Algorithmica (2011) 61:580–605
DOI 10.1007/s00453-010-9426-9

Efficiently Testing Sparse $GF(2)$ Polynomials

Ilias Diakonikolas · Homin K. Lee · Kevin Matulef · Rocco A. Servedio · Andrew Wan

Received: 20 November 2009 / Accepted: 22 June 2010 / Published online: 13 July 2010

Abstract We give the first algorithm that is both query-efficient and time-efficient for testing whether an unknown function $f : \{0,1\}^n \to \{-1,1\}$ is an s-sparse $GF(2)$ polynomial versus ϵ-far from every such polynomial. Our algorithm makes $\mathrm{poly}(s, 1/\epsilon)$ black-box queries to f and runs in time $n \cdot \mathrm{poly}(s, 1/\epsilon)$. The only previous algorithm for this testing problem (Diakonikolas et al. in Proceedings of the 48th Annual Symposium on Foundations of Computer Science, FOCS, pp. 549–558, 2007) used $\mathrm{poly}(s, 1/\epsilon)$ queries, but had running time exponential in s and superpolynomial in $1/\epsilon$.

Research of I. Diakonikolas was supported by NSF grants CCF-0728736, CCF-0525260, and by an Alexander S. Onassis Foundation Fellowship.
H.K. Lee, R.A. Servedio and A. Wan supported in part by NSF grants CCF-0347282, CCF-0523664 and CNS-0716245, and by DARPA award HR0011-08-1-0069.
K. Matulef supported in part by the National Natural Science Foundation of China Grant 60553001, the National Basic Research Program of China Grant 2007CB807900,2007CB807901, and the US National Science Foundation grants 0514771, 0732334, 0728645.

I. Diakonikolas · R.A. Servedio · A. Wan
Columbia University, New York, NY, USA

I. Diakonikolas
e-mail: ilias@cs.columbia.edu

R.A. Servedio
e-mail: rocco@cs.columbia.edu

A. Wan
e-mail: atw12@cs.columbia.edu

H.K. Lee (✉)
University of Texas, Austin, TX, USA
e-mail: homin@cs.utexas.edu

K. Matulef
Tsinghua University, Beijing, China
e-mail: matulef@itcs.tsinghua.edu.cn

Our approach significantly extends the "testing by implicit learning" methodology of Diakonikolas et al. (Proceedings of the 48th Annual Symposium on Foundations of Computer Science, FOCS, pp. 549–558, 2007). The learning component of that earlier work was a brute-force exhaustive search over a concept class to find a hypothesis consistent with a sample of random examples. In this work, the learning component is a sophisticated exact learning algorithm for sparse $GF(2)$ polynomials due to Schapire and Sellie (J. Comput. Syst. Sci. 52(2):201–213, 1996). A crucial element of this work, which enables us to simulate the membership queries required by Schapire and Sellie (J. Comput. Syst. Sci. 52(2):201–213, 1996), is an analysis establishing new properties of how sparse $GF(2)$ polynomials simplify under certain restrictions of "low-influence" sets of variables.

Keywords Property testing · $GF(2)$ polynomials · Sparse polynomials · Randomized algorithms

1 Introduction

Background and Motivation Given black-box access to an unknown function $f : \{0,1\}^n \to \{-1,1\}$, a natural question to ask is whether the function has a particular form. Is it representable by a small decision tree, or small circuit, or sparse polynomial? In the field of computational learning theory, the standard approach to this problem is to assume that f belongs to a specific class $\mathcal{C}$ of functions of interest, and the goal is to identify or approximate f. In contrast, in property testing nothing is assumed about the unknown function f, and the goal of the testing algorithm is to output "yes" with high probability if $f \in \mathcal{C}$ and "no" with high probability if f is ϵ-far from every $g \in \mathcal{C}$. (Here the distance between two functions f, g is measured with respect to the uniform distribution on $\{0,1\}^n$, so f and g are ϵ-far if they disagree on more than an ϵ fraction of all inputs.) The complexity of a testing algorithm is measured both in terms of the number of black-box queries it makes to f (*query complexity*) as well as the time it takes to process the results of those queries (*time complexity*).

There are many connections between learning theory and testing, and a growing body of work relating the two fields (see [24] and references therein). Testing algorithms have been given for a range of different function classes such as linear functions over $GF(2)$ (i.e. parities) [4]; degree-d $GF(2)$ polynomials [1]; Boolean literals, conjunctions, and s-term monotone DNF formulas [22]; k-juntas (i.e. functions which depend on at most k variables) [12]; halfspaces [20]; and more (see surveys of [10, 23, 26]).

Recently, Diakonikolas et al. [8] gave a general technique, called "testing by implicit learning," which they used to test a variety of different function classes that were not previously known to be testable. Intuitively, these classes correspond to functions with "concise representations," such as s-term DNFs, size-s Boolean formulas, size-s Boolean circuits, and s-sparse polynomials over constant-size finite fields. For each of these classes, the testing algorithm of [8] makes only $\mathrm{poly}(s, 1/\epsilon)$ queries (independent of n).

The main drawback of the [8] testing algorithm is its time complexity. For each of the classes mentioned above, the algorithm's running time is $2^{\omega(s)}$ as a function of s and $\omega(\text{poly}(1/\epsilon))$ as a function of ϵ.[1] Thus, a natural question asked by [8] is whether any of these classes can be tested with both time complexity and query complexity $\text{poly}(s, 1/\epsilon)$.

Our Result: Efficiently Testing Sparse GF(2) Polynomials In this paper we focus on the class of *s-sparse polynomials over GF(2)*. Polynomials over $GF(2)$ (equivalently, parities of ANDs of input variables) are a simple and well-studied representation for Boolean functions. It is well known that every Boolean function has a unique representation as a multilinear polynomial over $GF(2)$, so the sparsity (number of monomials) of this polynomial is a very natural measure of the complexity of f. Sparse $GF(2)$ polynomials have been studied by many authors from a range of different perspectives such as learning [3, 6, 7, 11, 27], approximation and interpolation [14, 16, 25], the complexity of (approximate) counting [9, 17, 18], and property testing [8].

The main result of this paper is a testing algorithm for s-sparse $GF(2)$ polynomials that is both time-efficient and query-efficient:

Theorem 1 *There is a* $\text{poly}(s, 1/\epsilon)$*-query algorithm with the following performance guarantee*: *given parameters* s, ϵ *and black-box access to any* $f : \{0,1\}^n \to \{-1,1\}$, *it runs in time* $\text{poly}(s, 1/\epsilon)$ *and tests whether* f *is an* s*-sparse* $GF(2)$ *polynomial versus* ϵ*-far from every* s*-sparse polynomial*.

This answers the question of [8] by exhibiting an interesting and natural class of functions with "concise representations" that can be tested efficiently, both in terms of query complexity and running time.

We obtain our main result by extending the "testing by implicit learning" approach of [8]. In that work the "implicit learning" step used a naive brute-force search for a consistent hypothesis, while in this paper we employ a sophisticated proper learning algorithm due to Schapire and Sellie [27]. However, it is much more difficult to "implicitly" run the [27] algorithm than the brute-force search of [8]. One of the main technical contributions of this paper is a new structural theorem about how s-sparse $GF(2)$ polynomials are affected by certain carefully chosen restrictions; this is an essential ingredient that enables us to use the [27] algorithm. We elaborate on this below.

Techniques We begin with a brief review of the main ideas of [8]. The approach of [8] builds on the observation of Goldreich et al. [13] that any *proper* learning algorithm for a function class $\mathcal{C}$ can be used as a testing algorithm for $\mathcal{C}$. (Recall that a proper learning algorithm for $\mathcal{C}$ is one which outputs a hypothesis h that itself belongs to $\mathcal{C}$.) The idea behind this observation is that if the function f being tested belongs

[1] We note that the algorithm also has a linear running time dependence on n, the number of input variables; this is in some sense inevitable since the algorithm must set n bit values just to pose a black-box query to f. Our algorithm has running time linear in n for the same reason. For the rest of the paper we discuss the running time only as a function of s and ϵ.

to $\mathcal{C}$ then a proper learning algorithm will succeed in constructing a hypothesis that is close to f, while if f is ϵ-far from every $g \in \mathcal{C}$ then any hypothesis $h \in \mathcal{C}$ that the learning algorithm outputs must necessarily be far from f. Thus any class $\mathcal{C}$ can be tested to accuracy ϵ using essentially the same number of queries that are required to properly learn the class to accuracy $\Theta(\epsilon)$.

The basic approach of [13] did not yield query-efficient testing algorithms (with query complexity independent of n) since virtually every interesting class of functions over $\{0,1\}^n$ requires $\Omega(\log n)$ examples for proper learning. However, [8] showed that for many classes of functions defined by a size parameter s, it is possible to "implicitly" run a (very naive) proper learning algorithm over a number of variables that is independent of n, and thus obtain an overall query complexity independent of n. More precisely, they first observed that for many classes $\mathcal{C}$ every $f \in \mathcal{C}$ is "very close" to a function $f' \in \mathcal{C}$ for which the number r of relevant variables is polynomial in s and independent of n; roughly speaking, the relevant variables for f' are the variables that have high influence in f. (For example, if f is an s-sparse $GF(2)$ polynomial, an easy argument shows that there is a function f'—obtained by discarding from f all monomials of degree more than $\log(s/\tau)$—that is τ-close to f and depends on at most $r = s\log(s/\tau)$ variables.) They then showed how, using ideas of Fischer et al. [12] for testing juntas, it is possible to construct a sample of uniform random examples over $\{0,1\}^r$ which with high probability are all labeled according to f'. At this point, the proper learning algorithm employed by [8] was a naive brute-force search. The algorithm tried all possible functions in $\mathcal{C}$ over r (as opposed to n) variables, to see if any were consistent with the labeled sample. Diakonikolas et al. [8] thus obtained a testing algorithm with overall query complexity $\mathrm{poly}(s/\epsilon)$ but whose running time was dominated by the brute-force search. For the class of s-sparse $GF(2)$ polynomials, their algorithm used $\tilde{O}(s^4/\epsilon^2)$ queries but had running time at least $2^{\omega(s)} \cdot (1/\epsilon)^{\log\log(1/\epsilon)}$ (for the required value of τ, which is $\mathrm{poly}(\epsilon/s)$, there are at least this many s-sparse $GF(2)$ polynomials over $r = s\log(s/\tau)$ variables).

Current Approach The high-level idea of the current work is to employ a much more sophisticated—and efficient—proper learning algorithm than brute-force search. In particular we would like to use a proper learning algorithm which, when applied to learn a function over only r variables, runs in time polynomial in r and in the size parameter s. For the class of s-sparse $GF(2)$ polynomials, precisely such an algorithm was given by Schapire and Sellie [27]. Their algorithm, which we describe in Sect. 4, is computationally efficient and generates a hypothesis h which is an s-sparse $GF(2)$ polynomial. But this power comes at a price: the algorithm requires access to a *membership query* oracle, i.e. a black-box oracle for the function being learned. Thus, in order to run the Schapire/Sellie algorithm in the "testing by implicit learning" framework, it is necessary to simulate membership queries to an approximating function $f' \in \mathcal{C}$ which is close to f but depends on only r variables. This is significantly more challenging than generating uniform random examples labeled according to f', which is all that is required in the original [8] approach.

