Learning Sparse Polynomial Functions

Alexandr Andoni* MSR
Rina Panigrahy† MSR
Gregory Valiant‡ Stanford and MSR
Li Zhang§ MSR

November 4, 2013

Abstract

We study the question of learning a sparse multivariate polynomial over the real domain. In particular, for some unknown polynomial $f(\vec{x})$ of degree-d and k monomials, we show how to reconstruct f, within error ϵ, given only a set of examples $\bar{x}_i$ drawn uniformly from the n-dimensional cube (or an n-dimensional Gaussian distribution), together with evaluations $f(\bar{x}_i)$ on them. The result holds even in the "noisy setting", where we have only values $f(\bar{x}_i) + g$ where g is noise (say, modeled as a Gaussian random variable). The runtime of our algorithm is polynomial in $n, k, 1/\epsilon$ and C_d where C_d depends only on d. Note that, in contrast, in the "boolean version" of this problem, where $\bar{x}$ is drawn from the hypercube, the problem is at least as hard as the "noisy parity problem," where we do not know how to break the $n^{\Omega(d)}$ time barrier, even for $k = 1$, and some believe it may be impossible to do so.

1 Introduction

Let $f(\vec{x})$ denote a function over n-tuple $\vec{x} = (x_1, ..., x_n) \in X$, drawn from some distribution over X. The problem of learning a class $\mathcal{C}$ of such functions is the problem of approximating a function $f \in \mathcal{C}$ up to a small error, given access to the function evaluations on a random set of samples: $(\vec{x}_1, f(\vec{x}_1)), \ldots (\vec{x}_m, f(\vec{x}_m))$ [Val84]. The fundamental question of learning theory is "For which classes, $\mathcal{C}$, and which distributions, can one efficiently learn?"

In the theoretical computer science literature, some of the most well-studied questions are for learning *boolean* functions, where the function $f \in \mathcal{C}$ has binary inputs and outputs. This is well-reasoned in TCS as many of the encountered functions are boolean. For example, some of the fundamental classes to learn include parity function $f(\vec{x}) = \prod_{i \in S} x_i$ for $\vec{x} \in \{-1, +1\}^n$, DNF functions where $f(\vec{x})$ is a k-term DNF function on $\vec{x} \in \{-1, +1\}^n$, juntas, and others.

As an illustrating example, let us elaborate on learning the parity function, which has emerged as a core problem not only in learning, but also in other fields such as coding and cryptography. Consider a parity function $f(\vec{x}) = \prod_{i \in S} x_i$ on $d = |S|$ variables. In the vanilla "noise free" setting, learning noisy parity is a simple problem: Gaussian elimination works with high probability as long as we have $m = \Omega(n)$ samples. However, such approaches fall apart when we consider the "noisy version", where one gets evaluations $f(\vec{x})$ corrupted by some random noise (say, the output is flipped with some probability $\eta \in (0, 1/2)$). In this setting, despite years of research, the best algorithms run only modestly faster than the trivial $O(n/d)^d$: [BKW03] gave an algorithm running in $2^{O\left(\frac{n}{\log n}\right)}$ time and [Val12] gave an $O(n^{0.8d})$ runtime. In fact the problem (and its variants) has been used as a "hard problem" in cryptographic settings [Ale03, Reg05, Pei09, ABW10].

While the learning theory community has placed more attention on learning in the Boolean setting, many of the real-world functions we encounter are real-valued. It is tempting to speculate that learning real-valued functions is morally similar to learning boolean functions. After all, for example, one of the powerful tools for learning Boolean functions is to use *invariance principle* that allows one to switch between learning Boolean functions and real-valued functions easily (see, e.g., [FGRW09, HKM12]).

However, we argue that there may be notable differences, and a more systematic study of learn-ability of real-valued functions may reveal a richer theory. Surprisingly, even learning of monomials $f(\vec{x}) = \prod_{i \in S} x_i$ over real $\vec{x}$, or sparse low-degree polynomials, has not been studied before, to the best of our knowledge.

