Sparse Polynomial Learning and Graph Sketching

Murat Kocaoglu[1*], **Karthikeyan Shanmugam**[1†], **Alexandros G.Dimakis**[1‡], **Adam Klivans**[2⋆]
[1]Department of Electrical and Computer Engineering, [2]Department of Computer Science
The University of Texas at Austin, USA
[*]mkocaoglu@utexas.edu, [†]karthiksh@utexas.edu
[‡]dimakis@austin.utexas.edu, [⋆]klivans@cs.utexas.edu

Abstract

Let $f : \{-1, 1\}^n \to \mathbb{R}$ be a polynomial with at most s non-zero real coefficients. We give an algorithm for exactly reconstructing f given random examples from the uniform distribution on $\{-1, 1\}^n$ that runs in time polynomial in n and 2^s and succeeds if the function satisfies the *unique sign property*: there is one output value which corresponds to a unique set of values of the participating parities. This sufficient condition is satisfied when every coefficient of f is perturbed by a small random noise, or satisfied with high probability when s parity functions are chosen randomly or when all the coefficients are positive. Learning sparse polynomials over the Boolean domain in time polynomial in n and 2^s is considered notoriously hard in the worst-case. Our result shows that the problem is tractable for almost all sparse polynomials.

Then, we show an application of this result to hypergraph sketching which is the problem of learning a sparse (both in the number of hyperedges and the size of the hyperedges) hypergraph from uniformly drawn random cuts. We also provide experimental results on a real world dataset.

1 Introduction

Learning sparse polynomials over the Boolean domain is one of the fundamental problems from computational learning theory and has been studied extensively over the last twenty-five years [1–6]. In almost all cases, known algorithms for learning or interpolating sparse polynomials require query access to the unknown polynomial. An outstanding open problem is to find an algorithm for learning s-sparse polynomials with respect to the uniform distribution on $\{-1, 1\}^n$ that runs in time polynomial in n and $g(s)$ (where g is any fixed function independent of n) and requires only randomly chosen examples to succeed. In particular, such an algorithm would imply a breakthrough result for the problem of learning k-juntas (functions that depend on only $k \ll n$ input variables; it is not known how to learn $\omega(1)$-juntas in polynomial time).

We present an algorithm and a set of natural conditions such that any sparse polynomial f satisfying these conditions can be learned from random examples in time polynomial in n and 2^s. In particular, any f whose coefficients have been subjected to a small perturbation (smoothed analysis setting) satisfies these conditions (for example, if a Gaussian with arbitrarily small variance is added independently to each coefficient, f satisfies these conditions with probability 1). We state our main result here:

Theorem 1. *Let f be an s-sparse function that satisfies at least one of the following properties: a) (smoothed analysis setting)The coefficients $\{c_i\}_{i=1}^s$ are in general position or all of them are perturbed by a small random noise. b) The s parity functions are linearly independent. c) All the coefficients are positive. Then we learn f with high probability in time* $\mathrm{poly}(n, 2^s)$.

We note that smoothed-analysis, pioneered in [7], has now become a common alternative for problems that seem intractable in the worst-case.

Our algorithm also succeeds in the presence of noise:

Theorem 2. *Let $f = f_1 + f_2$ be a polynomial such that f_1 and f_2 depend on mutually disjoint set of parity functions. f_1 is s-sparse and the values of f_1 are 'well separated'. Further, $\|f_2\|_1 \leq \nu$, (i.e., f is approximately sparse). If observations are corrupted by additive noise bounded by ϵ, then there exists an algorithm which takes $\epsilon + \nu$ as an input, that gives g in time polynomial in n and 2^s such that $\|f - g\|_2 \leq O(\nu + \epsilon)$ with high probability.*

The treatment of the noisy case, i.e., the formal statement of this theorem, the corresponding algorithm, and the related proofs are relegated to the supplementary material. All these results are based on what we call as the *unique sign property:* If there is one value that f takes which uniquely specifies the signs of the parity functions involved, then the function is efficiently learnable. Note that our results cannot be used for learning juntas or other Boolean-valued sparse polynomials, since the unique sign property does not hold in these settings.

We show that this property holds for the complement of the cut function on a hypergraph (no. of hyperedges $-$ cut value). This fact can be used to learn the cut complement function and eventually infer the structure of a sparse hypergraph from random cuts. Sparsity implies that the number of hyperedges and the size of each hyperedge is of constant size. Hypergraphs can be used to represent relations in many real world data sets. For example, one can represent the relation between the books and the readers (users) on the Amazon dataset with a hypergraph. Book titles and Amazon users can be mapped to nodes and hyperedges, respectively ([8]). Then a node belongs to a hyperedge, if the corresponding book is read by the user represented by that hyperedge. When such graphs evolve over time (and space), the difference graph filtered by time and space is often sparse. To locate and learn the few hyperedges from random cuts in such difference graphs constitutes hypergraph sketching. We test our algorithms on hypergraphs generated from the dataset that contain the time stamped record of messages between Yahoo! messenger users marked with the user locations (zip codes).

1.1 Approach and Related Work

The problem of recovering the sparsest solution of a set of underdetermined linear equations has received significant recent attention in the context of compressed sensing [9–11]. In compressed sensing, one tries to recover an unknown sparse vector using few linear observations (measurements), possibly in the presence of noise.

The recent papers [12,13] are of particular relevance to us since they establish a connection between learning sparse polynomials and compressed sensing. The authors show that the problem of learning a sparse polynomial is equivalent to recovering the unknown sparse coefficient vector using linear measurements. By applying techniques from compressed sensing theory, namely Restricted Isometry Property (see [12]) and incoherence (see [13]), the authors independently established results for reconstructing sparse polynomials using convex optimization. The results have near-optimal sample complexity. However, the running time of these algorithms is exponential in the underlying dimension, n. This is because the measurement matrix of the equivalent compressed sensing problem requires one column for every possible non-zero monomial.

In this paper, we show how to solve this problem in time polynomial in n and 2^s under the assumption of *unique sign property* on the sparse polynomial. Our key contribution is a novel identification procedure that can reduce the list of potentially non-zero coefficients from the naive bound of 2^n to 2^s when the function has this property.

On the theoretical side, there has been interesting recent work of [14] that *approximately* learns sparse polynomial functions when the underlying domain is Gaussian. Their results do not seem to translate to the Boolean domain. We also note the work of [15] that gives an algorithm for learning sparse Boolean functions with respect to a *randomly* chosen product distribution on $\{-1, 1\}^n$. Their work does not apply to the uniform distribution on $\{-1, 1\}^n$.

On the practical side, we give an application of the theory to the problem of hypergraph sketching. We generalize a prior work [12] that applied the compressed sensing approach discussed before to

graph sketching on evolving social network graphs. In our algorithm, while the sample complexity requirements are higher, the time complexity is greatly reduced in comparison. We test our algorithms on a real dataset and show that the algorithm is able to scale well on sparse hypergraphs created out of Yahoo! messenger dataset by filtering through time and location stamps.