To see why membership queries to f' are more difficult to simulate than uniform random examples, recall that f and the f' described above (obtained from f by dis-

carding high-degree monomials) are τ-close. Intuitively this is extremely close, disagreeing only on a $1/m$ fraction of inputs for an m that is much larger than the number of random examples required for learning f' via brute-force search (this number is "small"—independent of n—because f' depends on only r variables). Thus in the [8] approach it suffices to use f, the function to which we actually have black-box access, rather than f' to label the random examples used for learning f'; since f and f' are so close, and the examples are uniformly random, with high probability all the labels will also be correct for f'. However, in the membership query scenario of the current paper, things are no longer that simple. For any given f' which is close to f, one can no longer assume that the learning algorithm's queries to f' are uniformly distributed and hence unlikely to hit the error region—indeed, it is possible that the learning algorithm's membership queries to f' are clustered on the few inputs where f and f' disagree.

In order to successfully simulate membership queries, we must somehow consistently answer queries according to a particular f', even though we only have oracle access to f. Moreover this must be done implicitly in a query-efficient way, since explicitly identifying even a single variable relevant to f' requires at least $\Omega(\log n)$ queries. This is the main technical challenge in the paper.

We meet this challenge by showing that for any s-sparse polynomial f, an approximating f' can be obtained as a restriction of f by setting certain carefully chosen subsets of variables to zero. Roughly speaking, this restriction is obtained by randomly partitioning all of the input variables into r subsets and zeroing out all subsets whose variables have small "collective influence" (more precisely, small variation in the sense of [12]).[2] Our main technical theorem (Theorem 12, given in Sect. 5) shows that this f' is indeed close to f and has at most one of its relevant variables in each of the surviving subsets. We moreover show that these relevant variables for f' all have high influence in f.[3] This property is important in enabling our simulation of membership queries. In addition to the crucial role that Theorem 12 plays in the completeness proof for our test, we feel that the new insights the theorem gives into how sparse polynomials "simplify" under (appropriately defined) random restrictions may be of independent interest.

Organization In Sect. 3, we present our testing algorithm, *Test-Sparse-Poly*, along with a high-level description and sketch of correctness. In Sect. 4 we describe in detail the "learning component" of the algorithm. In Sect. 5 we prove Theorem 12, which provides intuition behind the algorithm and serves as the main technical tool in the completeness proof. The completeness and soundness proofs are given in Sects. 6 and 7, respectively. We make some concluding remarks in Sect. 8.

[2] We observe that it is important that the restriction sets these variables to zero rather than to a random assignment; intuitively this is because setting a variable to zero "kills" all monomials that contain the variable, whereas setting it to 1 does not.

[3] The converse is not true; examples can be given which show that not every variable that has "high influence" (in the required sense) in f will in general become a relevant variable for f'.

2 Preliminaries

GF(2) *Polynomials* A *GF*(2) polynomial is a parity of monotone conjunctions (monomials). It is *s-sparse* if it contains at most s monomials (including the constant-1 monomial if it is present). The *length* of a monomial is the number of distinct variables that occur in it; over *GF*(2), this is simply its degree.

Notation For $i \in \mathbb{N}^*$, denote $[i] \stackrel{\text{def}}{=} \{1, 2, \ldots, i\}$. It will be convenient to view the output range of a Boolean function f as $\{-1, 1\}$ rather than $\{0, 1\}$, i.e. $f : \{0,1\}^n \to \{-1, 1\}$. We view the hypercube as a measure space endowed with the uniform product probability measure. For $I \subseteq [n]$ we denote by $\{0,1\}^I$ the set of all partial assignments to the coordinates in I. For $w \in \{0,1\}^{[n]\setminus I}$ and $z \in \{0,1\}^I$, we write $w \sqcup z$ to denote the assignment in $\{0,1\}^n$ whose ith coordinate is w_i if $i \in [n] \setminus I$ and is z_i if $i \in I$. Whenever an element z in $\{0,1\}^I$ is chosen randomly (we denote $z \in_{\mathrm{R}} \{0,1\}^I$), it is chosen with respect to the uniform measure on $\{0,1\}^I$. We use $\mathbb{E}$ and $\mathbb{V}$ to denote the standard notions of expectation and variance of a random variable.

Influence, Variation and the Independence Test Recall the classical notion of *influence* [15]: The *influence* of the ith coordinate on $f : \{0,1\}^n \to \{-1,1\}$ is $\mathrm{Inf}_i(f) \stackrel{\text{def}}{=} \Pr_{x \in_{\mathrm{R}} \{0,1\}^n}[f(x) \neq f(x^{\oplus i})]$, where $x^{\oplus i}$ denotes x with the ith bit flipped.

The influence of a single coordinate can be generalized to a set of multiple coordinates by using the *variation*:

Definition 2 (Variation, [12]) Let $f : \{0,1\}^n \to \{-1,1\}$, and let $I \subseteq [n]$. We define the *variation* of f on I as $\mathrm{Vr}_f(I) \stackrel{\text{def}}{=} \mathbb{E}_{w \in_{\mathrm{R}} \{0,1\}^{[n]\setminus I}}[\mathbb{V}_{z \in_{\mathrm{R}} \{0,1\}^I}[f(w \sqcup z)]]$.

When $I = \{i\}$ we will sometimes write $\mathrm{Vr}_f(i)$ instead of $\mathrm{Vr}_f(\{i\})$. It is easy to check that $\mathrm{Vr}_f(i) = \mathrm{Inf}_i(f)$, so variation is indeed a generalization of influence. Intuitively, the variation is a measure of the ability of a set of variables to sway a function's output. The following two simple properties of the variation will be useful for the analysis of our testing algorithm:

Lemma 3 (Monotonicity and sub-additivity, [12]) *Let* $f : \{0,1\}^n \to \{-1,1\}$ *and* $A, B \subseteq [n]$. *Then* $\mathrm{Vr}_f(A) \leq \mathrm{Vr}_f(A \cup B) \leq \mathrm{Vr}_f(A) + \mathrm{Vr}_f(B)$.

Lemma 4 (Probability of detection, [12]) *Let* $f : \{0,1\}^n \to \{-1,1\}$ *and* $I \subseteq [n]$. *If* $w \in_{\mathrm{R}} \{0,1\}^{[n]\setminus I}$ *and* $z_1, z_2 \in_{\mathrm{R}} \{0,1\}^I$ *are chosen independently, then* $\Pr[f(w \sqcup z_1) \neq f(w \sqcup z_2)] = \frac{1}{2}\mathrm{Vr}_f(I)$.

We now recall the *independence test* from [12], a simple two query test used to determine whether a function f is independent of a given set $I \subseteq [n]$ of coordinates.

Independence Test Given inputs $f : \{0,1\}^n \to \{-1,1\}$ and $I \subseteq [n]$, the independence test chooses $w \in_{\mathrm{R}} \{0,1\}^{[n]\setminus I}$ and $z_1, z_2 \in_{\mathrm{R}} \{0,1\}^I$ independently. It accepts if $f(w \sqcup z_1) = f(w \sqcup z_2)$ and rejects if $f(w \sqcup z_1) \neq f(w \sqcup z_2)$.

Lemma 4 implies that the independence test rejects with probability exactly $\frac{1}{2}\mathrm{Vr}_f(I)$.

Random Partitions Throughout the paper we will use the following notion of a random partition of the set $[n]$ of input coordinates:

Definition 5 A *random partition* of $[n]$ into r subsets $\{I_j\}_{j=1}^r$ is constructed by independently assigning each $i \in [n]$ to a randomly chosen I_j for some $j \in [r]$.

We now define the notion of low- and high-variation subsets with respect to a partition of the set $[n]$ and a parameter $\alpha > 0$.

Definition 6 For $f : \{0,1\}^n \to \{-1,1\}$, a partition of $[n]$ into $\{I_j\}_{j=1}^r$ and a parameter $\alpha > 0$, define $L(\alpha) \stackrel{\text{def}}{=} \{j \in [r] \mid \mathrm{Vr}_f(I_j) < \alpha\}$ (low-variation subsets) and $H(\alpha) \stackrel{\text{def}}{=} [r] \setminus L(\alpha)$ (high-variation subsets). For $j \in [r]$ and $i \in I_j$, if $\mathrm{Vr}_f(i) \geq \alpha$ we say that the variable x_i is a *high-variation element* of I_j.

Finally, the notion of a *well-structured* subset will be important for us:

Definition 7 For $f : \{0,1\}^n \to \{-1,1\}$ and parameters $\alpha, \Delta > 0$ satisfying $\alpha > \Delta$, we say that a subset $I \subseteq [n]$ of coordinates is (α, Δ)-*well structured* if there is an $i \in I$ such that $\mathrm{Vr}_f(i) \geq \alpha$ and $\mathrm{Vr}_f(I \setminus \{i\}) \leq \Delta$.

Note that since $\alpha > \Delta$, by monotonicity, the $i \in I$ in the above definition is unique. Hence, a well-structured subset contains a single high-influence coordinate, while the remaining coordinates have small total variation.

3 The Testing Algorithm *Test-Sparse-Poly*

In this section we present our main testing algorithm and give high-level sketches of the arguments establishing its completeness and soundness. The algorithm, which is called *Test-Sparse-Poly*, takes as input the values $s, \epsilon > 0$ and black-box access to $f : \{0,1\}^n \to \{-1,1\}$. It is presented in full in Fig. 1.

Test-Sparse-Poly is based on the idea that if f is a sparse polynomial, then it only has a small number of "high-influence" variables, and it is close to another sparse polynomial f' that depends only on (some of) those high-influence variables. Roughly speaking, the algorithm works by first isolating the high-influence variables into distinct subsets, and then attempting to exactly learn f'. (This learning is done "implicitly," i.e. without ever explicitly identifying any of the relevant variables for f or f'.)

We now give a more detailed description of the test in tandem with a sketch of why the test is complete, i.e. why it accepts s-sparse polynomials (we give a sketch of the soundness argument in the next subsection). The first thing *Test-Sparse-Poly* does (Step 2) is to randomly partition the variables (coordinates) into $r = \tilde{O}(s^4/\tau)$

Algorithm *Test-Sparse-Poly*(f, s, ϵ)

Desired input: Black-box access to $f : \{0,1\}^n \to \{-1,1\}$; sparsity parameter $s \geq 1$; error parameter $\epsilon > 0$.
Desired output: "*yes*" if f is an s-sparse $GF(2)$ polynomial, *"no"* if f is ϵ-far from every s-sparse $GF(2)$ polynomial.