A learning algorithm. In this paper, we address this gap and provide one such surprising difference: it turns out that one can learn the real-valued equivalent of the noisy parity function. In particular, consider the function $f(\vec{x}) = \prod_{i \in S} x_i$ where $\vec{x}$ is drawn uniformly at random from the real cube $[-1, 1]^n$, and $|S| \leq d$.

*andoni@microsoft.com
†rina@microsoft.com
‡gregory.valiant@gmail.com
§lzha@microsoft.com

We give an algorithm to learn the function f in time $O(n2^{O(d)})$, which holds *even in the noisy setting*. This should be contrasted to the best runtime of $n^{\Omega(d)}$ for the case when $\vec{x}$ is uniform in $\{-1,+1\}^n$.

More generally, we consider learning sparse polynomials — we call a polynomial k sparse if it can be represented by k-monomials (or k basis polynomials under some product polynomial basis). We show that we can learn a k-sparse degree-d real-valued polynomial function within error ϵ in time $O(n(k/\epsilon)^{O(1)}C_\mu(d))$, under any product input distribution $X = \mu^n$ where $C_\mu(d)$ depends on the distribution μ and degree d only. In particular, for the uniform distribution over $[-1,1]$, $C_\mu(d) = 2^{O(d)}$, and for the Gaussian distribution over $\mathbb{R}$, $C_\mu(d) = 2^{O(d \log d)}$. Our algorithm, termed GROWING-BASIS, appears in Section 2.

We include further discussion on learning arbitrary real function in the statistical query model [Kea98] in Section 3. Also see additional remarks in Section 4.

1.1 Related Work There have been a number of positive learning results that deal with input distributions over high-dimensional real space like in our setting. These include, for example, the results on learning the threshold or intersection of threshold functions [Vem97, BK97, Lon95, KKMS05, KOS08] (but also many others). We note that these results are similar in spirit to learning boolean functions (the output of the function is boolean), in part because the functions use "boolean" operators (threshold), and in part because they use very structured input distributions (such as uniform distribution), allowing one to use Fourier analytic methods. In particular, many results would direct to learn a few Fourier coefficients of the unknown function, and the main challenge is to prove that is this enough. Such algorithms would naturally lead to only a $n^{\Omega(d)}$ time algorithms. (See also the related work of [DLM+07], which give a testing algorithm for polynomials over arbitrary fields, but their runtime depends on the field size.)

For learning real-valued polynomials, a natural classical approach is to perform polynomial interpolation. It seems reasonable to obtain runtime of essentially $O(n^d)$ for learning such a polynomial: for example via Support Vector Machines with a polynomial kernel (see, e.g., [SS02]). However, we are not aware of a more efficient algorithm for polynomial interpolation for sparse polynomials (including of a monomial).

Another related approach to learning sparse polynomial might be via compressed sensing. In particular, one can see a k-sparse polynomial P as a k-sparse vector V_P in the $\binom{n}{d}$ dimensional space, with a coordinate per each monomial of degree d. Evaluation of the function at a random point $\vec{x}$, corresponds to a "measurement", which is a scalar product of the vector V_P with a vector $\vec{x}^{\otimes d}$ obtained from taking all degree-d moments of $\vec{x}$. Even if we could apply general compressed sensing technology for reconstructing V_P from these measurements — such as ℓ_1 LP [Don06, CT06] — such methods usually have a runtime dependent on the ambient dimension, $O(n^d)$ in our case, as opposed to the sparsity k. While there are compressed sensing results that obtain recovery time sublinear in the dimension, they are for very specific or designed measurements (see, e.g., [CRT06, GSTV06, CM06, GSTV07, IR08, GLPS12, PS12] and references therein). It is unclear how to use these techniques to the measurement matrix that we obtain from degree-d moments of random vectors.

