that the marginal distribution of y is uniform when $(y, b) \sim D'$. Thus, the algorithm will find a k-monotone m-block function close to D (without using any queries to f) and accept.

Soundness

Suppose $\mathrm{dist}(f, f^*) \ge \varepsilon_2$. Then by the triangle inequality

$$\begin{aligned}\Pr_{(y,b)\sim D'}[h(y) \neq b] &\ge \Pr_{(y,b)\sim D'}[h(y) \neq f(y)] - \Pr_{(y,b)\sim D'}[f(y) \neq b] \\ &\ge \Pr_{(y,b)\sim D'}[f^*(y) \neq f(y)] - \Pr_{(y,b)\sim D'}[f(y) \neq b] \\ &\ge \varepsilon_2 - \frac{\alpha}{5}\end{aligned}$$

for every k-monotone m-block function h. Since $\varepsilon_2 - 2\alpha/5 \ge \varepsilon_1 + 3\alpha/5$, the algorithm never find a k-monotone m-block function h with low error with respect to D, and the algorithm will reject.

Query complexity

The algorithm only makes queries in constructing D; the number of queries required is $m^d \cdot t = O\Big(\frac{d}{\alpha^2}\left(\frac{5kd}{\alpha} + 1\right)^d \log\frac{kd}{\alpha}\Big)$. ◀

4.2 Tolerant testing *via* agnostic learning

We now present our second algorithm, Algorithm 2, proving the second item of Theorem 8. At its core is the use of an *agnostic learning algorithm* for k-monotone functions, which we first describe.[2]

▶ **Proposition 23.** *There exists an agnostic learning algorithm for k-monotone functions over $[r]^d \to \{0,1\}$ with excess error τ with sample complexity* $\exp(\widetilde{O}(k\sqrt{d}/\tau^2))$.

We will rely on tools from Fourier analysis to prove Proposition 23. For this reason, it will be convenient in this section to view the range as $\{-1, 1\}$ instead of $\{0, 1\}$.

[2] Recall that an *agnostic learner with excess error τ* for some class of functions $\mathcal{C}$ is an algorithm that, given an unknown distribution D, an unknown arbitrary function f, and access to random labelled samples $\langle x, f(x)\rangle$ where $x \sim D$, satisfies the following. It outputs a hypothesis function $\hat{h}$ such that $\Pr_{x\sim D}\left[\,f(x) \neq \hat{h}(x)\,\right] \le \mathrm{OPT}_D + \tau$ with probability at least $2/3$, where $\mathrm{OPT}_D = \min_{h\in\mathcal{C}} \Pr_{x\sim D}[\,f(x) \neq h(x)\,]$ (i.e., it performs "almost as well as the best function in $\mathcal{C}$").

Algorithm 2 Multiplicative approximation with $\exp(\widetilde{O}(k\sqrt{d}/((\varepsilon_2-3\varepsilon_1)^2))$ queries.

Require: Query access to $f\colon [n]^d \to \{0,1\}$, $\varepsilon_2 > 3\varepsilon_1 \geq 0$, a positive integer k

1: $\alpha \leftarrow (\varepsilon_2 - 3\varepsilon_1)$, $m \leftarrow \lceil 6kd/\varepsilon \rceil$, $t \leftarrow \lceil 3d(k+1)/\varepsilon \ln m + \ln 100 \rceil$
2: Define D to be the distribution over $[m]^d \times \{0,1\}$ such that $D(x,b) = \Pr_{y\in\mathcal{B}^{-1}(x)}[\, f(y) = b\,]$.
3: ▷ $\mathcal{A}_D(\tau, f)$ denotes the output of an agnostic learner of k-monotone functions with respect to D, with excess error τ and probability of failure $1/10$
4: $h\colon [m]^d \to \mathbb{R} \leftarrow \mathcal{A}_D(\alpha/12, f)$.
5: Estimate $\Pr_{(x,b)\sim D}[\, h(x) \neq b\,]$ to within $\pm\alpha/7$ with probability of failure $1/10$, using $O(1/\alpha^2)$ queries.
6: **if** the estimate is more than $\varepsilon_1 + \frac{5\alpha}{12}$ **then return** REJECT
7: **end if**
8: **if** $\mathrm{dist}(h,\ell) = \Pr_{x\in[m]^d}[\, h(x) \neq \ell(x)\,] \leq 2\varepsilon_1 + \frac{5\alpha}{12}$ for some k-monotone m-block function ℓ **then return** ACCEPT
9: **else return** REJECT
10: **end if**