2 Definitions

Consider a real-valued function over the Boolean hypercube $f : \{-1,1\}^n \to \mathbb{R}$. Given a sequence of labeled samples of the form $\langle f(\mathbf{x}), \mathbf{x}\rangle$, where $\mathbf{x}$ is sampled from the uniform distribution U over the hypercube $\{-1,1\}^n$, we are interested in an efficient algorithm that learns the function f with high probability. Through Fourier expansion, f can be written as a linear combination of monomials:

$$f(\mathbf{x}) = \sum_{S\subseteq[n]} c_S \chi_S(\mathbf{x}),\ \forall\, \mathbf{x} \in \{-1,1\}^n \tag{1}$$

where $[n]$ is the set of integers from 1 to n, $\chi_S(\mathbf{x}) = \prod_{i\in S} x_i$ and $c_S \in \mathbb{R}$. Let $\mathbf{c}$ be the vector of coefficients c_S. A monomial $\chi_S(\mathbf{x})$ is also called a parity function. More background on Boolean functions and the Fourier expansion can be found in [16].

In this work, we restrict ourselves to *sparse polynomials* f with sparsity s in the Fourier domain, i.e., f is a linear combination of unknown parity functions $\chi_{S_1}(\mathbf{x}), \chi_{S_2}(\mathbf{x}), \ldots \chi_{S_s}(\mathbf{x})$ with s unknown real coefficients given by $\{c_{S_i}\}_{i=1}^s$ such that $c_{S_i} \neq 0,\ \forall 1 \leq i \leq s$; all other coefficients are 0. Let the subsets corresponding to the s parity functions form a family of sets $\mathcal{I} = \{S_i\}_{i=1}^s$. Finding $\mathcal{I}$ is equivalent to finding the s parity functions.

Note: In certain places, where the context makes it clear, we slightly abuse the notation such that the set S_i identifying a specific parity function is replaced by just the index i. The coefficients may be denoted simply by c_i and the parity functions by $\chi_i(\cdot)$.

Let $\mathbb{F}_2$ denote the binary field. Every parity function $\chi_i(\cdot)$ can be represented by a vector $\mathbf{p}_i \in \mathbb{F}_2^{n\times 1}$. The j-th entry $\mathbf{p}_i(j)$ in the vector $\mathbf{p}_i$ is 1, if $j \in S_i$ and is 0 otherwise.

Definition 1. *A set of s parity functions $\{\chi_i(\cdot)\}_{i=1}^s$ are said to be linearly independent if the corresponding set of vectors $\{\mathbf{p}_i\}_{i=1}^s$ are linearly independent over $\mathbb{F}_2$.*

Similarly, they are said to have rank r if the dimension of the subspace spanned by $\{\mathbf{p}_i\}_{i=1}^s$ is r.

Definition 2. *The coefficients $\{c_i\}_{i=1}^s$ are said to be in general position if for all possible set of values $b_i \in \{0,1,-1\},\ \forall\, 1 \leq i \leq s$, with at least one nonzero b_i, $\sum_{i=1}^s c_i b_i \neq 0$*

Definition 3. *The coefficients $\{c_i\}_{i=1}^s$ are said to be μ-separated if for all possible set of values $b_i \in \{0,1,-1\},\ \forall\, 1 \leq i \leq s$ with at least one nonzero b_i, $\left|\sum_{i=1}^s c_i b_i\right| > \mu$.*

Definition 4. *A sign pattern is a distinct vector of signs $\mathbf{a} = [\chi_1(\cdot), \chi_2(\cdot), \ldots \chi_s(\cdot))] \in \{-1,1\}^{1\times s}$ assumed by the set of s parity functions.*

Since this work involves switching representations between the real and the binary field, we define a function q that does the switch.

Definition 5. *$q : \{-1,1\}^{a\times b} \to \mathbb{F}_2^{a\times b}$ is a function that converts a sign matrix $\mathbf{X}$ to a matrix $\mathbf{Y}$ over $\mathbb{F}_2$ such that $Y_{ij} = q(X_{ij}) = 1 \in \mathbb{F}_2$, if $X_{ij} = -1$ and $Y_{ij} = q(X_{ij}) = 0 \in \mathbb{F}_2$, if $X_{ij} = 1$. Clearly, it has an inverse function q^{-1} such that $q^{-1}(\mathbf{Y}) = \mathbf{X}$.*

We also present some definitions to deal with the case when the polynomial f is not exactly s-sparse and observations are noisy. Let $2^{[n]}$ denote the power set of $[n]$.

Definition 6. *A polynomial $f : \{-1,1\}^n \to \mathbb{R}$ is called approximately (s,ν)-sparse if there exists $\mathcal{I} \subset 2^{[n]}$ with $|\mathcal{I}| = s$ such that $\sum_{S\in\mathcal{I}^c} |c_S| < \nu$, where $\{c_S\}$ are the Fourier coefficients as in (1).*

In other words, the sum of the absolute values of all the coefficients except the ones corresponding to $\mathcal{I}$ are rather small.

3 Problem Setting

Suppose m labeled samples $\langle f(\mathbf{x}), \mathbf{x}\rangle_{i=1}^{m}$ are drawn from the uniform distribution U on the Boolean hypercube. For any $\mathcal{B} \subseteq 2^{[n]}$, let $\mathbf{c}_{\mathcal{B}} \in \mathbb{R}^{2^n \times 1}$ be the vector of real coefficients such that $c_{\mathcal{B}}(S) = c_S,\ \forall S \in \mathcal{B}$ and $c_{\mathcal{B}}(S) = 0,\ \forall S \notin \mathcal{B}$. Let $\mathbf{A} \in \mathbb{R}^{m \times 2^n}$ be such that every row of $\mathbf{A}$ corresponds to one random input sample $\mathbf{x} \sim U$. Let $\mathbf{x}$ also denote the row index and $S \subseteq [n]$ denote the column index of $\mathbf{A}$. $\mathbf{A}(\mathbf{x}, S) = \chi_S(\mathbf{x})$. Let $A_{\mathcal{S}}$ denote the sub matrix formed by the columns corresponding to the subsets in $\mathcal{S}$. Let $\mathcal{I}$ be the set consisting of the s parity functions of interest in both the sparse and the approximately sparse cases. A *sparse representation* of an approximately (s, ν)-sparse function f is $f_{\mathcal{I}} = \mathbf{A}(\mathbf{x})\,\mathbf{c}_{\mathcal{I}}$, where $\mathbf{c}_{\mathcal{I}}$ is as defined above.

We review the compressed sensing framework used in [12] and [13]. Specifically, for the remainder of the paper, we rely on [13] as a point of reference. We review their framework and explain how we use it to obtain our results, particularly for the noisy case.

Let $\mathbf{y} \in \mathbb{R}^m$ and $\beta_{\mathcal{S}} \in \mathbb{R}^{2^n}$, such that $\beta_S = 0,\ \forall S \subseteq \mathcal{S}^c$. Note that, here $\mathcal{S}$ is a subset of the power set $2^{[n]}$. Now, consider the following convex program for noisy compressed sensing in this setting:

$$\min \|\beta_{\mathcal{S}}\|_1 \text{ subject to } \sqrt{\frac{1}{m}}\|\mathbf{A}\beta_{\mathcal{S}} - \mathbf{y}\|_2 \leq \epsilon. \tag{2}$$

Let $\beta_{\mathcal{S}}^{\text{opt}}$ be an optimum for the program (2). Note that only the columns of $\mathbf{A}$ in $\mathcal{S}$ are used in the program. The convex program runs in time $\text{poly}(m, |\mathcal{S}|)$. The incoherence property of the matrix A in [13] implies the following.