1. Set $\tau = \epsilon/600, \Delta = \min\{\Delta_0, (\tau/8s^2)(\delta/\ln(2/\delta))\}$, $r = 4Cs/\Delta$ (for a suitable constant C from Theorem 12), where $\Delta_0 \stackrel{\text{def}}{=} \tau/(1600s^3 \log(8s^3/\tau))$ and $\delta \stackrel{\text{def}}{=} 1/(100s \log(8s^3/\tau) Q(s, s\log(8s^3/\tau), \epsilon/4, 1/100))$.
2. Set $\{I_j\}_{j=1}^r$ to be a random partition of $[n]$.
3. Choose α uniformly at random from the set $\mathcal{A}(\tau, \Delta) \stackrel{\text{def}}{=} \{\frac{\tau}{4s^2} + (8\ell - 4)\Delta : 1 \leq \ell \leq K\}$ where K is the largest integer such that $8K\Delta \leq \frac{\tau}{4s^2}$ (so we have $\frac{\tau}{4s^2} + 4\Delta \leq \alpha \leq \frac{\tau}{2s^2} - 4\Delta$).
4. For each subset $I_1, \ldots, I_r$ run the independence test $M \stackrel{\text{def}}{=} \frac{2}{\Delta^2}\ln(200r)$ times and let $\widetilde{\text{Vr}}_f(I_j)$ denote $2\times$ (fraction of the M runs on I_j that the test rejects). If any subset I_j has $\widetilde{\text{Vr}}_f(I_j) \in [\alpha - 2\Delta, \alpha + 3\Delta]$ then exit and return "*no*," otherwise continue.
5. Let $\widetilde{L}(\alpha) \subseteq [r]$ denote $\{j \in [r] : \widetilde{\text{Vr}}_f(I_j) \leq \alpha\}$ and let $\widetilde{H}(\alpha)$ denote $[r] \setminus \widetilde{L}(\alpha)$. Let $\widetilde{f'} : \{0,1\}^n \to \{-1,1\}$ denote the function $f|_{0 \leftarrow \bigcup_{j \in \widetilde{L}(\alpha)} I_j}$.
6. Draw a sample of $m \stackrel{\text{def}}{=} \frac{2}{\epsilon}\ln 12$ uniform random examples from $\{0,1\}^n$ and evaluate both $\widetilde{f'}$ and f on each of these examples. If f and $\widetilde{f'}$ disagree on any of the m examples then exit and return "*no*." If they agree on all examples then continue.
7. Run the learning algorithm *LearnPoly*$'(s, |\widetilde{H}(\alpha)|, \epsilon/4, 1/100)$ from [27] using *SimMQ*$(f, \widetilde{H}(\alpha), \{I_j\}_{j \in \widetilde{H}(\alpha)}, \alpha, \Delta, z, \delta/Q(s, |\widetilde{H}(\alpha)|, \epsilon/4, 1/100))$ to simulate each membership query on a string $z \in \{0,1\}^{|\widetilde{H}(\alpha)|}$ that *LearnPoly*$'$ makes.[a] If *LearnPoly*$'$ returns "not s-sparse" then exit and return "*no*." Otherwise the algorithm terminates successfully; in this case return "*yes*."

[a] See Sect. 4 for detailed explanations of the procedures *LearnPoly*$'$ and *SimMQ* and the function $Q(\cdot,\cdot,\cdot,\cdot)$.

Fig. 1 The algorithm *Test-Sparse-Poly*

subsets. If f is an s-sparse polynomial, then it indeed has few high-influence variables, so with high probability at most one such variable will be present in each subset. In Steps 3 and 4 the algorithm attempts to distinguish subsets that contain a high-influence variable from subsets that do not; this is done by using the independence test to estimate the variation of each subset (see Lemma 4). To show that, for sparse polynomials, this estimate can correctly identify the subsets that have a high-influence variable, we must show that if f is an s-sparse polynomial then with high probability there is an easy-to-find "gap" such that subsets with a high-influence variable have variation above the gap, and subsets with no high-influence variable have variation below the gap. This is established by Theorem 12.

Once the high-variation and low-variation subsets have been identified, intuitively we would like to focus our attention on the high-influence variables. Thus, Step 5 of the algorithm defines a function $\widetilde{f'}$ which "zeroes out" all of the variables in all

low-variation subsets.[4] Note that if the original function f is an s-sparse polynomial, then $\widetilde{f}'$ will be one too. Step 6 of *Test-Sparse-Poly* checks that f is close to $\widetilde{f}'$; Theorem 12 establishes that this is indeed the case if f is an s-sparse polynomial.

The final step of *Test-Sparse-Poly* is to run the algorithm *LearnPoly*$'$ of [27] to learn a sparse polynomial, which we call $\widetilde{f}''$, which is isomorphic to $\widetilde{f}'$ but is defined only over the high-influence variables of f (recall that there is at most one from each high-variation subset). The overall *Test-Sparse-Poly* algorithm accepts f if and only if *LearnPoly*$'$ successfully returns a final hypothesis (i.e. does not halt and output "fail"). The membership queries that the [27] algorithm requires are simulated using the *SimMQ* procedure, which we define in detail in Sect. 4. Theorem 12 ensures that for f an s-sparse polynomial, all of the subsets I_j that "survive" into $\widetilde{f}'$ are well-structured (see Definition 7); as we show later, this condition is sufficient to ensure that *SimMQ* can successfully simulate membership queries to $\widetilde{f}''$. Thus, for f an s-sparse polynomial the *LearnPoly*$'$ algorithm can run successfully, and the test will accept.

3.1 Sketch of Soundness

Here, we briefly argue that if *Test-Sparse-Poly* accepts f with high probability, then f must be close to some s-sparse polynomial. Note that if f passes Step 4, then *Test-Sparse-Poly* must have obtained a partition of variables into "high-variation" subsets and "low-variation" subsets. If f passes Step 6, then it must moreover be the case that f is close to the function $\widetilde{f}'$ obtained by zeroing out the low-variation subsets.

In the last step, *Test-Sparse-Poly* attempts to run the *LearnPoly*$'$ algorithm using $\widetilde{f}'$ and the high-variation subsets; in the course of doing this, it makes calls to *SimMQ*. Since f could be an arbitrary function, we do not know whether each high-variation subset has at most one variable relevant to $\widetilde{f}'$ (as would be the case, by Theorem 12, if f were an s-sparse polynomial). However, we are able to show (Lemma 24) that, if with high probability all calls to the *SimMQ* routine are answered without its ever returning "fail," then $\widetilde{f}'$ must be close to a junta g whose relevant variables are the individual "highest-influence" variables in each of the high-variation subsets. Now, given that *LearnPoly*$'$ halts successfully, it must be the case that it constructs a final hypothesis h that is itself an s-sparse polynomial and that agrees with many calls to *SimMQ* on random examples. Lemma 25 states that, in this event, h must be close to g, hence close to $\widetilde{f}'$, and hence close to f.

4 The *LearnPoly*$'$ Algorithm

In this section we describe the procedure *LearnPoly*$'$, thus completing our description of *Test-Sparse-Poly*. We close this section with a coarse analysis of the overall query

[4] The difference between $\widetilde{f}'$ and f' from Theorem 12 is that $\widetilde{f}'$ is defined by zeroing out variables in subsets which *Test-Sparse-Poly* empirically determines to have low variation, whereas f' is defined by zeroing out variables in subsets that actually have low variation. Thus $\widetilde{f}'$ is the "effective" version of f' that the algorithm can actually obtain. Theorem 12 will imply that if f is an s-sparse polynomial, then with high probability $\widetilde{f}'$ and f' are the same.

Algorithm *Set-High-Influence-Variable*$(f, I, \alpha, \Delta, b, \delta)$

Desired input: Black-box access to $f : \{0,1\}^n \to \{-1,1\}$; (α, Δ)-well-structured set $I \subseteq [n]$; bit $b \in \{0,1\}$; failure parameter δ.
Desired output: Assignment $w \in \{0,1\}^I$ to the variables in I such that the high-variation coordinate w_i equals b with probability $1-\delta$.

1. Draw x uniformly from $\{0,1\}^I$. Define $I^0 \stackrel{\text{def}}{=} \{j \in I : x_j = 0\}$ and $I^1 \stackrel{\text{def}}{=} \{j \in I : x_j = 1\}$.
2. Apply $c = \frac{2}{\alpha}\ln(\frac{2}{\delta})$ iterations of the *independence test* to (f, I^0). If any of the c iterations reject, mark I^0. Do the same for (f, I^1).
3. If both or neither of I^0 and I^1 are marked, stop and output "fail."
4. If I^b is marked then return the assignment $w = x$. Otherwise return the assignment $w = \overline{x}$ (the bitwise negation of x).

Fig. 2 The subroutine *Set-High-Influence-Variable*

complexity of *Test-Sparse-Poly* which establishes that it makes $\text{poly}(s, \frac{1}{\epsilon})$ queries to f. (We have made no effort to optimize or even determine the precise polynomial.)

Our test runs the *LearnPoly′* learning algorithm using simulated membership queries which are performed by a procedure called *SimMQ*, which in turn uses a subroutine called *Set-High-Influence-Variables*. We give a "bottom-up" description by first describing *Set-High-Influence-Variables* and then *SimMQ*. In Sect. 4.1 we describe *LearnPoly′* and explain how it uses *SimMQ*.

The procedure *Set-High-Influence-Variable* (*SHIV*) is presented in Fig. 2. The idea of this procedure is that when it is run on a well-structured subset of variables I, it returns an assignment in which the high-variation variable is set to the desired bit value. Intuitively, the executions of the independence test in the procedure are used to determine whether the high-variation variable $i \in I$ is set to 0 or 1 under the assignment x; depending on whether this setting agrees with the desired value, the algorithm either returns x or the bitwise negation of x. The following simple lemma shows that, for suitable values of the parameters, the procedure indeed performs as desired.

Lemma 8 *Let f, I, α, Δ be such that I is (α, Δ)-well-structured with $\Delta \le \alpha\delta/(2\ln(2/\delta))$, and let w_i be the coordinate in I with high variation in I. Then with probability at least $1-\delta$, the output of SHIV$(f, I, \alpha, \Delta, b, \delta)$ is an assignment $w \in \{0,1\}^I$ which has $w_i = b$.*

Proof We assume that I^b contains the high-variation variable i (the other case being very similar). Recall that by Lemma 4, each run of the independence test on I^b rejects with probability $\frac{1}{2}\text{Vr}_f(I^b)$; by Lemma 3 (monotonicity) this is at least $\frac{1}{2}\text{Vr}_f(i) \ge \alpha/2$. So the probability that I^b is not marked even once after c iterations of the independence test is at most $(1-\alpha/2)^c \le \delta/2$, by our choice of c. Similarly, the probability that $I^{\overline{b}}$ is ever marked during c iterations of the independence test is at most $c(\Delta/2) \le \delta/2$, by the condition of the lemma. Thus, the probability of failing at Step 3 of *SHIV* is at most δ, and since $i \in I^b$, the assignment w sets variable i correctly in Step 4. □

Algorithm *SimMQ*$(f, H, \{I_j\}_{j\in H}, \alpha, \Delta, z, \delta)$

Desired input: Black-box access to $f : \{0,1\}^n \to \{-1,1\}$; subset $H \subseteq [r]$; disjoint subsets $\{I_j\}_{j\in H}$ of $[n]$; parameters $\alpha > \Delta$; string $z \in \{0,1\}^{|H|}$; failure probability δ.
Desired output: Bit b which, with probability $1-\delta$ is the value of f' on a random assignment x in which each high-variation variable $i \in I_j$ $(j \in H)$ is set according to z.

1. For each $j \in H$, call *Set-High-Influence-Variable*$(f, I_j, \alpha, \Delta, z_j, \delta/|H|)$ and get back an assignment (call it w^j) to the variables in I_j.
2. Construct $x \in \{0,1\}^n$ as follows: for each $j \in H$, set the variables in I_j according to w^j. This defines x_i for all $i \in \bigcup_{j\in H} I_j$. Set $x_i = 0$ for all other $i \in [n]$.
3. Return $b = f(x)$.