▶ **Definition 24.** For a Boolean function $f\colon [r]^d \to \{-1,1\}$, we define

$$\mathbf{Inf}_i[f] = 2\Pr\Big[\, f(x) \neq f(x^{(i)})\,\Big]$$

where $x = (x_1, x_2, \ldots, x_d)$ is a uniformly random string over $[r]^d$, and

$$x^{(i)} = (x_1, x_2, \ldots, x_{i-1}, x_i', x_{i+1}, \ldots, x_d)$$

for x' drawn independently and uniformly from $[r]$. We also define $\mathbf{Inf}[f] = \sum_{i=1}^d \mathbf{Inf}_i[f]$.

We first generalize the following result, due to Blais *et al.*, for more general domains:

▶ **Proposition 25** ([9]). *Let $f\colon \{0,1\}^d \to \{-1,1\}$ be a k-monotone function. Then $\mathbf{Inf}[f] \leq k\sqrt{d}$.*

▶ **Lemma 26** (Generalization). *Let $f\colon [r]^d \to \{-1,1\}$ be a k-monotone function. Then $\mathbf{Inf}[f] \leq k\sqrt{d}$.*

Proof. For any two strings $y^0, y^1 \in [r]^d$, let $f_{y^0,y^1}\colon \{0,1\}^d \to \{-1,1\}$ be the function obtained by setting $f_{y^0,y^1}(x) = f(y^x)$, where $y^x \in [r]^d$ is defined as

$$y_i^x = \begin{cases} \min\{y_i^0, y_i^1\} & \text{if } x_i = 0 \\ \max\{y_i^0, y_i^1\} & \text{if } x_i = 1 \end{cases}$$

Since f was a k-monotone function, so is f_{y^0,y^1}. Thus $\mathbf{Inf}[f_{y^0,y^1}] \leq k\sqrt{d}$ for every choice of y^0 and y^1. It is not hard to see that for any fixed $i \in [d]$ the following two processes yield the same distribution over $[r]^d \times [r]^d$:

- Draw $z \in [r]^d$, $z_i' \in [r]$ independently and uniformly at random, set

$$z' \stackrel{\text{def}}{=} (z_1, \ldots, z_{i-1}, z_i', z_{i+1}, \ldots, z_d),$$

  and output (z, z');
- Draw $y^0, y^1 \in [r]^d, x \in \{0,1\}^d$ independently and uniformly at random, and output $(y^x, y^{x^{(i)}})$.

This implies that

$$\begin{aligned}
\mathbf{Inf}[f] &= \sum_{i=1}^{d} \mathbf{Inf}_i[f] = \sum_{i=1}^{d} 2 \Pr_{z\in[r]^d}[f(z) \neq f(z^{(i)})] \\
&= \sum_{i=1}^{d} 2\mathbb{E}_{y^0,y^1\in[r]^d}\left[\Pr_{x\in\{0,1\}^d}\left[f(y^x) \neq f(y^{x^{(i)}})\right]\right] \\
&= \mathbb{E}_{y^0,y^1\in[r]^d}\left[\sum_{i=1}^{d} 2 \Pr_{x\in\{0,1\}^d}\left[f(y^x) \neq f(y^{x^{(i)}})\right]\right] \\
&= \mathbb{E}_{y^0,y^1\in[r]^d}\left[\sum_{i=1}^{d} 2 \Pr_{x\in\{0,1\}^d}\left[f_{y_0,y_1}(x) \neq f_{y_0,y_1}(x^{(i)})\right]\right] \\
&= \mathbb{E}_{y^0,y^1}[\mathbf{Inf}[f_{y^0,y^1}]] \leq \mathbb{E}_{y^0,y^1}[k\sqrt{d}] = k\sqrt{d}.
\end{aligned}$$