Theorem 3. *([13]) For any family of subsets $\mathcal{I} \in 2^{[n]}$ such that $|\mathcal{I}| = s,\ m = 4096ns^2$ and $c_1 = 4, c_2 = 8$, for any feasible point $\beta_{\mathcal{S}}$ of program 2, we have:*

$$\|\beta_{\mathcal{S}} - \beta_{\mathcal{S}}^{\text{opt}}\|_2 \leq c_1\epsilon + c_2\left(\frac{n}{m}\right)^{1/4}\|\beta_{\mathcal{I}^c \bigcap \mathcal{S}}\|_1 \tag{3}$$

with probability at least $1 - O\left(\frac{1}{4^n}\right)$

When $\mathcal{S}$ is set to the power set $2^{[n]}, \epsilon = 0$ and $\mathbf{y}$ is the vector of observed values for an s-sparse polynomial, the s-sparse vector $\mathbf{c}_{\mathcal{I}}$ is a feasible point to program (2). By Theorem 3, the program recovers the sparse vector $\mathbf{c}_{\mathcal{I}}$ and hence learns the function. The only caveat is that the complexity is exponential in n.

The main idea behind our algorithms for noiseless and noisy sparse function learning is to 'capture' the actual s-sparse set $\mathcal{I}$ of interest in a small set $\mathcal{S} : |\mathcal{S}| = O(2^s)$ of coefficients by a separate algorithm that runs in time $\text{poly}(n, 2^s)$. Using the restricted set of coefficients $\mathcal{S}$, we search for the sparse solution under the noisy and noiseless cases using program (2).

Lemma 1. *Given an algorithm that runs in time $\text{poly}(n, 2^s)$ and generates a set of parities $\mathcal{S}$ such that $|\mathcal{S}| = O(2^s), \mathcal{I} \subseteq \mathcal{S}$ with $|\mathcal{I}| = s$, program (2) with $\mathcal{S}$ and $m = 4096ns^2$ random samples as inputs runs in time $\text{poly}(n, 2^s)$ and learns the correct function with probability $1 - O\left(\frac{1}{4^n}\right)$.*

Unique Sign Pattern Property: The key property that lets us find a small $\mathcal{S}$ efficiently is the unique sign pattern property. Observe that an s-sparse function can produce at most 2^s different real values. If the maximum value obtained always corresponds to a *unique pattern of signs of parities*, by looking only at the random samples $\mathbf{x}$ corresponding to the subsequent $O(n)$ occurrences of this maximum value, we show that all the parity functions needed to learn f are captured in a small set of size 2^{s+1} (see Lemma 2 and its proof). The unique sign property again plays an important role, along with Theorem 3 with more technicalities added, in the noisy case, which we visit in Section 7.2 of the supplementary material.

In the next section, we provide an algorithm to generate the bounded set $\mathcal{S}$ for the noiseless case for an s-sparse function f and provide guarantees for the algorithm formally.

4 Algorithm and Guarantees: Noiseless case

Let $\mathcal{I}$ be the family of s subsets $\{S_i\}_{i=1}^{s}$ each corresponding to the s parity functions $\chi_{S_i}(\cdot)$ in an s-sparse function f. In this section, we provide an algorithm, named *LearnBool*, that finds a small

subset $\mathcal{S}$ of the power set $2^{[n]}$ that contains elements of $\mathcal{I}$ first and then uses program (2) with $\mathcal{S}$. We show that the algorithm learns f in time $\text{poly}\,(n, 2^s)$ from uniformly randomly drawn labeled samples from the Boolean hypercube with high probability under some natural conditions.

Recall that if the function is such that $f(\mathbf{x})$ attains its maximum value only if $[\chi_1(\mathbf{x}), \chi_2(\mathbf{x}) \ldots \chi_s(\mathbf{x})] = \mathbf{a}_{max} \in \{-1, 1\}^s$ for some unique sign pattern $\mathbf{a}_{\max}$, then the function is said to possess the *unique sign property*. Now we state the main technical lemma for the unique sign property.

Lemma 2. *If an s-sparse function f has the unique sign property then, in Algorithm 1, $\mathcal{S}$ is such that $\mathcal{I} \subseteq \mathcal{S},\ |\mathcal{S}| \leq 2^{s+1}$ with probability $1 - O\left(\frac{1}{n}\right)$ and runs in time $\text{poly}(n, 2^s)$.*

Proof. See the supplementary material. □

The proof of the above lemma involves showing that the random matrix $\mathbf{Y}_{\max}$ (see Algorithm 1) has rank at least $n - s$, leading to at most 2^s solutions for each equation in (4). The feasible solutions can be obtained by Gaussian elimination in the binary field.

Theorem 4. *Let f be an s-sparse function that satisfies at least one of the following properties:*

(a) The coefficients $\{c_i\}_{i=1}^s$ are in general position.
(b) The s parity functions are linearly independent.
(c) All the coefficients are positive.

Given labeled samples, Algorithm 1 learns f exactly (or $\mathbf{v}^{\text{opt}} = \mathbf{c}$) in time $\text{poly}\,(n, 2^s)$ with probability $1 - O\left(\frac{1}{n}\right)$.

Proof. See the supplementary material. □

Smoothed Analysis Setting: Perturbing c_i's with Gaussian random variables of standard deviation $\sigma > 0$ or by random variables drawn from any set of reasonable continuous distributions ensures that the perturbed function satisfies property (a) with probability 1.

Random Parity Functions: When c_i's are arbitrary and the set of s parity functions are drawn uniformly randomly from $2^{[n]}$, then property (b) holds with high probability if s is a constant.

Input: Sparsity parameter s, $m_1 = 2n2^s$ random labeled samples $\{\langle f(\mathbf{x}_i), \mathbf{x}_i\rangle\}_{i=1}^{m_1}$.
Pick samples $\{\mathbf{x}_{i_j}\}_{j=1}^{n_{\max}}$ corresponding to the maximum value of f observed in all the m samples.
Stack all $\mathbf{x}_{i_j}$ row wise into a matrix $\mathbf{X}_{\max}$ of dimensions $n_{\max} \times n$.
Initialise $\mathcal{S} = \emptyset$. Let $\mathbf{Y}_{\max} = q(\mathbf{X}_{\max})$.
Find all feasible solutions $\mathbf{p} \in \mathbb{F}_2^{n\times 1}$ such that:

$$\mathbf{1}_{n_{\max}\times 1} = \mathbf{Y}_{\max}\mathbf{p} \text{ or } \mathbf{0}_{n_{\max}\times 1} = \mathbf{Y}_{\max}\mathbf{p} \tag{4}$$

Collect all feasible solutions $\mathbf{p}$ to either of the above equations in the set $P \subseteq \mathbb{F}_2^{n\times 1}$.
$\mathcal{S} = \{\{j \in [n] : \mathbf{p}(j) = 1\} | \mathbf{p} \in P\}$.
Using $m = 4096ns^2$ more samples (number of rows of $\mathbf{A}$ is m corresponding to these new samples), solve:

$$\beta_{\mathcal{S}}^{\text{opt}} = \min \|\beta_{\mathcal{S}}\|_1 \text{ such that } \mathbf{A}\beta_{\mathcal{S}} = \mathbf{y}, \tag{5}$$

where $\mathbf{y}$ is the vector of m observed values.
Set $\mathbf{v}^{\text{opt}} = \beta_{\mathcal{S}}^{opt}$.
Output: $\mathbf{v}^{\text{opt}}$.