Fig. 3 The subroutine *SimMQ*

For the soundness proof, we will require the following lemma which specifies the behavior of *SHIV* when it is called with parameters α, Δ that do not quite match the real values α', Δ' for which I is (α', Δ')-well-structured:

Lemma 9 *If I is (α', Δ')-well-structured (but not necessarily (α, Δ)-well-structured), then the probability that SHIV$(f, I, \alpha, \Delta, b, \delta)$ passes (i.e. does not output "fail") and sets the high variation variable incorrectly is at most $(\delta/2)^{\alpha'/\alpha} \cdot (1/\alpha) \cdot \Delta' \cdot \ln(2/\delta)$.*

Proof The only way for *SHIV* to pass with an incorrect setting of the high-variation variable i is if it fails to mark the subset containing i for c iterations of the independence test, and marks the other subset at least once. Since $\mathrm{Vr}(i) > \alpha'$ and $\mathrm{Vr}(I \setminus i) < \Delta'$, the probability of this occurring is at most $(1-\alpha'/2)^c \cdot \Delta' \cdot c/2$. Since *SHIV* is called with failure parameter δ, c is set to $\frac{2}{\alpha}\ln\frac{2}{\delta}$. □

Figure 3 gives the *SimMQ* procedure. The high-level idea is as follows: we have a function f and a collection $\{I_j\}_{j\in H}$ of disjoint well-structured subsets of variables. *SimMQ* takes as input a string z of length $|H|$ which specifies a desired setting for each high-variation variable in each I_j $(j \in H)$. *SimMQ* constructs a random assignment $x \in \{0,1\}^n$ such that the high-variation variable in each I_j $(j \in H)$ is set in the desired way in x, and it returns the value $f'(x)$.

In the completeness proof we shall show that if f is an s-sparse polynomial, then w.h.p. every call to *SimMQ* that the test performs correctly simulates a membership query to a certain s-sparse polynomial $\widetilde{f}' : \{0,1\}^{|\widetilde{H}(\alpha)|} \to \{-1,1\}$. In the soundness proof we will show that if w.h.p. no call to *SimMQ* outputs 'fail,' then f must be close to a junta which agrees with many of the queries returned by *SimMQ*.

4.1 The *LearnPoly*′ Procedure

4.1.1 Background on Schapire and Sellie's Algorithm

In [27] Schapire and Sellie gave an algorithm, which we refer to as *LearnPoly*, for exactly learning s-sparse $GF(2)$ polynomials using membership queries (i.e. black-box

queries) and equivalence queries. Their algorithm is *proper*; this means that every equivalence query the algorithm makes (including the final hypothesis of the algorithm) is an s-sparse polynomial. (We shall see that it is indeed crucial for our purposes that the algorithm is proper.) Recall that in an equivalence query the learning algorithm proposes a hypothesis h to the oracle: if h is logically equivalent to the target function being learned then the response is "correct" and learning ends successfully, otherwise the response is "no" and the learner is given a counterexample x such that $h(x) \neq f(x)$.

Schapire and Sellie proved the following about their algorithm:

Theorem 10 ([27], Theorem 10) *Algorithm LearnPoly is a proper exact learning algorithm for the class of s-sparse $GF(2)$ polynomials over $\{0,1\}^n$. The algorithm runs in poly(n,s) time and makes at most* poly(n,s) *membership queries and at most $ns+2$ equivalence queries.*

We can easily also characterize the behavior of *LearnPoly* if it is run on a function f that is not an s-sparse polynomial. In this case, since the algorithm is proper all of its equivalence queries have s-sparse polynomials as their hypotheses, and consequently no equivalence query will ever be answered "correct." So if the $(ns+2)$th equivalence query is not answered "correct," the algorithm may infer that the target function is not an s-sparse polynomial, and it returns "not s-sparse."

As pointed out in [27], a well-known result due to Angluin [2] says that in a Probably Approximately Correct or PAC setting (where there is a distribution $\mathcal{D}$ over examples and the goal is to construct an ϵ-accurate hypothesis with respect to that distribution), equivalence queries can be straightforwardly simulated using random examples. This is done simply by drawing a sufficiently large sample of random examples for each equivalence query and evaluting both the hypothesis h and the target function f on each point in the sample. This either yields a counterexample (which simulates an equivalence query), or if no counterexample is obtained then simple arguments show that for a large enough ($O(\log(1/\delta)/\epsilon)$-size) sample, with probability $1-\delta$ the functions f and h must be ϵ-close under the distribution $\mathcal{D}$, which is the success criterion for PAC learning. This directly gives the following corollary of Theorem 10:

Corollary 11 *There is a uniform distribution membership query proper learning algorithm which makes $Q(s,n,\epsilon,\delta) \stackrel{\text{def}}{=} \text{poly}(s,n,1/\epsilon,\log(1/\delta))$ membership queries and runs in* poly(Q) *time to learn s-sparse polynomials over $\{0,1\}^n$ to accuracy ϵ and confidence $1-\delta$ under the uniform distribution.*

We shall refer to this algorithm as *LearnPoly*$'(s,n,\epsilon,\delta)$.

As stated in Fig. 1, the *Test-Sparse-Poly* algorithm runs *LearnPoly*$'(s, |\widetilde{H}(\alpha)|, \epsilon/4, 1/100)$ using *SimMQ*$(f, \widetilde{H}(\alpha), \{I_j\}_{j\in\widetilde{H}(\alpha)}, \alpha, \Delta, z, 1/(100Q(s, |\widetilde{H}(\alpha)|, z, 1/100)))$ to simulate each membership query on an input string $z \in \{0,1\}^{|\widetilde{H}(\alpha)|}$. Thus the algorithm is being run over a domain of $|\widetilde{H}(\alpha)|$ variables. Since we certainly have $|\widetilde{H}(\alpha)| \leq r \leq \text{poly}(s, \frac{1}{\epsilon})$, Corollary 11 gives that *LearnPoly*$'$ makes at most poly$(s, \frac{1}{\epsilon})$ many calls to *SimMQ*. From this point, by inspection of *SimMQ*,

SHIV and *Test-Sparse-Poly*, it is straightforward to verify that *Test-Sparse-Poly* indeed makes $\text{poly}(s, \frac{1}{\epsilon})$ many queries to f and runs in time $\text{poly}(s, \frac{1}{\epsilon})$ as claimed in Theorem 1. Thus, to prove Theorem 1 it remains only to establish correctness of the test.

5 On Restrictions which Simplify Sparse Polynomials

This section presents Theorem 12, which is at the heart of the completeness proof for our test. Before we proceed with the formal statement, we give an intuitive explanation.

Roughly speaking the theorem is as follows: Consider any s-sparse $GF(2)$ polynomial p. Suppose that its coordinates (variables) are randomly partitioned into $r = \text{poly}(s)$ many subsets $\{I_j\}_{j=1}^r$. The first two statements say that (w.h.p.) a randomly chosen "threshold value" $\alpha \approx 1/\text{poly}(s)$ will have the property that no single coordinate i, $i \in [n]$, or subset I_j, $j \in [r]$, has $\text{Vr}_p(i)$ or $\text{Vr}_p(I_j)$ "too close" to α. Moreover, the high-variation subsets (w.r.t. α) are precisely those that contain a single high variation element i (i.e. $\text{Vr}_p(i) \geq \alpha$), and in fact each such subset I_j is well-structured (part 3). Also, the number of such high-variation subsets is "small" (part 4). Finally, let p' be the restriction of p obtained by setting all variables in the low-variation subsets to 0. Then, p' has some "nice" structure: its relevant variables are spread out (at most) one per high-variation subset (part 5), and it is close to p (part 6).

Theorem 12 *Let $p : \{0,1\}^n \to \{-1,1\}$ be an s-sparse polynomial. Fix $\tau \in (0,1)$ and Δ such that $\Delta \leq \Delta_0 \stackrel{\text{def}}{=} \tau/(1600s^3 \log(8s^3/\tau))$ and $\Delta = \text{poly}(\tau/s)$. Let $r \stackrel{\text{def}}{=} 4Cs/\Delta$, for a suitably large constant C. Let $\{I_j\}_{j=1}^r$ be a random partition of $[n]$. Choose α uniformly at random from the set $\mathcal{A}(\tau,\Delta) \stackrel{\text{def}}{=} \{\frac{\tau}{4s^2} + (8\ell - 4)\Delta : \ell \in [K]\}$ where K is the largest integer such that $8K\Delta \leq \frac{\tau}{4s^2}$. Then with probability at least $9/10$ (over the choice of α and $\{I_j\}_{j=1}^r$), all of the following statements hold:*

1. *Every variable x_i, $i \in [n]$, has $\text{Vr}_p(i) \notin [\alpha - 4\Delta, \alpha + 4\Delta]$.*
2. *Every subset I_j, $j \in [r]$, has $\text{Vr}_p(I_j) \notin [\alpha - 3\Delta, \alpha + 4\Delta]$.*
3. *For every $j \in H(\alpha)$, I_j is (α,Δ)-well structured.*
4. *$|H(\alpha)| \leq s \log(8s^3/\tau)$.*

Let $p' \stackrel{\text{def}}{=} p|_{0 \leftarrow \bigcup_{j \in L(\alpha)} I_j}$ (the restriction obtained by fixing all variables in low-variation subsets to 0).

5. *For every $j \in H(\alpha)$, p' has at most one relevant variable in I_j (hence p' is a $|H(\alpha)|$-junta).*
6. *The function p' is τ-close to p.*

In Sect. 5.1 we prove some useful preliminary lemmas about the variation of individual variables in sparse polynomials. In Sect. 5.2 we extend this analysis to get high-probability statements about variation of subsets $\{I_j\}_{j=1}^r$ in a random partition. We put the pieces together to finish the proof of Theorem 12 in Sect. 5.3.

Throughout this section the parameters τ, Δ, r and α are all as defined in Theorem 12.

5.1 The Influence of Variables in s-Sparse Polynomials

We start with a simple lemma stating that only a small number of variables can have large variation:

Lemma 13 *Let $p : \{0,1\}^n \to \{-1,1\}$ be an s-sparse polynomial. For any $\delta > 0$, there are at most $s \log(2s/\delta)$ many variables x_i that have $\mathrm{Vr}_p(i) \geq \delta$.*

Proof Any variable x_i with $\mathrm{Vr}_p(i) \geq \delta$ must occur in some term of length at most $\log(2s/\delta)$. (Otherwise each occurrence of x_i would contribute less than δ/s to the variation of the ith coordinate, and since there are at most s terms this would imply $\mathrm{Vr}_p(i) < s \cdot (\delta/s) = \delta$.) Since at most $s \log(2s/\delta)$ distinct variables can occur in terms of length at most $\log(2s/\delta)$, the lemma follows. □

We next prove that with high probability, we can identify a real interval such that no coordinate x_i has variation in that interval. The following lemma essentially proves the first part of Theorem 12.

Lemma 14 *With probability at least $96/100$ over the choice of α, no variable x_i has $\mathrm{Vr}_p(i) \in [\alpha - 4\Delta, \alpha + 4\Delta]$.*

Proof The uniform random variable α has support $\mathcal{A}(\tau, \Delta)$ of size no less than $50s \log(8s^3/\tau)$. Each possible value of α defines the interval of variations $[\alpha - 4\Delta, \alpha + 4\Delta]$. Note that $\alpha - 4\Delta \geq \tau/(4s^2)$. In other words, the only variables which could lie in $[\alpha - 4\Delta, \alpha + 4\Delta]$ are those with variation at least $\tau/(4s^2)$. By Lemma 13 there are at most $k \stackrel{\text{def}}{=} s \log(8s^3/\tau)$ such candidate variables. Since we have at least $50k$ intervals (two consecutive such intervals overlap at a single point) and at most k candidate variables, at least $48k$ intervals will be empty. □

Lemma 13 is based on the observation that, in a sparse polynomial, a variable with "high" influence (variation) must occur in some "short" term. The following lemma is in some sense a quantitative converse: it states that a variable with "small" influence can only appear in "long" terms.