◀

For two functions $f, g\colon [r]^d \to \mathbb{R}$, we define the inner product $\langle f, g\rangle = \mathbb{E}_x[f(x)g(x)]$, where the expectation is taken with respect to the uniform distribution. It is known that for functions $f\colon [r]^d \to \mathbb{R}$, there is a "Fourier basis" of orthonormal functions f. To construct such a basis, we can take any orthonormal basis $\{\phi_0 \equiv 1, \phi_1, \ldots, \phi_{|r|-1}\}$ for functions $f\colon [r] \to \mathbb{R}$. Given such a basis, a Fourier basis is the collection of functions ϕ_α, where $\alpha \in [r]^d$, and $\phi_\alpha(x) = \prod_{i=1}^{d} \phi_{\alpha_i}(x_i)$. Then every $f\colon [r]^d \to \mathbb{R}$ has a unique representation $f = \sum_{\alpha\in[r]^d} \widehat{f}(\alpha)\phi_\alpha$, where $\widehat{f}(\alpha) = \langle f, \phi_\alpha\rangle \in \mathbb{R}$.

Many Fourier formulæ hold in arbitrary Fourier bases, an important example being Parseval's Identity: $\sum_{\alpha\in[r]^d} \widehat{f}(\alpha)^2 = 1$. We will use the following property:

▶ **Lemma 27** ([39, Proposition 8.23]). *For $\alpha \in [r]^d$, let $|\alpha|$ denote the number of nonzero coordinates in α. Then we have*

$$\mathbf{Inf}[f] = \sum_{\alpha\in[r]^d} |\alpha|\, \hat{f}(\alpha)^2.$$

▶ **Lemma 28.** *If $\mathbf{Inf}[f] \leq k$, then $\sum_{\alpha:|\alpha|>k/\varepsilon} \widehat{f}(\alpha)^2 \leq \varepsilon$.*

Proof. If not, then $\mathbf{Inf}[f] = \sum_\alpha |\alpha|\, \widehat{f}(\alpha)^2 \geq \sum_{\alpha:|\alpha|>k/\varepsilon} |\alpha|\, \widehat{f}(\alpha)^2 \geq \frac{k}{\varepsilon} \sum_{\alpha:|\alpha|>k/\varepsilon} \widehat{f}(\alpha)^2 > \frac{k}{\varepsilon} \cdot \varepsilon = k$, a contradiction. ◀

▶ **Lemma 29.** *Let p be the function $\sum_{\alpha:|\alpha|\leq t} \widehat{f}(\alpha)\phi_\alpha$. Then*

- **(i)** $\|p - f\|_2^2 = \mathbb{E}_{x\in[r]^d}[(p(x) - f(x))^2] = \sum_{\alpha:|\alpha|>t} \widehat{f}(\alpha)^2$;
- **(ii)** *p is expressible as a linear combination of real-valued functions over $[r]^d$, each of which only depends on at most t coordinates;*
- **(iii)** *p is expressible as a degree-t polynomial over the rd indicator functions $\mathbb{1}_{\{x_i=j\}}$ for $1 \leq i \leq d$ and $j \in [r]$.*

▶ **Theorem 30** ([30, Theorem 5]). *Let $\mathcal{C}$ be a class of Boolean functions over $\mathcal{X}$ and $\mathcal{S}$ a collection of real-valued functions over $\mathcal{X}$ such that for every $f\colon \mathcal{X} \to \{-1, 1\}$ in $\mathcal{C}$, there exists a function $p\colon \mathcal{X} \to \mathbb{R}$ such that p is expressible as a linear combination of functions from $\mathcal{S}$ and $\|p - f\|_2^2 \leq \tau^2$. Then there is an agnostic learning algorithm for $\mathcal{C}$ achieving excess error τ which has sample complexity* $\mathrm{poly}(|\mathcal{S}|, 1/\tau)$.