Algorithm 1: LearnBool

5 A Sparse Polynomial Learning Application: Hypergraph Sketching

Hypergraphs can be used to model the relations in real world data sets (e.g., books read by users in Amazon). We show that the cut functions on hypergraphs satisfy the unique sign property. Learning a cut function of a sparse hypergraph from random cuts is a special case of learning a sparse

polynomial from samples drawn uniformly from the Boolean hypercube. To track the evolution of large hypergraphs over a small time interval, it is enough to learn the cut function of the difference graph which is often sparse. This is called the *graph sketching problem*. Previously, graph sketching was applied to social network evolution [12]. We generalize this to hypergraphs showing that they satisfy the unique sign property, which enable faster algorithms, and provide experimental results on real data sets.

5.1 Graph Sketching

A *hypergraph* $G = (V, E)$ is a set of vertices V along with a set E of subsets of V called the *hyperedges*. *The size of a hyperedge* is the number of variables that the hyperedge connects. Let d be the maximum hyperedge size of graph G. Let $|V| = n$ and $|E| = s$.

A random cut $S \subseteq V$ is a set of vertices selected uniformly at random. Define the value of the cut S to be $c(S) = |\{e \in E : e \bigcap S \neq \emptyset,\ e \bigcap V - S \neq \emptyset\}|$. Graph sketching is the problem of identifying the graph structure from random queries that evaluate the value of a random cut, where $s \ll n$ (sparse setting). Hypergraphs naturally specify relations among a set of objects through hyperedges. For example, Amazon users can form the set E and Amazon books can form the set V. Each user may read a subset of books which represents the hyperedge. Learning the hypergraph corresponds to identifying the sets of books bought by each user. For more examples of hypergraphs in real data sets, we refer the reader to [8]. Such hypergraphs evolve over time. The difference graph between two consecutive time instants is expected to be sparse (number of edges s and maximum hyperedge size d are small). We are interested in learning such hypergraphs from random cut queries.

For simplicity and convenience, we consider *the cut complement* query, i.e., c$-$cut, which returns $s - c(S)$. One can easily represent the c$-$cut query with a sparse polynomial as follows: Let node i correspond to variable $x_i \in \{-1, +1\}$. A random cut involves choosing x_i uniformly randomly from $\{-1, +1\}$. The variables assigned to $+1$ belong to the random cut S. The value is given by the polynomial

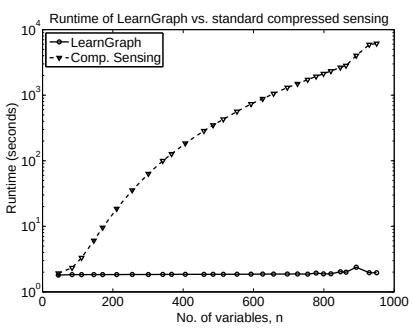
$$f_{c-cut}(\mathbf{x}) = \sum_{\mathcal{I} \in E} \left(\prod_{i \in \mathcal{I}} \frac{(1 + x_i)}{2} + \prod_{i \in \mathcal{I}} \frac{(1 - x_i)}{2} \right) = \sum_{\mathcal{I} \in E} \frac{1}{2^{|\mathcal{I}|-1}} \left(\sum_{\substack{\mathcal{J} \subseteq \mathcal{I}, \\ |\mathcal{J}| \text{is even}}} (1 + \prod_{i \in \mathcal{J}} x_i) \right). \quad (6)$$

Hence, the c$-$cut function is a sparse polynomial where the sparsity is at most $s2^{d-1}$. The variables corresponding to the nodes that belong to some hyperedge appear in the polynomial. We call these *the relevant variables* and the number of relevant variables is denoted by k. Note that, in our sparse setting $k \leq sd$. We note that for a hypergraph with no singleton hyperedge, given the c$-$cut function, it is easy to recover the hyper edges from (6). Therefore, we focus on learning the c$-$cut function to sketch the hypergraph.

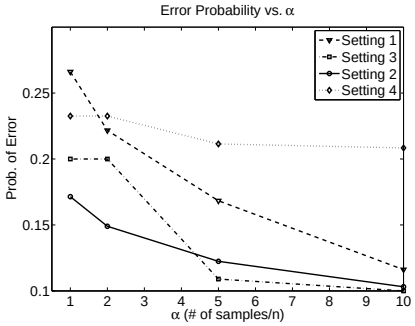
When G is a graph with edges (of cardinality 2), the compressed sensing approach (using program 2) using the cut (or c$-$cut) values as measurements is shown to be very efficient in [12] in terms of the sample complexity, i.e., the required number of queries. The run time is efficient because total number of candidate parities is $O(n^2)$. However when we consider hypergraphs, i.e., when d is a large constant, the compressed sensing approach cannot scale computationally ($\text{poly}(n^d)$ runtime). Here, based on the theory developed, we give a faster algorithm based on the unique sign property with sample complexity $m_1 = O(2^k d \log n + 2^{2d+1} s^2 (\log n + k))$ and run time of $O(m_1 2^k, n^2 \log n))$.

We observe that the c$-$cut polynomial satisfies the unique sign property. From (6), it is evident that the polynomial has only positive coefficients. Therefore, by Theorem 4, algorithm LearnBool succeeds. The maximum value of the c$-$cut function is the number of edges. Notice that the maximum value is definitely observed in two configurations of the relevant variables: If either all relevant variables are $+1$ or all are -1. Therefore, the maximum value is observed in every $2^{k-1} \leq 2^{sd}$ samples. Thus, a direct application of *LearnBool* yields $\text{poly}(n, 2^{k-1})$ time complexity, which improves the $O(n^d)$ bound for small s and d.

Improving further, we provide a more efficient algorithm tailored for the hypergraph sketching problem, which makes use of the unique sign property and some other properties of the cut function. Algorithm LearnGraph (Algorithm 4) is provided in the supplementary material.



(a) Runtime vs. # of variables, $d = 3$ and $s = 1$.



(b) Probability of error vs. α.

Figure 1: Performance figures comparing LearnGraph and Compressed Sensing approach.

Theorem 5. *Algorithm 4 exactly learns the c−cut function with probability* $1 - O(\frac{1}{n})$*with sample complexity* $m_1 = O(2^k d \log n + 2^{2d+1} s^2 (\log n + k))$ *and time complexity* $O(2^k m_1 + n^2 d \log n))$.

Proof. See the supplementary material. □

5.2 Yahoo! Messenger User Communication Pattern Dataset

We performed simulations using MATLAB on an Intel(R) Xeon(R) quad-core 3.6 GHz machine with 16 GB RAM and 10M cache. We run our algorithm on the Yahoo! Messenger User Communication Pattern Dataset [17]. This dataset contains the timestamped user communication data, i.e., information about a large number of messages sent over Yahoo! Messenger, for a duration of 28 days.

Dataset: Each row represents a message. The first two columns show the day and time (time stamp) of the message respectively. The third and fifth columns show the ID of the transmitting and receiving users, respectively. The fourth column shows the zipcode (spatial stamp) from which this particular message is transmitted. The sixth column shows if the transmitter was in the contact list of the reciver user (y) or not (n). If a transmitter sends the same receiver more than one message from the same zipcode, only the first message is shown in the dataset. In total, there are 100000 unique users and 5649 unique zipcodes.

We form a hypergraph from the dataset as follows: The transmitting users form the hyperedges and the receiving users form the nodes of the hypergraph. A hyperedge connects a set T of users if there is a transmitting user that sends a message to all the users in T. In any given time interval δt (short time interval) and small set of locations δx specified by the number of zip codes, there are few users who transmit (s) and they transmit to very few users (d). The complete set of nodes in the hypergraph (n) is taken to be those receiving users who are active during m consecutive intervals of length δt and in a set of δx zipcodes. This gives rise to a sparse graph. We identify the active set of transmitting users (hyperedges) and their corresponding receivers (nodes in these hyperedges) during a short time interval δt and a randomly selected space interval (δx, i.e., zip codes) from a large pool of receivers (nodes) that are observed during m intervals of length δt. Details of δt, m and δx chosen for experiments are given in Table 1. We note that n is in the order of 1000 usually.