Finally, we also mention that there has been numerous efforts in the machine learning community to learn real-valued functions, as these are often much closer to practice than the boolean setting. Of course, the foremost example is exactly the aforementioned kernel SVM algorithm, with $O(n^d)$ runtime. Very recently, there has been work that more directly addresses learning real-valued polynomials, in particular the Vanishing Components algorithm [LLS+13], and the deep-network algorithm of [LSS13]. Yet their setting of learning a polynomial is somewhat different, and does not seem to readily apply to efficiently learning sparse polynomials.

1.2 Techniques We give the gist of our learning algorithm next. The classic method for learning polynomials is by correlating the target function, say P, with each basis polynomial (or "monomial") in certain orthonormal polynomial basis (determined by the sampling distribution). By the orthogonality of the basis, such correlation is precisely the coefficient of each basis polynomial in the representation of P. The difficulty of such a method is that we can learn some coefficient only if we guess a full basis polynomial correctly as otherwise the correlation is 0. Thus, such an approach seems to require checking against n^d degree-d basis polynomials even when P contains only one basis polynomial.

The crux of our approach is to consider a product basis and look for the correlation between the *magnitude* of P and partial product basis functions. This allows us to detect presence of particular variables, or their degrees, to be more precise, in the monomials in P, even if we don't know the entire corresponding monomial. Hence, we can "grow" monomials (polynomial basis elements) variable by variable. Once we have learned a participating monomial, we can remove it from P and recurse on the rest. Unlike the case when we correlate P with basis polynomial, there is "interference" from other monomials. However we show that such interference

cannot happen to the highest degree term. Therefore our algorithm identifies the highest degree term first and then recursively extract the monomials one by one. This is significantly different from the previous approaches in which either the order is unimportant or the learning is done from lower to higher degree.

We term our learning algorithm GROWING-BASIS, and show that it can learn a polynomial in time that is polynomial in n, k and C_μ, where C_μ is a constant dependent on the input distribution, and is $2^{O(d)}$ for the uniform distribution and $2^{O(d \log d)}$ for the Gaussian distribution.

We note that some aspects of our algorithm are similar to an algorithm of [KST09] for learning boolean sparse polynomials under a certain "smoothed model". Specifically, [KST09] consider the situation where the input vector $x \in \{-1, +1\}^n$ comes from a p-biased distribution, where p itself is chosen at random from a fixed range. To learn a sparse polynomial under this distribution, [KST09] also consider correlation of the square of the unknown polynomial P with a "test variable". However the two settings are quite different: for example to learn whether a variable x_k participates in a (noisy) parity $P = \prod_{i \in S} x_i$ can be deduced directly from the correlation $\mathbb{E}[Px_k]$ under the "smoothed model"[1], but not under the uniform or Gaussian distribution. Furthermore, in the boolean domain $\{-1, +1\}^n$, it is enough to consider only multi-linear polynomials, as opposed to general higher-degree polynomials as in the real case studied here.

2 Learning Algorithm for Sparse Polynomials

In this paper, we consider learning polynomials for samples drawn from a given distribution D. The inner product between two function f, g is defined as $\langle f, g \rangle = \mathbb{E}_{x \sim D} f(x)g(x)$. Define $\|P\| = \sqrt{\langle P, P \rangle}$. Throughout the paper, we assume $\|P\| = 1$. We say a procedure learns a function P within error ϵ if it outputs a function $\widehat{P}$ such that $\|\widehat{P} - P\| \leq \epsilon$.