Importantly, this algorithm is still successful with inconsistent labelled samples (examples), as long as they come from a distribution on $\mathcal{X} \times \{-1, 1\}$, where the marginal distribution on $\mathcal{X}$ is uniform.

Now we put all the pieces together. To agnostically learn a k-monotone function, we simply perform the agnostic learning algorithm of [30] on the distribution D over $[m]^d \times \{-1, 1\}$ defined by

$$D(x, b) = \Pr_{y \in \mathcal{B}^{-1}(x)} [\, f(y) = b \,] .$$

To generate a sample (x, b) from D, we draw a uniformly random string in $x \in [m]^d$, and b is the result of a query for the value of $f(y)$ for a uniformly random $y \in \mathcal{B}^{-1}(x)$. From Theorem 29, we can take $\mathcal{S}$ to be the set of $(k\sqrt{d}/\tau^2)$-way products of rd indicator functions. It follows that $|\mathcal{S}| = \binom{rd}{k\sqrt{d}/\tau^2} = \exp(\widetilde{O}(k\sqrt{d}/\tau^2))$.

▶ **Proposition 31.** *Algorithm 2 accepts all functions ε_1-close to k-monotone functions, and rejects all functions ε_2-far from k-monotone, when $\varepsilon_2 > 3\varepsilon_1$ (with probability at least $2/3$). Its query complexity is $\exp(\widetilde{O}(k\sqrt{d}/(\varepsilon_2 - 3\varepsilon_1)^2))$.*

Proof. By a union bound, we have that with probability at least $8/10$ both Step 5 and Step 4 succeed. We hereafter condition on this.

Completeness

Suppose f is ε_1-close to k-monotone. Theorem 21 and the triangle inequality imply that there is a k-monotone m-block function g^* such that $\mathrm{dist}(f, g^*) \leq \varepsilon_1 + \alpha/6$. The agnostic learning algorithm thus returns a hypothesis h such that $\mathrm{dist}(f, h) \leq \varepsilon_1 + \alpha/4$. The algorithm estimates this closeness to within $\alpha/7$, so the estimate obtained in Step 5 is at most $\varepsilon_1 + \varepsilon/4 + \varepsilon/7 < \varepsilon_1 + 5\alpha/12$ and the algorithm does not reject in this step. By the triangle inequality, h is $(2\varepsilon_1 + 5\alpha/12)$-close to k-monotone, and the algorithm will accept. There is no estimation error here, since no queries to f are required.

Soundness

Now suppose f is ε_2-far from k-monotone, where $\varepsilon_2 = 3\varepsilon_1 + \alpha$ for some $\alpha > 0$. Suppose the algorithm does not reject when estimating $\mathrm{dist}(f, h)$, where h is the hypothesis returned by the agnostic learning algorithm. Then $\mathrm{dist}(f, h) \leq \varepsilon_1 + 5\alpha/12 + \alpha/7 < \varepsilon_1 + 7\alpha/12$. By the triangle inequality, if t is a k-monotone function, $\mathrm{dist}(h, t) \geq \mathrm{dist}(f, t) - \mathrm{dist}(f, h) > \varepsilon_2 - (\varepsilon_1 + 7\alpha/12) = 2\varepsilon_1 + 5\alpha/12$. The algorithm will thus reject in the final step.

Query complexity

The query complexity of the algorithm is dominated by the query complexity of the agnostic learning algorithm, which is $\exp(\widetilde{O}(k\sqrt{d}/\alpha^2)) = \exp(\tilde{O}\Big(k\sqrt{d}/(\varepsilon_2 - 3\varepsilon_1)^2\Big))$. ◀

Acknowledgments. We would like to thank Eric Blais for helpful remarks on an earlier version of this paper, and an anonymous reviewer for very detailed and insightful comments.

References

1 Nir Ailon and Bernard Chazelle. Information theory in property testing and monotonicity testing in higher dimension. *Inf. Comput.*, 204(11):1704–1717, 2006.