Lemma 15 *Let $p : \{0,1\}^n \to \{-1,1\}$ be an s-sparse polynomial. Suppose that i is such that $\mathrm{Vr}_p(i) < \tau/(s^2 + s)$. Then the variable x_i appears only in terms of length greater than $\log(s/\tau)$.*

Proof By contradiction. Assuming that x_i appears in some term of length at most $\log(s/\tau)$, we will show that $\mathrm{Vr}_p(i) \geq \tau/(s^2 + s)$. Let T be a shortest term that x_i appears in. The function p can be uniquely decomposed as follows: $p(x_1, x_2, \ldots, x_n) = x_i \cdot (T' + p_1) + p_2$, where $T = x_i \cdot T'$, the term T' has length less than $\log(s/\tau)$ and does not depend on x_i, and p_1, p_2 are s-sparse polynomials that do not depend on x_i.

Observe that since T is a shortest term that contains x_i, the polynomial p_1 does not contain the constant term 1.

Since T' contains fewer than $\log(s/\tau)$ many variables, it evaluates to 1 on at least a τ/s fraction of all inputs. The partial assignment that sets all the variables in T' to 1 induces an s-sparse polynomial p'_1 (the restriction of p_1 according to the partial assignment). Now observe that p'_1 still does not contain the constant term 1 (for since each term in p_1 is of length at least the length of T', no term in p_1 is a subset of the variables in T'). We now recall the following (nontrivial) result of Karpinski and Luby [17]:

Claim 16 ([17], Corollary 1) *Let g be an s-sparse multivariate $GF(2)$ polynomial which does not contain the constant-1 term. Then $g(x) = 0$ for at least a $1/(s+1)$ fraction of all inputs.*

Applying this corollary to the polynomial p'_1, we have that p'_1 is 0 on at least a $1/(s+1)$ fraction of its inputs. Therefore, the polynomial $T' + p_1$ is 1 on at least a $(\tau/s) \cdot 1/(s+1)$ fraction of all inputs in $\{0,1\}^n$; this in turn implies that $\mathrm{Vr}_p(i) \geq (\tau/s) \cdot 1/(s+1) = \tau/(s^2+s)$. □

By a simple application of Lemma 15 we can show that setting low-variation variables to zero does not change the polynomial by much:

Lemma 17 *Let $p : \{0,1\}^n \to \{-1,1\}$ be an s-sparse polynomial. Let g be a function obtained from p by setting to 0 some subset of variables all of which have $\mathrm{Vr}_p(i) < \tau/(2s^2)$. Then g and p are τ-close.*

Proof Setting a variable to 0 removes all the terms that contain it from p. By Lemma 15, doing this only removes terms of length greater than $\log(s/\tau)$. Removing one such term changes the function on at most a τ/s fraction of the inputs. Since there are at most s terms in total, the lemma follows by a union bound. □

5.2 Partitioning Variables into Random Subsets

The following lemma is at the heart of Theorem 12. The lemma states that when we randomly partition the variables (coordinates) into subsets, (i) each subset gets at most one "high-influence" variable (the term "high-influence" here means relative to an appropriate threshold value $t \ll \alpha$), and (ii) the remaining (low-influence) variables (w.r.t. t) have a "very small" contribution to the subset's total variation.

The first part of the lemma follows easily from a birthday-paradox type argument, since there are many more subsets than high-influence variables. As intuition for the second part, we note that in expectation, the total variation of each subset is very small. A more careful argument lets us argue that the total contribution of the low-influence variables in a given subset is unlikely to highly exceed its expectation.

Lemma 18 *Fix a value of α satisfying the first statement of Theorem 12. Let $t \stackrel{\text{def}}{=} \Delta\tau/(4C's)$, where C' is a suitably large constant. Then with probability $99/100$ over the random partition the following statements hold true*:

- *For every* $j \in [r]$, I_j *contains at most one variable* x_i *with* $\mathrm{Vr}_p(i) > t$.
- *Let* $I_j^{\leq t} \stackrel{\text{def}}{=} \{i \in I_j \mid \mathrm{Vr}_p(i) \leq t\}$. *Then, for all* $j \in [r]$, $\mathrm{Vr}_p(I_j^{\leq t}) \leq \Delta$.

Proof We show that each statement of the lemma fails independently with probability at most $1/200$ from which the lemma follows.

By Lemma 13 there are at most $k \stackrel{\text{def}}{=} s\log(2s/t)$ coordinates in $[n]$ with variation more than t. A standard argument yields that the probability there exists a subset I_j with more than one such variable is at most k^2/r. It is easy to verify that this is less than $1/200$, as long as C is large enough relative to C'. Therefore, with probability at least $199/200$, every subset contains at most one variable with variation greater than t. So the first statement fails with probability no more than $1/200$.

Now for the second statement. Consider a fixed subset I_j. We analyze the contribution of variables in $I_j^{\leq t}$ to the total variation $\mathrm{Vr}_p(I_j)$. We will show that with high probability the contribution of these variables is at most Δ.

Let $S = \{i \in [n] \mid \mathrm{Vr}_p(i) \leq t\}$ and renumber the coordinates such that $S = [k']$. Each variable x_i, $i \in S$, is contained in I_j independently with probability $1/r$. Let $X_1, \ldots, X_{k'}$ be the corresponding independent Bernoulli random variables. Recall that, by sub-additivity, the variation of $I_j^{\leq t}$ is upper bounded by $X = \sum_{i=1}^{k'} \mathrm{Vr}_p(i) \cdot X_i$. It thus suffices to upper bound the probability $\Pr[X > \Delta]$. Note that $\mathbb{E}[X] = \sum_{i=1}^{k'} \mathrm{Vr}_p(i) \cdot \mathbb{E}[X_i] = (1/r) \cdot \sum_{i=1}^{k'} \mathrm{Vr}_p(i) \leq (s/r)$, since $\sum_{i=1}^{k'} \mathrm{Vr}_p(i) \leq \sum_{i=1}^{n} \mathrm{Vr}_p(i) \leq s$ (the last inequality here is easily seen to follow from the fact that p is an s-sparse $GF(2)$ polynomial). To finish the proof, we need the following version of the Chernoff bound:

Fact 19 [21] *For* $k' \in \mathbb{N}^*$, *let* $\alpha_1, \ldots, \alpha_{k'} \in [0,1]$ *and let* $X_1, \ldots, X_{k'}$ *be independent Bernoulli trials. Let* $X' = \sum_{i=1}^{k'} \alpha_i X_i$ *and* $\mu \stackrel{\text{def}}{=} \mathbb{E}[X'] \geq 0$. *Then for any* $\gamma > 1$ *we have* $\Pr[X' > \gamma \cdot \mu] < (\frac{e^{\gamma-1}}{\gamma^\gamma})^\mu$.

We apply the above bound for the X_i's with $\alpha_i = \mathrm{Vr}_p(i)/t \in [0,1]$. (Recall that the coordinates in S have variation at most t.) We have $\mu = \mathbb{E}[X'] = \mathbb{E}[X]/t \leq s/(rt) = C's/C\tau$, and we are interested in the event $\{X > \Delta\} \equiv \{X' > \Delta/t\}$. Note that $\Delta/t = 4C's/\tau$. Hence, $\gamma \geq 4C$ and the above bound implies that $\Pr[X > \Delta] < (e/(4C))^{4C's/\tau} < (1/4C^4)^{C's/\tau}$.

Therefore, for a fixed subset I_j, we have $\Pr[\mathrm{Vr}_p(I_j^{\leq t}) > \Delta] < (1/4C^4)^{C's/\tau}$. By a union bound, we conclude that this happens in every subset with failure probability at most $r \cdot (1/4C^4)^{C's/\tau}$. This is less than $1/200$ as long as C' is a large enough absolute constant (independent of C), which completes the proof. □

Next we show that by "zeroing out" the variables in low-variation subsets, we are likely to "kill" all terms in p that contain a low-influence variable.

Lemma 20 *With probability at least* $99/100$ *over the random partition, every monomial of* p *containing a variable with influence at most* α *has at least one of its variables in* $\bigcup_{j \in L(\alpha)} I_j$.

Proof By Lemma 13 there are at most $b = s\log(8s^3/\tau)$ variables with influence more than α. Thus, no matter the partition, at most b subsets from $\{I_j\}_{j=1}^r$ contain such variables. Fix a low-influence variable (influence at most α) from every monomial containing such a variable. For each fixed variable, the probability that it ends up in the same subset as a high-influence variable is at most b/r. Union bounding over each of the (at most s) monomials, the failure probability of the lemma is upper bounded by $sb/r < 1/100$. □

5.3 Proof of Theorem 12

Proof of Theorem 12 We prove each statement in turn. The first statement of the theorem is implied by Lemma 14. (Note that, as expected, the validity of this statement does not depend on the random partition.)

We claim that statements 2–5 essentially follow from Lemma 18. (In contrast, the validity of these statements crucially depends on the random partition.)

Let us first prove the third statement. We want to show that (w.h.p. over the choice of α and $\{I_j\}_{j=1}^r$) for every $j \in H(\alpha)$, (i) there exists a *unique* $i_j \in I_j$ such that $\mathrm{Vr}_p(i_j) \geq \alpha$ and (ii) that $\mathrm{Vr}_p(I_j \setminus \{i_j\}) \leq \Delta$. Fix some $j \in H(\alpha)$. By Lemma 18, for a given value of α satisfying the first statement of the theorem, we have: (i') I_j contains at most one variable x_{i_j} with $\mathrm{Vr}_p(i_j) > t$ and (ii') $\mathrm{Vr}_p(I_j \setminus \{i_j\}) \leq \Delta$. Since $t < \tau/4s^2 < \alpha$ (with probability 1), (i') clearly implies that, if I_j has a high-variation element (w.r.t. α), then it is unique. In fact, we claim that $\mathrm{Vr}_p(i_j) \geq \alpha$. For otherwise, by sub-additivity of variation, we would have $\mathrm{Vr}_p(I_j) \leq \mathrm{Vr}_p(I_j \setminus \{i_j\}) + \mathrm{Vr}_p(i_j) \leq \Delta + \alpha - 4\Delta = \alpha - 3\Delta < \alpha$, which contradicts the assumption that $j \in H(\alpha)$. Note that we have used the fact that α satisfies the first statement of the theorem, that is $\mathrm{Vr}_p(i_j) < \alpha \Rightarrow \mathrm{Vr}_p(i_j) < \alpha - 4\Delta$. Hence, for a "good" value of α (one satisfying the first statement of the theorem), the third statement is satisfied with probability at least 99/100 over the random partition. By Lemma 14, a "good" value of α is chosen with probability 96/100. By independence, the conclusions of Lemmas 14 and 18 hold simultaneously with probability more than 9/10.