We further consider the case when D is a product distribution $D = \mu_1 \times \mu_2 \ldots \times \mu_n$. We can construct an associated basis for polynomials as follows. For each i, construct polynomials $H_0(x_i), H_1(x_i), \ldots$ where H_t is of degree t by performing Gram-Schmidt orthonormalization on $1, x, x^2, \ldots$ with respect to the inner product defined as $\langle f, g \rangle = \int f(x)g(x)\mu_i(x)dx$. This way, we obtain $\langle H_i(x), H_j(x) \rangle = \delta_{ij}$(where $\delta_{ij} = 1$ if $i = j$ and 0 otherwise), and $H_i(x)$ has the degree i. For example, if the distribution μ is the uniform distribution over the interval $[-1, 1]$ then this produces the Legendre polynomial basis; if it is the Gaussian distribution then it gives the Hermite polynomials. We will assume that these polynomials have been normalized to have unit norm, with respect to their corresponding distributions. The product basis is constructed by taking the product of basis polynomials for different x_i's. For presentation simplicity, we assume all the μ_i's are identical. For $S = (S_1, \ldots, S_n)$ where S_i is a non-negative integer, define $H_S(\vec{x}) = \prod_i H_{S_i}(x_i)$. Then H_S consist of an orthonormal basis with respect to D. Any polynomial P can be written as $P(\vec{x}) = \sum_S a_S H_S(\vec{x})$ for some reals $a_S \in \mathbb{R}$. The degree $\deg(P)$ of P is defined as $\max_{S: a_S \neq 0} \sum_i S_i$. The sparsity k is defined as the number of S such that $a_S \neq 0$. While the sparsity may vary according to the basis, it is robust for product basis according to the following observation.

LEMMA 2.1. *For any two product basis H' and H'', if $P(\vec{x})$ is k-sparse in H', then it is $k2^d$-sparse in H'' where $d = \deg(P)$. Furthermore, a polynomial with k monomials is $k2^d$ sparse in any product polynomial basis H.*

Proof. Suppose that $H'_S(\vec{x}) = \prod_i H'_{S_i}(x_i)$ is a term in the representation of $P(\vec{x})$ in H' basis. Each $H'_{S_i}(x_i)$ can be written as $\sum_{j=0}^{S_i} H''_j(x_i)$ since H'' is a basis. If we expand $H'_S(\vec{x})$ with respect to H'' basis, we obtain $\prod_i (S_i + 1) \leq 2^{\sum_i S_i} \leq 2^d$ terms. Hence $P(\vec{x})$ has at most $k2^d$ terms in H'' basis. Similarly, each $\prod_{i \in S} x_i^{d_i}$ can be decomposed into a sum of basis elements with 2^d terms.

First we show that if we can accurately measure the correlation between polynomials, there is an algorithm to learn P within time $O(kdn)$. Then we extend it to the case when we can only estimate the correlation by random samples, thereby proving our main result. The number of required samples will depend on the distribution μ, and in particular on two parameters defined later. In particular, for the uniform distribution over $[-1, 1]^n$, the final runtime (and sample complexity) becomes $O(\text{poly}(n, k, 2^d, 1/\epsilon))$; and for Gaussian distribution — $O(\text{poly}(n, k, 2^{d \log d}, 1/\epsilon))$.

2.1 Learning with a correlation oracle We first consider the idealized case when an oracle returns the accurate correlation between $P(\vec{x})^2$ and any basis function $H_t(\vec{x})$. We first show that with such an oracle, we can compute $P(\vec{x})$ in time $O(kdn)$. We prove the following theorem:

THEOREM 2.1. *Fix a distribution $D = \mu^n$ on $\mathbb{R}^n$, and let $\{H_i\}$ be the polynomial basis under μ. Fix a*

[1] Once we condition on p, which one can essentially learn for each sample by can computing it from the empirical bias of ones. Then $\mathbb{E}[Px_k] = p^{|S|}$ if $k \in S$ and $p^{|S|+1}$ otherwise.

polynomial P that is n-variate, d-degree, and k-sparse in $\{H_i\}_i^n$, for which we have access to a correlation oracle for $\langle f, P\rangle$ and $\langle f, P^2\rangle$ for any polynomial f of choice. The Growing-Basis *algorithm can learn the polynomial P in time $O(kdn)$ (i.e., with this many oracle calls).*

The gist of the algorithm is to detect all the basis functions $H_S(\vec{x}) = \prod_{i=1}^n H_{S_i}(x_i)$ in P. This is done by correlating the basis function with $P(\vec{x})^2$ and by growing the basis function variable by variable. More precisely, we examine the correlation $\langle H_{2t}(x_i), P^2(\vec{x})\rangle$ to detect presence of x_i in P. Let d_i denote the maximum degree of x_i in P. We will show that if $t = d_i$, the above correlation is non-zero; and if $d_i < t$, the correlation is 0. With these properties, we can check, from d down to 0, for non-zero correlation with $H_{2t}(x_1)$ to detect d_1. Once we obtain d_1, by similar method we detect the highest degree of x_2 among all the terms that contain the factor $H_{d_1}(x_1)$. By repeating this, we can detect the "largest" term in the "lexicographic order", using only $O(dn)$ correlation queries. We then subtract this term and repeat the process until we discover all the terms.