2 Kazuyuki Amano and Akira Maruoka. A superpolynomial lower bound for a circuit computing the Clique function with at most $(1/6) \log \log n$ negation gates. *SIAM Journal on Computing*, 35(1):201–216, 2005.

3 Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1987.

4 Maria-Florina Balcan, Eric Blais, Avrim Blum, and Liu Yang. Active property testing. In *FOCS*, pages 21–30. IEEE Computer Society, 2012.

5 Tŭgkan Batu, Ronitt Rubinfeld, and Patrick White. Fast approximate PCPs for multidimensional bin-packing problems. *Inf. Comput.*, 196(1):42–56, 2005.

6 Aleksandrs Belovs and Eric Blais. A polynomial lower bound for testing monotonicity. In *STOC*, pages 1021–1032. ACM, 2016.

7 Piotr Berman, Sofya Raskhodnikova, and Grigory Yaroslavtsev. L_p-testing. In *STOC*, pages 164–173. ACM, 2014.

8 Eric Blais, Joshua Brody, and Kevin Matulef. Property testing lower bounds via communication complexity. *Computational Complexity*, 21(2):311–358, 2012.

9 Eric Blais, Clément L. Canonne, Igor Carboni Oliveira, Rocco A. Servedio, and Li-Yang Tan. Learning circuits with few negations. In *APPROX-RANDOM*, volume 40 of *LIPIcs*, pages 512–527. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

10 Jop Briët, Sourav Chakraborty, David García-Soriano, and Arie Matsliah. Monotonicity testing and shortest-path routing on the cube. *Combinatorica*, 32(1):35–53, 2012.

11 Nader H. Bshouty and Christino Tamon. On the Fourier spectrum of monotone functions. *J. ACM*, 43(4):747–770, 1996.

12 Clément L. Canonne, Elena Grigorescu, Siyao Guo, Akash Kumar, and Karl Wimmer. Testing k-monotonicity. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:136, 2016.

13 Clément L. Canonne and Ronitt Rubinfeld. Testing probability distributions underlying aggregated data. In *ICALP (1)*, volume 8572 of *Lecture Notes in Computer Science*, pages 283–295. Springer, 2014.

14 Deeparnab Chakrabarty and C. Seshadhri. An $o(n)$ monotonicity tester for boolean functions over the hypercube. In *STOC*, pages 411–418. ACM, 2013. Journal version as [17].

15 Deeparnab Chakrabarty and C. Seshadhri. Optimal bounds for monotonicity and Lipschitz testing over hypercubes and hypergrids. In *STOC*, pages 419–428, 2013.

16 Deeparnab Chakrabarty and C. Seshadhri. An optimal lower bound for monotonicity testing over hypergrids. *Theory of Computing*, 10:453–464, 2014.

17 Deeparnab Chakrabarty and C. Seshadhri. An $o(n)$ Monotonicity Tester for Boolean Functions over the Hypercube. *SIAM J. Comput.*, 45(2):461–472, 2016.

18 Xi Chen, Anindya De, Rocco A. Servedio, and Li-Yang Tan. Boolean function monotonicity testing requires (almost) $n^{1/2}$ non-adaptive queries. In *STOC*, pages 519–528. ACM, 2015.

19 Xi Chen, Rocco A. Servedio, and Li-Yang Tan. New algorithms and lower bounds for monotonicity testing. In *FOCS*, pages 286–295. IEEE Computer Society, 2014.

20 Yevgeniy Dodis, Oded Goldreich, Eric Lehman, Sofya Raskhodnikova, Dana Ron, and Alex Samorodnitsky. Improved testing algorithms for monotonicity. In *RANDOM-APPROX*, volume 1671 of *Lecture Notes in Computer Science*, pages 97–108. Springer, 1999.

21 Funda Ergün, Sampath Kannan, Ravi Kumar, Ronitt Rubinfeld, and Mahesh Viswanathan. Spot-checkers. *J. Comput. Syst. Sci.*, 60(3):717–751, 2000.

22 Shahar Fattal and Dana Ron. Approximating the distance to monotonicity in high dimensions. *ACM Trans. Algorithms*, 6(3), 2010.