Remark: Our task is to learn the c−cut function from the random queries, i.e., random +/-1 assignment of variables and corresponding c−cut values. The generated sparse graph contains only hyperedges that have more than 1 node. Other hyperedges (transmitting users) with just one node in the sparse hypergraph are not taken into account. This is because a singleton hyperedge i is always counted in the c−cut function thereby effectively its presence is masked. First, we identify the relevant variables that participate in the sparse graph. After identifying this set of candidates, correlating the corresponding candidate parities with the function output yields the Fourier coefficient of that parity (see Algorithm 4).

Table 1: Runtime for different graphs. **LG:** LearnGraph, **CS:** Compressed sensing based alg.

(a) Runtime for $d = 4$ and $s = 1$ graph.

Alg. \ n	88	159	288	556	1221
LG	1.96	2.13	2.23	2.79	4.94
CS	265.63	-	-	-	-

(b) Runtime for $d = 4$ and $s = 3$ graph.

Alg. \ n	52	104	246	412	1399
LG	1.91	2.08	2.08	2.30	4.98
CS	39.89	> 10823	-	-	-

(c) Simulation parameters for Fig. 1b

Setting No.	Interval	# of Int.	n	$\max(d)$	$\max(s)$	Zip. Set Size
Setting 1	5 min.	20	6822	10	19	20
Setting 2	20 sec.	200	5730	22	4	200
Setting 3	10 min.	10	6822	11	13	2
Setting 4	2 min.	50	6822	30	21	50

5.2.1 Performance Comparison with Compressed Sensing Approach

First, we compare the runtime of our implementation $\mathrm{LearnGraph}$ with the compressed sensing based algorithm from [12]. Both algorithms correctly identify the relevant variables in all the considered range of parameters. The last step of finding the corresponding Fourier coefficients is omitted and can be easily implemented (Algorithm 4) without significantly affecting the running time. As can be seen in Tables 1a, 1b and Fig. 1a, LearnGraph scales well to graphs on thousands of nodes. On the contrary, the compressed sensing approach must handle a measurement matrix of size $O(n^d)$, which becomes prohibitively large on graphs involving more than a few hundred nodes.

5.2.2 Error Performance of LearnGraph

Error probability (probability that the correct c$-$cut function is not recovered) versus the number of samples used is plotted for four different experimental settings of δt, δx and m in Fig. 1b. For each time interval, the error probability is calculated by averaging the number of errors among 100 different trials. For each value of α (number of samples), the error probability is averaged over time intervals to illustrate the error performance. We only keep the intervals for which the graph filtered with the considered zipcodes contains at least one user with more than one neighbor. We find that for the first 3 settings, the error probability decreases with more samples. For the fourth setting, d and s are very large and hence a large number of samples are required. For that reason, the error probability does not improve significantly. The probability of error can be reduced by repeating the experiment multiple times and taking a majority, at the cost of significantly more samples. Our plot shows only the probability of error without such a majority amplification.

6 Conclusions

We presented a novel algorithm for learning sparse polynomials by random samples on the Boolean hypercube. While the general problem of learning all sparse polynomials is notoriously hard, we show that almost all sparse polynomials can be efficiently learned using our algorithm. This is because our unique sign property holds for randomly perturbed coefficients, in addition to several other natural settings. As an application, we show that graph and hypergraph sketching lead to sparse polynomial learning problems that always satisfy the unique sign property. This allows us to obtain efficient reconstruction algorithms that outperform the previous state of the art for these problems.

An important open problem is to achieve the sample complexity of [12] while keeping the computational complexity polynomial in n.

Acknowledgments

M.K, K.S. and A.D. acknowledge the support of NSF via CCF 1422549, 1344364, 1344179 and DARPA STTR and a ARO YIP award.

7 Appendices

7.1 Proof of Theorem 4

We prove Theorem 4 at the end of this section. Next, we provide the proof for Lemma 2 about Algorithm 1 that will be used in the proof. Since the function f is s-sparse, it takes at most 2^s distinct real values.

Proof of Lemma 2

Let E_1 be the event that the maximum value observed among m_1 samples in the algorithm 1 is the maximum value attained by f. Note that, the probability that the function attains the maximum value is at least $\frac{1}{2^s}$. To see this, if the parity functions have rank r, then the set of r linearly independent parity functions take values uniformly in the hypercube $\{-1, 1\}^r$ and other are determined by these r signs. Hence, the probability of finding the maximum value is $\frac{1}{2^r} \geq \frac{1}{2^s}$. If the functions satisfies the unique sign property for the maximum value and if E_1 is true, it is easily seen that the actual party functions $\mathbf{p}_i$ are in the set P in the algorithm 1.

Consider the algorithm 1. Let E_3 be the event that the matrix $\mathbf{Y}_{\max}$ has at least rank $n - s$. E_3 implies that $|P| = |\mathcal{S}| \leq 2^{s+1}$, $\forall 1 \leq i \leq s$. Let E_2 be the event that $n_{\max} > 2n$. Conditioned on E_2 and E_1 being true, we first argue that the rank of $\mathbf{Y}_{\max}$ is at least $n - s$ with high probability. Let the rank of the actual set of parity functions $[\mathbf{p}_1, \mathbf{p}_2 \dots \mathbf{p}_s]$ be $k \leq s$.

If E_1 and E_2 are true, then $\mathbf{Y}_{\max}$ contains $2n$ random samples such that they all produce the same sign pattern $\mathbf{a}_{\max}$ because the actual function f satisfies the unique sign pattern property for the maximum value. Let $\mathbf{z}_{\max} = q(\mathbf{a}_{\max})$. Observe that rows of $\mathbf{Y}_{\max}$ are random samples uniformly drawn from the hyperplane $\mathcal{H} = \{\mathbf{x} \in \mathbb{F}_2^{n\times 1} : \mathbf{x}^T [\mathbf{p}_i] = z_{\max}(i),\ \forall 1 \leq i \leq s\}$. Since the rank of the parity functions is k, the dimension of $\mathcal{H}$ is $n - k$. Now, the rank of space spanned by $2n$ samples drawn randomly uniformly from $\mathcal{H}$ is at least the rank of space spanned by $2n$ samples drawn randomly uniformly from $\mathbb{F}_2^{1\times n-k}$. The probability that a random $2n \times n - k$ binary matrix is full rank is given by:

$$\begin{aligned}\Pr\left(\text{a random } 2n \times n - k \text{ binary matrix is full rank}\right) &= \prod_{i=0}^{n-k-1} \left(1 - \frac{1}{2^{2n-i}}\right) \geq \left(1 - \frac{1}{2^n}\right)^{n-k} \\ &\geq \left(1 - \frac{1}{2^n}\right)^{n} \geq 1 - O\left(\frac{1}{n}\right) \end{aligned} \tag{7}$$