We now establish the second statement. We assume as before that α is a "good" value. Consider a fixed subset I_j, $j \in [r]$. If $j \in H(\alpha)$ (i.e. I_j is a high-variation subset) then by Lemma 18, with probability at least 99/100 (over the random partition), there exists $i_j \in I_j$ such that $\mathrm{Vr}_p(i_j) \geq \alpha + 4\Delta$. The monotonicity of variation yields $\mathrm{Vr}_p(I_j) \geq \mathrm{Vr}_p(i_j) \geq \alpha + 4\Delta$. If $j \in L(\alpha)$ then I_j contains no high-variation variable, i.e. its maximum variation element has variation at most $\alpha - 4\Delta$ and by the second part of Lemma 18 the remaining variables contribute at most Δ to its total variation. Hence, by sub-additivity we have that $\mathrm{Vr}_p(I_j) \leq \alpha - 3\Delta$. Since a "good" value of α is chosen with probability 96/100, the desired statement follows.

The fourth statement follows from the aforementioned and the fact that there exist at most $s\log(8s^3/\tau)$ variables with variation at least α (as follows from Lemma 13, given that $\alpha > \tau/(4s^2)$).

Now for the fifth statement. Lemma 20 and monotonicity imply that the only variables that remain relevant in p' are (some of) those with high influence (at least α) in p, and, as argued above, each high-variation subset I_j contains at most one such

variable. By a union bound, the conclusion of Lemma 20 holds simultaneously with the conclusions of Lemmas 14 and 18 with probability at least $9/10$.

The sixth statement (that p and p' are τ-close) is a consequence of Lemma 17 (since p' is obtained from p by setting to 0 variables with variation less than $\alpha < \tau/(2s^2)$). This concludes the proof of Theorem 12. □

6 Completeness of the Test

In this section we show that *Test-Sparse-Poly* is complete:

Theorem 21 *Suppose f is an s-sparse $GF(2)$ polynomial. Then Test-Sparse-Poly accepts f with probability at least* $2/3$.

Proof Fix f to be an s-sparse $GF(2)$ polynomial over $\{0,1\}^n$. We will first show that, with high probability, *Test-Sparse-Poly* given access to f passes Steps 4–6 and the randomly chosen subsets of variables $I_1, \ldots, I_r$ satisfy certain properties (based on Theorem 12). We then use these properties to prove Lemma 22, which states that all the calls to *Sim-MQ* in Step 7 are likely to simulate calls to a function f'' that is an s-sparse polynomial over few variables. (We will define f'' below—roughly speaking, f'' is the junta isomorphic to the restricted function f' obtained by assigning 0 to variables in low-influence buckets.) Theorem 21 follows easily from Lemma 22 and the guarantee of the learning algorithm *LearnPoly*$'$.

We now establish that *Test-Sparse-Poly* is likely to make it to Step 7 and that the randomly chosen subsets are likely to have useful properties. By the choice of the Δ and r parameters in Step 1 of *Test-Sparse-Poly* we may apply Theorem 12, so with failure probability at most $1/10$ over the choice of α and $I_1, \ldots, I_r$ in Steps 2 and 3, statements 1–6 of Theorem 12 all hold. We shall write f' to denote $f|_{0 \leftarrow \bigcup_{j \in L(\alpha)} I_j}$. Note that at each successive stage of the proof we shall assume that the "failure probability" events do not occur, i.e. henceforth we shall assume that statements 1–6 all hold for f; we take a union bound over all failure probabilities at the end of the proof.

Now consider the M executions of the independence test for a given fixed I_j in Step 4. Lemma 4 gives that each run rejects with probability $\frac{1}{2}\mathrm{Vr}_f(I_j)$. A standard Hoeffding bound implies that for the algorithm's choice of $M = \frac{2}{\Delta^2}\ln(200r)$, the value $\widetilde{\mathrm{Vr}}_f(I_j)$ obtained in Step 4 is within $\pm\Delta$ of the true value $\mathrm{Vr}_f(I_j)$ with failure probability at most $\frac{1}{100r}$. A union bound over all $j \in [r]$ gives that with failure probability at most $1/100$, we have that each $\widetilde{\mathrm{Vr}}_f(I_j)$ is within an additive $\pm\Delta$ of the true value $\mathrm{Vr}_f(I_j)$. This means that (by statement 2 of Theorem 12) every I_j has $\widetilde{\mathrm{Vr}}_f(I_j) \notin [\alpha - 2\Delta, \alpha + 3\Delta]$, and hence in Step 5 of the test, the sets $\widetilde{L}(\alpha)$ and $\widetilde{H}(\alpha)$ are identical to $L(\alpha)$ and $H(\alpha)$ respectively, which in turn means that the function $\widetilde{f'}$ defined in Step 5 is identical to f' defined above.

We now turn to Step 6 of the test. By statement 6 of Theorem 12 we have that f and f' disagree on at most a τ fraction of inputs. A union bound over the m random examples drawn in Step 6 implies that with failure probability at most $\tau m < 1/100$ the test proceeds to Step 7.

By statement 3 of Theorem 12 we have that each I_j, $j \in \widetilde{H}(\alpha) \equiv H(\alpha)$, contains precisely one high-variation element i_j (i.e. which satisfies $\mathrm{Vr}_f(i_j) \geq \alpha$), and these are all of the high-variation elements. Consider the set of these $|\widetilde{H}(\alpha)|$ high-variation variables; statement 5 of Theorem 12 implies that these are the only variables which f' can depend on (it is possible that it does not depend on some of these variables). Let us write f'' to denote the function $f'' : \{0,1\}^{|\widetilde{H}(\alpha)|} \to \{-1,1\}$ corresponding to f' but whose input variables are these $|\widetilde{H}(\alpha)|$ high-variation variables in f, one per I_j for each $j \in \widetilde{H}(\alpha)$. We thus have that f'' is isomorphic to f' (obtained from f' by discarding irrelevant variables).

The main idea behind the rest of the completeness proof is that in Step 7 of *Test-Sparse-Poly*, the learning algorithm *LearnPoly*′ is being run with target function f''. Since f'' is isomorphic to f', which is an s-sparse polynomial (since it is a restriction of an s-sparse polynomial f), with high probability *LearnPoly*′ will run successfully and the test will accept. To show that this is what actually happens, we must show that with high probability each call to *SimMQ* which *LearnPoly*′ makes correctly simulates the corresponding membership query to f''. This is established by the following lemma:

Lemma 22 *With total failure probability at most* $1/100$, *each of the* $Q(s, |\widetilde{H}(\alpha)|, \epsilon/4, 1/100)$ *calls to* $\mathit{SimMQ}(f, \widetilde{H}(\alpha), \{I_j\}_{j\in\widetilde{H}(\alpha)}, \alpha, \Delta, z, 1/(100Q(s, |\widetilde{H}(\alpha)|, \epsilon/4, 1/100)))$ *that LearnPoly*′ *makes in Step* 7 *of Test-Sparse-Poly returns the correct value of* $f''(z)$.

Proof Consider a single call to the procedure $\mathit{SimMQ}(f, \widetilde{H}(\alpha), \{I_j\}_{j\in\widetilde{H}(\alpha)}, \alpha, \Delta, z, 1/(100Q(s, |\widetilde{H}(\alpha)|, \epsilon/4, 1/100)))$ made by *LearnPoly*′. We show that with failure probability at most $\delta' \stackrel{\text{def}}{=} 1/(100Q(s, |\widetilde{H}(\alpha)|, \epsilon/4, 1/100)$ this call returns the value $f''(z)$, and the lemma then follows by a union bound over the $Q(s, |\widetilde{H}(\alpha)|, \epsilon/4, 1/100)$ many calls to *SimMQ*.

This call to *SimMQ* makes $|\widetilde{H}(\alpha)|$ calls to $\mathit{SHIV}(f, I_j, \alpha, \Delta, z_j, \delta'/\widetilde{H}(\alpha)|)$, one for each $j \in \widetilde{H}(\alpha)$. Consider any fixed $j \in \widetilde{H}(\alpha)$. Statement 3 of Theorem 12 gives that I_j ($j \in \widetilde{H}(\alpha)$) is (α, Δ)-well-structured. Since $\alpha > \frac{\tau}{4s^2}$ (also by the statement of Theorem 12), it is easy to check the condition of Lemma 8 holds where the role of "δ" in that inequality is played by $\delta'/|\widetilde{H}(\alpha)|$, so we may apply Lemma 8 and conclude that with failure probability at most $\delta'/|\widetilde{H}(\alpha)|$ (recall that by statement 4 of Theorem 12 we have $|\widetilde{H}(\alpha)| \leq s\log(8s^3/\tau)$), *SHIV* returns an assignment to the variables in I_j which sets the high-variation variable to z_j as required. By a union bound, the overall failure probability that any I_j ($j \in \widetilde{H}(\alpha)$) has its high-variation variable not set according to z is at most δ'. Now statement 5 and the discussion preceding this lemma (the isomorphism between f' and f'') give that *SimMQ* sets all of the variables that are relevant in f' correctly according to z in the assignment x it constructs in Step 2. Since this assignment x sets all variables in $\bigcup_{j\in\widetilde{L}} I_j$ to 0, the bit $b = f(x)$ that is returned is the correct value of $f''(z)$, with failure probability at most δ' as required. □

With Lemma 22 in hand, we have that with failure probability at most $1/100$, the execution of $\mathit{LearnPoly}'(s, |\widetilde{H}(\alpha)|, \epsilon/4, 1/100)$ in Step 7 of *Test-Sparse-Poly*

correctly simulates all membership queries. As a consequence, Corollary 11 thus gives that *LearnPoly*′$(s, |\widetilde{H}(\alpha)|, \epsilon/4, 1/100))$ returns "not s-sparse" with probability at most $1/100$. Summing all the failure probabilities over the entire execution of the algorithm, the overall probability that *Test-Sparse-Poly* does not output "yes" is at most

$$\overbrace{1/10}^{\text{Theorem 12}} + \overbrace{1/100}^{\text{Step 4}} + \overbrace{1/100}^{\text{Step 6}} + \overbrace{1/100}^{\text{Lemma 22}} + \overbrace{1/100}^{\text{Corollary 11}} < 1/5,$$

and the completeness theorem (Theorem 21) is proved. □

7 Soundness of the Test

In this section we prove the soundness of *Test-Sparse-Poly*:

Theorem 23 *If f is ϵ-far from any s-sparse polynomial, then Test-Sparse-Poly accepts with probability at most* $1/3$.

Proof To prove the soundness of the test, we start by assuming that the function f has progressed to Step 5, so there are subsets $I_1, \ldots, I_r$ and $\widetilde{H}(\alpha)$ satisfying $\widetilde{\mathrm{Vr}}_f(I_j) > \alpha + 2\Delta$ for all $j \in \widetilde{H}(\alpha)$. As in the proof of completeness, we have that the actual variations of all subsets should be close to the estimates, i.e. that $\mathrm{Vr}_f(I_j) > \alpha + \Delta$ for all $j \in \widetilde{H}(\alpha)$ except with probability at most $1/100$. We may then complete the proof in two parts by establishing the following:

- If f and $\widetilde{f}'$ are ϵ_a-far, step 6 will accept with probability at most δ_a.
- If $\widetilde{f}'$ is ϵ_b-far from every s-sparse polynomial, Step 7 will accept with probability at most δ_b.

Establishing these statements with $\epsilon_a = \epsilon_b = \epsilon/2$, $\delta_a = 1/12$ and $\delta_b = 1/6$ will allow us to complete the proof (and we may assume throughout the rest of the proof that $\mathrm{Vr}_f(I_j) > \alpha$ for each $j \in \widetilde{H}(\alpha)$).