23 Eldar Fischer. On the strength of comparisons in property testing. *Inf. Comput.*, 189(1):107–116, 2004.

24 Eldar Fischer, Eric Lehman, Ilan Newman, Sofya Raskhodnikova, Ronitt Rubinfeld, and Alex Samorodnitsky. Monotonicity testing over general poset domains. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 474–483, 2002.

25 Oded Goldreich, Shafi Goldwasser, Eric Lehman, Dana Ron, and Alex Samorodnitsky. Testing monotonicity. *Combinatorica*, 20(3):301–337, 2000.

26 Siyao Guo and Ilan Komargodski. Negation-limited formulas. In *APPROX-RANDOM*, volume 40 of *LIPIcs*, pages 850–866. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

27 Siyao Guo, Tal Malkin, Igor Carboni Oliveira, and Alon Rosen. The power of negations in cryptography. In *TCC (1)*, volume 9014 of *Lecture Notes in Computer Science*, pages 36–65. Springer, 2015.

28 Shirley Halevy and Eyal Kushilevitz. Testing monotonicity over graph products. *Random Struct. Algorithms*, 33(1):44–67, 2008.

29 Stasys Jukna. *Boolean Function Complexity*. Springer, 2012.

30 Adam Tauman Kalai, Adam R. Klivans, Yishay Mansour, and Rocco A. Servedio. Agnostically learning halfspaces. *SIAM J. Comput.*, 37(6):1777–1805, 2008.

31 Michael J. Kearns and Dana Ron. Testing problems with sublearning sample complexity. *J. Comput. Syst. Sci.*, 61(3):428–456, 2000.

32 Michael J. Kearns and Leslie G. Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. *J. ACM*, 41(1):67–95, 1994.

33 Subhash Khot, Dor Minzer, and Muli Safra. On monotonicity testing and Boolean isoperimetric type theorems. In *FOCS*, pages 52–58. IEEE Computer Society, 2015.

34 Pravesh Kothari, Amir Nayyeri, Ryan O'Donnell, and Chenggang Wu. Testing surface area. In *SODA*, pages 1204–1214. SIAM, 2014.

35 Chengyu Lin and Shengyu Zhang. Sensitivity conjecture and log-rank conjecture for functions with small alternating numbers. *CoRR*, abs/1602.06627, 2016.

36 A. A. Markov. On the inversion complexity of systems of functions. *Doklady Akademii Nauk SSSR*, 116:917–919, 1957. English translation in [37].

37 A. A. Markov. On the inversion complexity of a system of functions. *Journal of the ACM*, 5(4):331–334, October 1958.

38 Joe Neeman. Testing surface area with arbitrary accuracy. In *STOC*, pages 393–397. ACM, 2014.

39 Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.

40 Ryan O'Donnell and Rocco A. Servedio. Learning monotone decision trees in polynomial time. *SIAM J. Comput.*, 37(3):827–844, 2007.

41 Ryan O'Donnell and Karl Wimmer. KKL, Kruskal–Katona, and monotone nets. In *FOCS*, pages 725–734. IEEE Computer Society, 2009.

42 Michal Parnas, Dana Ron, and Ronitt Rubinfeld. Tolerant property testing and distance approximation. *Journal of Computer and System Sciences*, 72(6):1012–1042, 2006.

43 Ran Raz and Avi Wigderson. Monotone circuits for matching require linear depth. *J. ACM*, 39(3):736–744, 1992.

44 Alexander A Razborov. Lower bounds on the monotone complexity of some Boolean functions. *Doklady Akademii Nauk SSSR*, 281(4):798–801, 1985.

45 Benjamin Rossman. Correlation bounds against monotone NC^1. In *Conference on Computational Complexity* (CCC), 2015.

46 Rocco A. Servedio. On learning monotone DNF under product distributions. *Inf. Comput.*, 193(1):57–74, 2004.

47 List of open problems in sublinear algorithms: Problem 70, 2016. Originally posed in [7]. URL: `http://sublinear.info/70`.

48 Leslie G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984.