Hence, $\Pr\left(E_3 | E_2 \bigcap E_1\right) \geq 1 - O\left(\frac{1}{n}\right)$. $\Pr\left(E_1 \bigcap E_2\right)$ is the probability that there are at least $n_{\max}$ samples corresponding to the maximum value of the actual function in the $2n2^s$ samples drawn. Therefore, $\Pr\left(E_1 \bigcap E_2\right) \geq 1 - \left(1 - \frac{1}{2^s}\right)^{2n2^s - 2n}$ because the maximum value of f is seen with probability at least $\frac{1}{2^s}$. Using this in the following chain, we have:

$$\begin{aligned}\Pr\left(|\mathcal{S}| \leq 2^{s+1},\, \mathcal{I} \subseteq \mathcal{S}\right) &\geq \Pr\left(E_1 \bigcap E_2 \bigcap E_3\right) \geq \Pr\left(E_1 \bigcap E_2\right) \Pr\left(E_3 | E_2 \bigcap E_1\right) \\ &\geq \Pr\left(E_1 \bigcap E_2\right) \left(1 - O\left(\frac{1}{n}\right)\right) \ \text{(by (7))} \\ &\geq \left(1 - \left(1 - \frac{1}{2^s}\right)^{2n2^s - 2n}\right) \left(1 - O\left(\frac{1}{n}\right)\right) \\ &\geq \left(1 - \exp\left(-2n\left(1 - \frac{1}{2^s}\right)\right)\right) \left(1 - O\left(\frac{1}{n}\right)\right) \\ &\geq \left(1 - O\left(\frac{1}{n}\right)\right) \left(1 - O\left(\frac{1}{n}\right)\right) \geq 1 - O\left(\frac{1}{n}\right). \end{aligned} \tag{8}$$

Now, we relate the unique sign property to the conditions mentioned in Theorem 4 for its proof.

Proof of Theorem 4

Due to Lemmas 2 and 1, we just need to show that each of the conditions in the theorem implies the unique sign property, i.e., the maximum value of the function f is attained when the set of parity functions takes a unique sign pattern.

Case 1: If the coefficients are in general position (Definition 2), all values taken by the function correspond to distinct sign patterns. This implies the unique sign property for the maximum value.

Case 2: If all the parity functions are linearly independent, any sign pattern can be realized. Then, the sign pattern $[\text{sign}\,(c_1)\,, \text{sign}\,(c_2) \ldots \text{sign}\,(c_s)]$ can be realized by the set of parity functions and this produces the value $\sum_{i=1}^{s} |c_i|$. And any other sign pattern will produce a strictly lesser value as all c_i are nonzero. Hence, the maximum value is unique in this case.

Case 3: Let us consider the case when all the coefficients are positive. Even if the parity functions are linearly dependent, the sign pattern with all $+1$'s can be produced and this attains the unique maximum value $\sum_{i=1}^{s} |c_i|$. This implies the unique sign property.

7.2 Algorithms and Guarantees: Noisy Case

In this section, we provide our algorithm for learning an approximately (s, ν)-sparse function with noisy samples, and prove guarantees regarding the error between the function learnt and the actual function. When m random samples are observed, the noisy output model for an approximately (s, ν)-sparse function f is given by:

$$\mathbf{y} = \mathbf{A}\mathbf{c} + \varepsilon \tag{9}$$

where $\mathbf{A}$ is the m by 2^n matrix where each row corresponds to a sample $\mathbf{x}$ and each column corresponds to a parity function and $\mathbf{c}$ is the set of Fourier coefficients for f and the noise $|\varepsilon_i| \leq \epsilon,\ 1 \leq i \leq m$. We recall that $\sum_{S \subseteq \mathcal{I}^c} |c_S| < \nu$ for an approximately sparse f. We assume that $\epsilon + \nu$ is known.

Input: The sequence of labeled samples $\langle f(\mathbf{x}_i) + \varepsilon_i, \mathbf{x}_i \rangle_{i=1}^{m}$
Initialise $\mathbf{X}_{max} = \emptyset$.
Let η be the maximum value observed.
Stack all the inputs $\mathbf{x}_i$, such that $f\,(\mathbf{x}_i) + \varepsilon_i$ is in the neighborhood of radius $2\,(\epsilon + \nu)$ around η, into $\mathbf{X}_{max}$.
Output: $\mathbf{X}_{max}$.

Algorithm 2: MaxCluster

Input: Sparsity parameter s, $\epsilon + \nu$, $m_1 = 2n2^s$ random labeled samples $\{\langle f\,(\mathbf{x}_i)\,, \mathbf{x}_i \rangle\}_{i=1}^{m_1}$.
Run MaxCluster algorithm to obtain $\mathbf{X}_{\max}$.
Initialise $P = \emptyset$.
Find all feasible solutions $\mathbf{p}$ such that: $\mathbf{1} = q(\mathbf{X}_{\max})\mathbf{p}$ or $\mathbf{0} = q(\mathbf{X}_{\max})\mathbf{p}$.
Collect all feasible $\mathbf{p}$ in the set $P \subseteq \mathbb{F}_2^n$.
$\mathcal{S} = \{\{j \in [n] : \mathbf{p}_i(j) = 1\} | \mathbf{p}_i \in P\}$.
Using $m = 4096ns^2$ more samples (number of rows of $\mathbf{A}$ is m corresponding to these new samples), solve:

$$\beta_{\mathcal{S}}^{\text{opt}} = \min \|\beta_{\mathcal{S}}\|_1 \text{ such that } \sqrt{\frac{1}{m}} \|\mathbf{A}\beta_{\mathcal{S}} - \mathbf{y}\|_2 \leq \epsilon + \nu, \tag{10}$$

where $\mathbf{y}$ is the vector of m noisy observed values (as in (5))
Set $\mathbf{v}^{\text{opt}} = \beta_{\mathcal{S}}^{opt}$.
Output: $\mathbf{v}^{\text{opt}}$.

Algorithm 3: LearnBoolNoisy

Now we state our main thoerem for learning a sparse function from noisy observations.

Theorem 6. *Assume f is an approximately (s, ν)-sparse function as given in Definition 6 and observed samples satisfy the noise model in (9). Then, Algorithm 3 outputs $\mathbf{v}^{\text{opt}}$ in time $\text{poly}(n, 2^s)$ with probability $1 - O\left(\frac{1}{n}\right)$ satisfying $\|\mathbf{c} - \mathbf{v}^{\text{opt}}\|_2 \leq \alpha_1\epsilon + \alpha_2\nu$, if f satisfies at least one of the following properties:*

(a) The coefficients $\{c_S\}_{S\in\mathcal{I}}$ are $4(\nu + \epsilon)$-separated.
(b) The set of parity functions $\chi_i(\cdot)$ are linearly independent, and $\min_{S\in\mathcal{I}} c_S > 4(\epsilon + \nu)$.
(c) All the coefficients are positive, and $\min_{S\in\mathcal{I}} c_S > 4(\epsilon + \nu)$.

Here, α_1 and α_2 are some constants.

7.3 Proof of Theorem 6

Although the observations are noisy as in the noise model given by (9), the set of inputs for which the sparse representation of the function f, i.e., $f_{\mathcal{I}}$ (this depends on only Fourier coefficients in $\mathcal{I}$) attains its maximum, can still be perfectly identified under certain conditions given in the Lemma below. Algorithm 2 identifies those inputs.