The first statement follows immediately by our choice of $m = \frac{1}{\epsilon_a} \ln \frac{1}{\delta_a}$ with $\epsilon_a = \epsilon/2$ and $\delta_a = 1/12$ in Step 6. Our main task is to establish the second statement, which we do using Lemmas 24 and 25 stated below. Intuitively, we would like to show that if *LearnPoly*′ outputs a hypothesis h (which must be an s-sparse polynomial since *LearnPoly*′ is proper) with probability greater than $1/6$, then $\widetilde{f}'$ is close to a junta isomorphic to h. To do this, we establish that if *LearnPoly*′ succeeds with high probability, then the last hypothesis on which an equivalence query is performed in *LearnPoly*′ is a function which is close to $\widetilde{f}'$. Our proof uses two lemmas: Lemma 25 tells us that this holds if the high variation subsets satisfy a certain structure, and Lemma 24 tells us that if *LearnPoly*′ succeeds with high probability then the subsets indeed satisfy this structure. We now state these lemmas formally and complete the proof of the theorem, deferring the proofs of the lemmas until later.

Recall that the algorithm *LearnPoly*′ will make repeated calls to *SimMQ* which in turn makes repeated calls to *SHIV*. Lemma 24 states that if, with probability greater than δ_2, all of these calls to *SHIV* return without failure, then the subsets associated with $\widetilde{H}(\alpha)$ have a special structure.

Lemma 24 *Let $J \subset [n]$ be a subset of variables obtained by including the highest-variation element in I_j for each $j \in \widetilde{H}(\alpha)$ (breaking ties arbitrarily). Suppose that $k > 300|\widetilde{H}(\alpha)|/\epsilon_2$ queries are made to SimMQ. Suppose moreover that* Pr [*every call to SHIV that is made during these k queries returns without outputting 'fail'*] *is greater than δ_2 for $\delta_2 = 1/\Omega(k)$. Then the following both hold*:

- *every subset I_j for $j \in \widetilde{H}(\alpha)$ satisfies $\mathrm{Vr}_f(I_j \setminus J) \leq 2\epsilon_2/|\widetilde{H}(\alpha)|$; and*
- *the function $\widetilde{f}'$ is ϵ_2-close to the junta $g : \{0,1\}^{|\widetilde{H}(\alpha)|} \to \{-1,1\}$ defined as*:

$$g(x) \stackrel{\text{def}}{=} \mathrm{sign}(\mathbb{E}_{z \in \{0,1\}^{[n]\setminus J}}[\widetilde{f}'((x \cap J) \sqcup z)]).$$

Given that the subsets associated with $\widetilde{H}(\alpha)$ have this special structure, Lemma 25 tells us that the hypothesis output by *LearnPoly′* should be close to the junta g.

Lemma 25 *Define Q_E as the maximum number of calls to SimMQ that will be made by LearnPoly′ in all of its equivalence queries. Suppose that for every $j \in \widetilde{H}(\alpha)$, it holds that $\mathrm{Vr}_f(I_j \setminus J) < 2\epsilon_2/|\widetilde{H}(\alpha)|$ with $\epsilon_2 < \frac{\alpha}{800Q_E}$. Then the probability that LearnPoly′ outputs a hypothesis h which is $\epsilon/4$-far from the junta g is at most $\delta_3 = 1/100$.*

We will prove Lemmas 24 and 25 shortly, but first we argue that they suffice to prove the desired result. Suppose that *LearnPoly′* accepts with probability at least $\delta_b = 1/6$. Assume *LearnPoly′* makes at least k queries to *SimMQ* (we address this in the next paragraph); then it follows from Lemma 24 that the bins associated with $\widetilde{H}(\alpha)$ satisfy the conditions of Lemma 25 and that $\widetilde{f}'$ is ϵ_2-close to the junta g. Now applying Lemma 25, we have that with failure probability at most $1/100$, *LearnPoly′* outputs a hypothesis which is $\epsilon/4$-close to g. But then $\widetilde{f}'$ must be $(\epsilon_2 + \epsilon/4)$-close to this hypothesis, which is an s-sparse polynomial.

We need to establish that *LearnPoly′* indeed makes $k > 300|\widetilde{H}(\alpha)|/\epsilon_2$ *SimMQ* queries for an ϵ_2 that satisfies the condition on ϵ_2 in Lemma 25. (Note that if *LearnPoly′* does not actually make this many queries, we can simply have it make artificial calls to *SHIV* to achieve this. An easy extension of our completeness proof handles this slight extension of the algorithm; we omit the details.) Since we need $\epsilon_2 < \alpha/800Q_E$ and Theorem 10 gives us that $Q_E = (|\widetilde{H}(\alpha)|s + 2) \cdot \frac{4}{\epsilon} \ln 300(|\widetilde{H}(\alpha)|s + 2)$ (each equivalence query is simulated using $\frac{4}{\epsilon} \ln 300(|\widetilde{H}(\alpha)|s + 2)$ random examples), an easy computation shows that it suffices to take $k = \mathrm{poly}(s, 1/\epsilon)$, and the proof of Theorem 23 is complete. □

We now give a proof of Lemma 25, followed by a proof of Lemma 24.

Proof of Lemma 25 By assumption each $\mathrm{Vr}_f(I_j \setminus J) \leq 2\epsilon_2/|\widetilde{H}(\alpha)|$ and $\mathrm{Vr}_f(I_j) > \alpha$, so subadditivity of variation gives us that for each $j \in \widetilde{H}(\alpha)$, there exists an $i \in I_j$ such that $\mathrm{Vr}_f(i) > \alpha - 2\epsilon_2/|\widetilde{H}(\alpha)|$. Thus for every each call to *SHIV* made by *SimMQ*, the conditions of Lemma 9 are satisfied with $\mathrm{Vr}_f(i) > \alpha - 2\epsilon_2/|\widetilde{H}(\alpha)|$ and $\mathrm{Vr}_f(I_j \setminus J) < 2\epsilon_2/|\widetilde{H}(\alpha)|$. We show that as long as $\epsilon_2 < \frac{\alpha}{800Q_E}$, the probability that any particular query z to *SimMQ* has a variable set incorrectly is at most $\delta_3/3Q_E$.

Suppose *SHIV* has been called with failure probability δ_4, then the probability given by Lemma 9 is at most:

$$(\delta_4/2)^{1-2\epsilon_2/(\alpha\cdot|\widetilde{H}(\alpha)|)} \cdot \frac{2}{\alpha} \ln\left(\frac{2}{\delta_4}\right) \cdot 2\epsilon_2/|\widetilde{H}(\alpha)|. \tag{1}$$

We shall show that this is at most $\delta_3/3|\widetilde{H}(\alpha)|Q_E = 1/300Q_E|\widetilde{H}(\alpha)|$. Taking $\epsilon_2 \leq \alpha/800Q_E$ simplifies (1) to:

$$\frac{1}{300Q_E|\widetilde{H}(\alpha)|} \cdot (\delta_4/2)^{1-2\epsilon_2/(\alpha\cdot|\widetilde{H}(\alpha)|)} \cdot \frac{3}{4} \ln\frac{2}{\delta_4},$$

which is at most $1/300|\widetilde{H}(\alpha)|Q_E$ as long as

$$(2/\delta_4)^{1-2\epsilon_2/(\alpha\cdot|\widetilde{H}(\alpha)|)} > \frac{3}{4} \ln\frac{2}{\delta_4},$$

which certainly holds for our choice of ϵ_2 and the setting of $\delta_4 = 1/100k|\widetilde{H}(\alpha)|$. Each call to *SimMQ* uses $|\widetilde{H}(\alpha)|$ calls to *SHIV*, so a union bound gives that each random query to *SimMQ* returns an incorrect assignment with probability at most $1/300Q_E$.

Now, since $\widetilde{f}'$ and g are ϵ_2-close and ϵ_2 satisfies $\epsilon_2 Q_E \leq \delta_3/3$, in the uniform random samples used to simulate the final (accepting) equivalence query, *LearnPoly*′ will receive examples labeled correctly according to g with probability at least $1 - 2\delta_3/3$. Finally, note that *LearnPoly*′ makes at most $|\widetilde{H}(\alpha)|s + 2$ equivalence queries and hence each query is simulated using $\frac{4}{\epsilon} \ln \frac{3(|\widetilde{H}(\alpha)|s+2)}{\delta_3}$ random examples (for a failure probability of $\frac{\delta_3}{|\widetilde{H}(\alpha)|s+2}$ for each equivalence query). Then *LearnPoly*′ will reject with probability at least $1 - \delta_3/3$ unless g and h are $\epsilon/4$-close. This concludes the proof of Lemma 25. □

Proof of Lemma 24 We prove that if $\mathrm{Vr}_f(I_j \setminus J) > 2\epsilon_2/|\widetilde{H}(\alpha)|$ for some $j \in \widetilde{H}(\alpha)$, then the probability that all calls to *SHIV* return successfully is at most δ_2. The closeness of $\widetilde{f}'$ and g follows easily by the subadditivity of variation and Proposition 3.2 of [12].

First, we prove a much weaker statement whose analysis and conclusion will be used to prove the proposition. We show in Proposition 26 that if the test accepts with high probability, then the variation from each variable in any subset is small. We use the bound on each variable's variation to obtain the concentration result in Proposition 27, and then complete the proof of Lemma 24.

Proposition 26 *Suppose that k calls to SHIV are made with a particular subset I, and let i be the variable with the highest variation in I. If* $\mathrm{Vr}_f(j) > \epsilon_2/100|\widetilde{H}(\alpha)|$ *for some* $j \in I \setminus i$, *then the probability that SHIV returns without outputting 'fail' for all k calls is at most* $\delta^* = e^{-k/18} + 2e^{-c}$.

Proof Suppose that there exist $j, j' \in I$ with $\mathrm{Vr}_f(j) \geq \mathrm{Vr}_f(j') \geq \epsilon_2/100|\widetilde{H}(\alpha)|$. A standard Chernoff bound gives that except with probability at most $e^{-k/18}$, for

at least $(1/3)k$ of the calls to *SHIV*, variables j and j' are in different partitions. In these cases, the probability *SHIV* does not output 'fail' is at most $2(1-\epsilon_2/100|\widetilde{H}(\alpha)|)^c$, since for each of the c runs of the independence test, one of the partitions must not be marked. The probability no call outputs 'fail' is at most $e^{-k/18}+2(1-\epsilon_2/100|\widetilde{H}(\alpha)|)^{ck/3}$. Using the standard bound $(1+x)\leq e^x$, we have $(1-\epsilon_2/100|\widetilde{H}(\alpha)|)^{ck/3}\leq(1/e)^{ck\epsilon_2/300|\widetilde{H}(\alpha)|}$, and our choice of $k>300|\widetilde{H}(\alpha)|/\epsilon_2$ ensures that $(1/e)^{ck\epsilon_2/300|\widetilde{H}(\alpha)|}\leq(1/e)^c$. □

Since in our setting $|I_j|$ may depend on n, using the monotonicity of variation with the previous claim does not give a useful bound on $\mathrm{Vr}_f(I\setminus i)$. But we see from the proof that if the variation of each partition is not much less than $\mathrm{Vr}_f(I\setminus i)$ and $\mathrm{Vr}_f(I\setminus i)>2\epsilon_2/|\widetilde{H}(\alpha)|$, then with enough calls to *SHIV* one of these calls should output "fail." Hence the lemma will be easily proven once we establish the following proposition:

Proposition 27 *Suppose that k calls to SHIV are made with a particular subset I having $\mathrm{Vr}_f(I\setminus i)>2\epsilon_2/|\widetilde{H}(\alpha)|$ and $\mathrm{Vr}_f(j)\leq\epsilon_2/100|\widetilde{H}(\alpha)|$ for every $j\in I\setminus i$. Then with probability greater than $1-\delta^{**}=1-e^{-k/18}$, at least $1/3$ of the k calls to SHIV yield both $\mathrm{Vr}_f(I^1)>\eta\mathrm{Vr}_f(I\setminus i)/2$ and $\mathrm{Vr}_f(I^0)>\eta\mathrm{Vr}_f(I\setminus i)/2$, where $\eta=1/e-1/50$.*

Proof We would like to show that a random partition of I into two parts will result in parts each of which has variation not much less than the variation of $I\setminus i$. Choosing a partition is equivalent to choosing a random subset I' of $I\setminus i$ and including i in I' or $I\setminus I'$ with equal probability. Thus it suffices to show that for random $I'\subseteq I\setminus i$, it is unlikely that $\mathrm{Vr}_f(I')$ is much smaller than $\mathrm{Vr}_f(I\setminus i)$.