Lemma 3. *If the function f is approximately (s, ν)-sparse, observations follow the noise model in (9), and if the values of $f_{\mathcal{I}}$ are separated by at least $4(\epsilon + \nu)$, then the output matrix $\mathbf{X}_{\max}$ in Algorithm 2 will contain exactly those inputs for which $f_{\mathcal{I}}$ attains the maximum value among the drawn samples.*

Proof. Consider a sample $\mathbf{x}$. Clearly, from the noise model and the definition of approximate sparsity, $|f(\mathbf{x}_i + \varepsilon_i) - f_{\mathcal{I}}(\mathbf{x}_i)| \leq \nu + \epsilon$. Hence, when using a radius of $2(\nu + \epsilon)$ for clustering, clearly no two samples with different $f_{\mathcal{I}}$ will be included in $\mathbf{X}_{max}$ and definitely one sample belonging to the maximum $f_{\mathcal{I}}$ among the observed samples will be included. □

Proof of Theorem 6:

The three properties in the statement of Theorem 6 imply that $f_{\mathcal{I}}$ has the unique sign property for the maximum value due to the same arguments in the proof of Theorem 4. Further, they also imply that the values of $f_{\mathcal{I}}$ are separated by $4(\epsilon + \nu)$ in each of the cases. By Lemma 3, rows of $\mathbf{X}_{\max}$ contain only the inputs at which $f_{\mathcal{I}}$ attains its maximum among the observed values.

Using Lemma 2 on $f_{\mathcal{I}}$, which is exactly s-sparse, it can be seen that $|\mathcal{S}| \leq 2^{s+1}$ and contains all the parity functions in $f_{\mathcal{I}}$ with probability $1 - O\left(\frac{1}{n}\right)$ as in Algorithm 1. This is because P is formed using inputs in $\mathbf{X}_{\max}$ that give the maximum $f_{\mathcal{I}}$ among the observed samples in an identical fashion as in Algorithm 1. Now, we have the following chain of inequalities:

$$\begin{aligned}
\|\mathbf{c} - \mathbf{v}^{\text{opt}}\|_2 &\leq \|\mathbf{c}_{\mathcal{S}} - \beta_{\mathcal{S}}^{\text{opt}}\|_2 + \|\mathbf{c}_{\mathcal{S}^c}\|_2 \qquad \text{(triangle inequality)} \\
&\leq \|\mathbf{c}_{\mathcal{S}} - \beta_{\mathcal{S}}^{\text{opt}}\|_2 + \|\mathbf{c}_{\mathcal{S}^c}\|_1 \qquad (\|\cdot\|_2 \leq \|\cdot\|_1) \\
&\overset{a}{\leq} c_1(\nu + \epsilon) + c_2\left(\frac{n}{m}\right)^{1/4} \|\mathbf{c}_{\mathcal{I}^c \bigcap \mathcal{S}}\|_1 + \|\mathbf{c}_{\mathcal{S}^c}\|_1 \\
&\leq c_1(\nu + \epsilon) + c_2(\nu) + \nu \qquad (n < m) \qquad (11)
\end{aligned}$$

For inequality (a), it is easy to see that $\mathbf{c}_{\mathcal{S}}$ is a feasible solution to program 10 and therefore Theorem 3 can be applied with $\beta_{\mathcal{S}} = \mathbf{c}_{\mathcal{S}}$ with noise threshold $\nu + \epsilon$. Further, $\|\mathbf{c}_{\mathcal{I}^c}\|_1 < \nu$.

Since $|\mathcal{S}| \leq 2^{s+1}$, the optimization program 10 runs in time poly $(n, 2^s)$.

7.4 Algorithm LearnGraph

We provide the algorithm LearnGraph below. Let k be the number of relevant variables, i.e. variables that are part of at least one hyperedge. Note that $k \leq sd$.

Note: In the above algorithm, $\text{round}(.)$ function rounds a real number to the nearest integer.

Input: Number of edges s, $m_1 = \max c2^k d\log n, c2^{2d+1}s^2(\log n + k)$ random labeled samples $\{\langle f_{c-cut}(\mathbf{x}_i), \mathbf{x}_i\rangle\}_{i=1}^{m_1}$.
Pick samples $\{\mathbf{x}_{i_j}\}_{j=1}^{n_{\max}}$ corresponding to the maximum value of f_{c-cut} observed in all the m samples. Stack all $\mathbf{x}_{i_j}$ row wise into a matrix $\mathbf{X}_{\max}$ of dimensions $n_{\max} \times n$.
$\mathbf{R} \Leftarrow \mathbf{X}_{\max}^T \mathbf{X}_{\max}$.
Estimate d by $d = \max_i |\{j : \mathbf{R}(i,j) = \max(\mathbf{R}(i,:))\}|$
$c_0 = \frac{\sum_{i=1}^{m_1} f_{c-cut}(\mathbf{x}_i)}{m_1}$
Identify the constant Fourier coefficient by rounding c_0 to the nearest integer multiple of $\frac{1}{2^d}$,
$c_0 \Leftarrow \frac{\text{round}(c_0 2^d)}{2^d}$.
$f_{c-cut} \Leftarrow f_{c-cut} - c_0$.
Initialize $\mathcal{S}_i = \emptyset\ \forall i \in \{1, 2, ..., n\}$
Stack x_j for all (i,j) s.t. $\mathbf{R}(i,j) = n_{\max}$, into $\mathcal{S}_i$.
For all $M_{k_i} \subseteq \mathcal{S}_i$ s.t. $|M_{k_i}| \leq d$, $|M_{k_i}|$ is even, calculate $c_{\chi_{M_{k_i}}} = \frac{\sum_{i=1}^{m_1} f_{c-cut}(\mathbf{x}_i)\chi_{M_{k_i}}(\mathbf{x}_i)}{m_1}$.
Find Fourier coefficients of parities by rounding $\chi_{\mathcal{M}}$ to the nearest integer multiple of $\frac{1}{2^d}$,
$c_{\chi_{\mathcal{M}}} \Leftarrow \frac{\text{round}(c_{\chi_{\mathcal{M}}} 2^d)}{2^d}$
Stack all non-zero parity coefficients and parity variables into $\mathbf{c}$ and $\mathcal{M}$, respectively.
Output: $\mathbf{c}, \mathcal{M}$.

Algorithm 4: LearnGraph

Lemma 4. *(Chernoff's bound) [18] Let $X_i,\ 1 \leq i \leq n$ be i.i.d random variables taking values in $[b,c]$. Let $X = \sum_{i=1}^{n} X_i$. Let $\mathbb{E}[X] = \mu$. Then,* $\Pr\left(\sum_i X_i \geq \mu + a\right) \leq \exp\left(-\frac{a^2}{2(b-c)^2 n}\right)$ *and* $\Pr\left(\sum_i X_i \leq \mu - a\right) \leq \exp\left(-\frac{a^2}{2(b-c)^2 n}\right)$.

Proof of Theorem 5:

Without loss of generality, let us consider the case when a hyperedge involves more than two vertices. Let us consider a variable to be relevant only if it is involved in at least one hyperedge with more than one vertex. Note that the number of relevant variables is $k \leq sd$. The c$-$cut function counts a hyperedge if either all its nodes are assigned $+1$ or when all its nodes are assigned -1. When the c$-$cut function attains its maximum values, every hyperedge is counted. Clearly, when all the relevant variables are assigned the same value from $\{+1, -1\}$, then c$-$cut attains its maximum value. This happens with probability $1/2^{k-1}$. Let n_1 denote the number of samples where all relevant variables are assigned the same sign. Therefore, out of m_1 samples taken,

$$\begin{aligned}\Pr(n_1 \geq cd\log n) &\geq 1 - \left(1 - \frac{1}{2^{k-1}}\right)^{m_1 - cd\log n} \\ &\geq 1 - \exp\left(-2\left(1 - \frac{1}{2^k}\right)cd\log n\right) \\ &\geq 1 - O\left(\frac{1}{n^{cd}}\right) \end{aligned} \tag{12}$$

Therefore, $n_{\max} \geq cd\log n$ with very high probability. Let E_1 denote the event $n_{\max} \geq cd\log n$. Suppose E_1 is true, then any two variables that belong to the same hyperedge will have identical columns in $\mathbf{X}_{max}$. Therefore, if i and j are in the same hyperedge , then $R(i,j) = n_{\max}$. Let $\hat{\mathbf{x}}_i$ be the i-th column consisting of signs of the i-th variable. Let x_{ik} be the k-th entry of the ith column. Then, $R(i,j) = \hat{\mathbf{x}}_i^T \hat{\mathbf{x}}_j = \sum_{k=1}^{n_{\max}} x_{ik}x_{jk}$. $Y_k \triangleq x_{ik}x_{jk} \in \{-1, 1\}$ are identically distributed independent random variables for $i \neq j$. Let E_2 denote the event that $R_{i,j} \leq \frac{cd\log n}{1+\epsilon},\ \forall j \neq i,\ \forall i$ which is irrelevant. Observe that for an irrelevant variable i, $\mathbb{E}[R(i,j)] = 0$ for any $j \neq i$.

Thus, applying Lemma 4 with $a = \frac{n_{\max}}{(1+\epsilon)}$ for some constant $\epsilon > 0$ and $b = -1$ and $c = 1$, we have:

$$\begin{aligned}\Pr\left(E_2|E_1\right) &= 1 - \Pr\left(\exists \text{ irrelevant variable } i,\ j \neq i : R(i,j) > a|E_1\right) \\ &\geq 1 - n^2 \exp\left(-\frac{n_{\max}}{8(1+\epsilon)^2}\right) \text{ (union bound)} \\ &\geq 1 - O\left(\frac{1}{n}\right) \text{ (for a large enough constant } c) \end{aligned} \tag{13}$$

Therefore, when both E_1 and E_2 are true, then for all irrelevant variable i, $\mathcal{S}_i = i$. If i is relevant, then $\mathcal{S}_i$ also contains variable(s) other than i. Now, for every i, $\mathcal{S}_i$ represents variables which participate in some hyperedge along with i if i is relevant, since for all such variable j, $R(i,j) = n_{max}$. Then, if d is known, we take all possible d subsets M_{k_i} of $\mathcal{S}_i$ and correlate the corresponding parity function M_{k_i} with the function values to find the coefficient. Since $m_1 = c2^k d \log n$ samples are available, error can be made less than $1/2^d$, and this gives an exact estimate with high probability when the result is rounded off to the nearest multiple of $1/2^d$. Let E_3 be the event that $\forall M_{k_i} \subset \mathcal{S}_i,\ \forall \text{ relevant } i : |\sum_i f\left(\mathbf{x}_i\right) \chi_{M_{k_i}}\left(\mathbf{x}_i\right) - m_1 \mathbb{E}[f\left(\mathbf{x}\right) \chi_{M_{k_i}}\left(\mathbf{x}\right)]| \leq m_1 \frac{1}{2^d}$. Since, the function takes values between 0 and s (the number of hyperedges), taking $a = m_1 \frac{1}{2^d}$, $b = -s$ and $c = s$, and applying Lemma 4, we have:

$$\Pr\left(E_3|E_1, E_2\right) \geq 1 - 2^k \exp\left(-\frac{m_1}{2^{2d+1}s^2}\right) \geq 1 - O\left(\frac{1}{n}\right).$$

Therefore, $\Pr\left(E_1 \bigcap E_2 \bigcap E_3\right) \geq 1 - O\left(\frac{1}{n}\right)$ concluding the proof of correctness for the algorithm.

There are at most 2^k parity functions to correlate. Thus the sample complexity of the algorithm is $m_1 = O(2^k d \log n + 2^{2d+1} s^2 (\log n + k))$. The running time is $O(n^2 d \log n) + O(2^{2^k} d \log n + 2^k 2^{2d+1} s^2 (\log n + k))$. The first term in the running time is for forming the matrix $\mathbf{R}$. The second term in the running time is for correlation with m_1 samples for each of the 2^k parity functions.

Remark: Here, we have analyzed the algorithm in such a way that the first stage of forming $\mathbf{R}$ and thresholding using $n_{\max}$ only seems to tell us the relevant variables involved. In reality, running the algorithm yields $\mathbf{R}$ which after thresholding at $n_{\max}$ can identify distinct connected components and only the sub-structure of connected components has to be identified in the correlation step. Two variables are in the same connected component if they are in the same hyperedge. But our analysis is for the worst case when there is only one connected component. But it is possible to give a better bound in terms of the size of the largest component instead of k (the total number of relevant variables). We do not pursue that in this proof.

References

[1] E. Kushilevitz and Y. Mansour, "Learning decision trees using the Fourier spectrum," in *SIAM J. Comput.*, vol. 22, no. 6, 1993, pp. 1331–1348.

[2] Y. Mansour, "Randomized interpolation and approximation of sparse polynomials," in *SIAM J. Comput.*, vol. 24, no. 2. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1995, pp. 357–368.

[3] R. Schapire and R. Sellie, "Learning sparse multivariate polynomials over a field with queries and counterexamples," in *JCSS: Journal of Computer and System Sciences*, vol. 52, 1996.

[4] A. C. Gilbert, S. Guha, P. Indyk, S. Muthukrishnan, and M. Strauss, "Near-optimal sparse Fourier representations via sampling," in *Proceedings of STOC*, 2002, pp. 152–161.

[5] P. Gopalan, A. Kalai, and A. Klivans, "Agnostically learning decision trees," in *Proceedings of STOC*, 2008, pp. 527–536.

[6] A. Akavia, "Deterministic sparse Fourier approximation via fooling arithmetic progressions," in *Proceedings of COLT*, 2010, pp. 381–393.

[7] D. Spielman and S. Teng, "Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time," in *JACM: Journal of the ACM*, vol. 51, 2004.

[8] P. Li, "Relational learning with hypergraphs," Ph.D. dissertation, École Polytechnique Fédérale de Lausanne, 2013.

[9] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *Information Theory, IEEE Transactions on*, vol. 52, no. 2, pp. 489–509, 2006.

[10] E. J. Candès and T. Tao, "Decoding by linear programming," *Information Theory, IEEE Transactions on*, vol. 51, no. 12, pp. 4203–4215, 2005.

[11] D. L. Donoho, "Compressed sensing," *Information Theory, IEEE Transactions on*, vol. 52, no. 4, pp. 1289–1306, 2006.

[12] P. Stobbe and A. Krause, "Learning Fourier sparse set functions," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2012, pp. 1125–1133.

[13] S. Negahban and D. Shah, "Learning sparse boolean polynomials," in *Proceedings of the Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*. IEEE, 2012, pp. 2032–2036.

[14] A. Andoni, R. Panigrahy, G. Valiant, and L. Zhang, "Learning sparse polynomial functions," in *Proceedings of SODA*, 2014.

[15] A. T. Kalai, A. Samorodnitsky, and S.-H. Teng, "Learning and smoothed analysis," in *Proceedings of FOCS*. IEEE Computer Society, 2009, pp. 395–404.

[16] R. O'Donnell, *Analysis of Boolean Functions*. Cambridge University Press, 2014.

[17] Yahoo, "Yahoo! webscope dataset ydata-ymessenger-user-communication-pattern-v1_0," http://research.yahoo.com/Academic_Relations.

[18] S. Jukna, *Extremal Combinatorics*. Springer, 2011.