This does not hold for general I, but by bounding the variation of any particular variable in I, which we have done in Proposition 26, and computing the *unique-variation* (a technical tool introduced in [12]) of I', we may obtain a deviation bound on $\mathrm{Vr}_f(I')$. The following statement follows from Lemma 3.4 of [12]:

Proposition 28 [12] *Define the unique-variation of variable j (with respect to i) as*

$$\mathrm{Ur}_f(j)=\mathrm{Vr}_f([j]\setminus i)-\mathrm{Vr}_f([j-1]\setminus i).$$

Then for any $I'\subseteq I\setminus i$,

$$\mathrm{Vr}_f(I')\geq\sum_{j\in I'}\mathrm{Ur}_f(j)=\sum_{j\in I'}\mathrm{Vr}_f([j]\setminus i)-\mathrm{Vr}_f([j-1]\setminus i).$$

Now $\mathrm{Vr}_f(I')$ is lower bounded by a sum of independent, non-negative random variables whose expectation is given by

$$\mathbb{E}\left[\sum_{j\in I'}\mathrm{Ur}_f(j)\right]=\sum_{j=1}^{n}(1/2)\mathrm{Ur}_f(j)=\mathrm{Vr}_f(I\setminus i)/2\stackrel{\mathrm{def}}{=}\mu.$$

To obtain a concentration property, we require a bound on each $\mathrm{Ur}_f(j) \leq \mathrm{Vr}_f(j)$, which is precisely what we showed in the previous proposition. Note that $\mathrm{Ur}_f(i) = 0$, and recall that we have assumed that $\mu > \epsilon_2/|\widetilde{H}(\alpha)|$ and every $j \in I \setminus i$ satisfies $\mathrm{Vr}_f(j) < \mu/100$.

Now we may use the bound from [12] in Proposition 3.5 with $\eta = 1/e - 2/100$ to obtain:

$$\Pr\left[\sum_{j \in I'} \mathrm{Ur}_f(j) < \eta\mu\right] < \exp\left(\frac{100}{e}(\eta e - 1)\right) \leq 1/e^2.$$

Thus the probability that one of I^0 and I^1 has variation less than $\eta\mu$ is at most $1/2$. We expect that half of the k calls to *SHIV* will result in I^0 and I^1 having variation at least $\eta\mu$, so a Chernoff bound completes the proof of the claim with $\delta^{**} \leq e^{-k/18}$. This concludes the proof of Proposition 27. □

Finally, we proceed to prove the lemma. Suppose that there exists some I such that $\mathrm{Vr}_f(I \setminus i) > 2\epsilon_2/|\widetilde{H}(\alpha)|$. Now the probability that a particular call to *SHIV* with subset I succeeds is:

$$\Pr[\mathrm{marked}(I^0); \neg\,\mathrm{marked}(I^1)] + \Pr[\mathrm{marked}(I^1); \neg\,\mathrm{marked}(I^0)].$$

By Propositions 26 and 27, if with probability at least $\delta^* + \delta^{**}$ none of the k calls to *SHIV* return fail, then for $k/3$ runs of *SHIV* both $\mathrm{Vr}_f(I^1)$ and $\mathrm{Vr}_f(I^0)$ are at least $\eta\epsilon_2/|\widetilde{H}(\alpha)| > \epsilon_2/4|\widetilde{H}(\alpha)|$ and thus both probabilities are at most $(1 - \epsilon_2/4|\widetilde{H}(\alpha)|)^c$.

As in the analysis of the first proposition, we may conclude that every subset I which is called with *SHIV* at least k times either satisfies $\mathrm{Vr}_f(I \setminus i) < 2\epsilon_2/|\widetilde{H}(\alpha)|$ or will cause the test to reject with probability at least $1 - \delta^{**} - 2\delta^*$. Recall that $\delta^* = 2e^{-c} + e^{-k/18}$; since *SHIV* is set to run with failure probability at most $1/|\widetilde{H}(\alpha)|k$, we have that δ_2 is $1/\Omega(k)$. This concludes the proof of Lemma 24. □

8 Conclusion and Future Directions

An obvious question raised by our work is whether similar methods can be used to efficiently test s-sparse polynomials over a general finite field $\mathbb{F}$, with query and time complexity polynomial in s, $1/\epsilon$, and $|\mathbb{F}|$. The basic algorithm of [8] uses $\tilde{O}((s|\mathbb{F}|)^4/\epsilon^2)$ queries to test s-sparse polynomials over $\mathbb{F}$, but has running time $2^{\omega(s|\mathbb{F}|)} \cdot (1/\epsilon)^{\log\log(1/\epsilon)}$ (arising, as discussed in Sect. 1, from brute-force search for a consistent hypothesis.). One might hope to improve that algorithm by using techniques from the current paper. However, doing so requires an algorithm for properly learning s-sparse polynomials over general finite fields. To the best of our knowledge, the most efficient algorithm for doing this (given only black-box access to $f : \mathbb{F}^n \to \mathbb{F}$) is the algorithm of Bshouty [5] which requires $m = s^{O(|\mathbb{F}| \log |\mathbb{F}|)} \log n$ queries and runs in $\mathrm{poly}(m, n)$ time. (Other learning algorithms are known which do not have this exponential dependence on $|\mathbb{F}|$, but they either require evaluating the polynomial at complex roots of unity [19] or on inputs belonging to an extension field of $\mathbb{F}$ [14, 16].) It would be interesting to know whether there is a testing algorithm

that simultaneously achieves a polynomial runtime (and hence query complexity) dependence on both the size parameter s and the cardinality of the field $|\mathbb{F}|$.

Another goal for future work is to apply our methods to other classes beyond just polynomials. Is it possible to combine the "testing by implicit learning" approach of [8] with other membership-query-based learning algorithms, to achieve time and query efficient testers for other natural classes?

References

1. Alon, N., Kaufman, T., Krivelevich, M., Litsyn, S., Ron, D.: Testing low-degree polynomials over $GF(2)$. In: Proceedings of RANDOM-APPROX, pp. 188–199 (2003)
2. Angluin, D.: Queries and concept learning. Mach. Learn. **2**, 319–342 (1988)
3. Blum, A., Singh, M.: Learning functions of k terms. In: Proceedings of the 3rd Annual Workshop on Computational Learning Theory, COLT, pp. 144–153 (1990)
4. Blum, M., Luby, M., Rubinfeld, R.: Self-testing/correcting with applications to numerical problems. J. Comput. Syst. Sci. **47**, 549–595 (1993). Earlier version in STOC'90
5. Bshouty, N.: Simple learning algorithms using divide and conquer. Comput. Complex. **6**, 174–194 (1997)
6. Bshouty, N.: On learning multivariate polynomials under the uniform distribution. Inf. Process. Lett. **61**(3), 303–309 (1997)
7. Bshouty, N., Mansour, Y.: Simple learning algorithms for decision trees and multivariate polynomials. SIAM J. Comput. **31**(6), 1909–1925 (2002)
8. Diakonikolas, I., Lee, H., Matulef, K., Onak, K., Rubinfeld, R., Servedio, R., Wan, A.: Testing for concise representations. In: Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science, FOCS, pp. 549–558 (2007)
9. Ehrenfeucht, A., Karpinski, M.: The computational complexity of (XOR, AND)-counting problems. Technical report. Preprint (1989)
10. Fischer, E.: The art of uninformed decisions: a primer to property testing. Comput. Complex. Column Bull. Eur. Assoc. Theor. Comput. Sci. **75**, 97–126 (2001)
11. Fischer, P., Simon, H.U.: On learning ring-sum expansions. SIAM J. Comput. **21**(1), 181–192 (1992)
12. Fischer, E., Kindler, G., Ron, D., Safra, S., Samorodnitsky, A.: Testing juntas. J. Comput. Syst. Sci. **68**, 753–787 (2004)
13. Goldreich, O., Goldwaser, S., Ron, D.: Property testing and its connection to learning and approximation. J. ACM **45**, 653–750 (1998)
14. Grigoriev, D., Karpinski, M., Singer, M.: Fast parallel algorithms for sparse multivariate polynomial interpolation over finite fields. SIAM J. Comput. **19**(6), 1059–1063 (1990)
15. Kahn, J., Kalai, G., Linial, N.: The influence of variables on boolean functions. In: Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science, FOCS, pp. 68–80 (1988)
16. Karpinski, M.: Boolean circuit complexity of algebraic interpolation problems. TR-89-027 (1989)
17. Karpinski, M., Luby, M.: Approximating the number of zeros of a $GF[2]$ polynomial. J. Algorithms **14**, 280–287 (1993)
18. Luby, M., Velickovic, B., Wigderson, A.: Deterministic approximate counting of depth-2 circuits. In: Proceedings of the 2nd ISTCS, pp. 18–24 (1993)
19. Mansour, Y.: Randomized interpolation and approximation of sparse polynomials. SIAM J. Comput. **24**(2), 357–368 (1995)
20. Matulef, K., O'Donnell, R., Rubinfeld, R., Servedio, R.: Testing halfspaces. Technical Report 128, Electronic Colloquium in Computational Complexity (2007)
21. Motwani, R., Raghavan, P.: Randomized Algorithms. Cambridge University Press, New York (1995)
22. Parnas, M., Ron, D., Samorodnitsky, A.: Testing basic boolean formulae. SIAM J. Discrete Math. **16**, 20–46 (2002)
23. Ron, D.: Property testing (a tutorial). In: Rajasekaran, S., Pardalos, P.M., Reif, J.H., Rolim, J.D.P. (eds.) Handbook of Randomized Computing, vol. II. Kluwer Academic, Norwell (2001)
24. Ron, D.: Property testing: a learning theory perspective. COLT 2007 Invited Talk, slides available at http://www.eng.tau.ac.il/danar/Public-ppt/colt07.ppt (2007)

25. Roth, R., Benedek, G.: Interpolation and approximation of sparse multivariate polynomials over $GF(2)$. SIAM J. Comput. **20**(2), 291–314 (1991)
26. Rubinfeld, R.: Sublinear time algorithms. In: Proceedings of the International Congress of Mathematicians, ICM, 2006
27. Schapire, R., Sellie, L.: Learning sparse multivariate polynomials over a field with queries and counterexamples. J. Comput. Syst. Sci. **52**(2), 201–213 (1996)