Algorithm 1 describes our Growing-Basis algorithm more formally. In the algorithm, we assume that we have the access to an oracle that, for any given polynomials f, can evaluate the correlation of the form $\langle f(\vec{x}), P(\vec{x})\rangle$ and $\langle f(\vec{x}), P(\vec{x})^2\rangle$. From such an oracle, we can also evaluate, for any polynomial $f(\vec{x}), g(\vec{x})$, $\langle f(\vec{x}), (P(\vec{x}) - g(\vec{x}))^2\rangle = \langle f(\vec{x}), P(\vec{x})^2\rangle + \langle -2f(\vec{x})g(\vec{x}), P(\vec{x})\rangle + \langle f(\vec{x}), g(\vec{x})^2\rangle$.

In the following, we will show the correctness of the Growing-Basis algorithm. The main observation we will use is that for any t, $H_t(x)^2$ has a constant term and a term corresponding to $H_{2t}(x)$. That is $H_t^2(x) = \sum_{j=0}^{2t} c_{t,j} H_j(x)$. The constant term c_0 corresponding to $H_0(x) = 1$ must be one because $\langle H_t(x), H_t(x)\rangle = 1$ for the normalized basis function H_t, i.e.,

$$H_t^2(x) = 1 + \sum_{j=1}^{2t} c_{t,j} H_j(x)\,. \tag{2.1}$$

Thus the product of such squares when expanded will produce basis polynomials in individual variables. Let $c_t = c_{t,2t}$.

Lemma 2.2. *Let $b_t = \langle H_{2t,0,\ldots,0}(\vec{x}), P^2(\vec{x})\rangle = \langle H_{2t}(x_1), P^2(\vec{x})\rangle$. If $t > d_1$, then $b_t = 0$. If $t = d_1$, then $b_t = c_t \sum_{S:S_i=d_1} a_S^2$, where $d_1 = \max_{S:a_S\neq 0} S_1$ is the maximum degree of x_1 in $P(\vec{x})$.*

Algorithm 1 Growing-Basis polynomial learning algorithm

Input: correlation oracles for $\langle f(\vec{x}), P(\vec{x})\rangle$ and $\langle f(\vec{x}), P(\vec{x})^2\rangle$
Output: $\widehat{P}$

```
1:  P̂ = 0;
2:  while ⟨1, (P − P̂)²⟩ > 0 do
3:    H = 1;
4:    B = 1;
5:    for r = 1 to n do
6:      for t = d downto 0 do
7:        Compute C = ⟨H · H_{2t}(x_r), (P − P̂)²⟩;
8:        if C > 0 then
9:          H := H · H_{2t}(x_r);
10:         B := B · H_t(x_r);
11:         continue to 14;
12:       end if
13:     end for
14:   end for
15:   Compute a = ⟨B, P⟩;
16:   Set P̂ = P̂ + a · B
17: end while
```

Proof. We examine the expansion of $P(\vec{x})^2$:

$$\begin{aligned} P(\vec{x})^2 &= \left(\sum_S a_S \prod_{i=1}^n H_{S_i}(x_i)\right)^2 \\ &= \sum_S a_S^2 \prod_{i=1}^n H_{S_i}(x_i)^2 + \sum_{S\neq T} a_S a_T \prod_{i=1}^n H_{S_i}(x_i) H_{T_i}(x_i) \\ &=: \Delta_1 + \Delta_2\,. \end{aligned}$$

Since $H_{S_1}(x)^2 = \sum_{j=0}^{2S_1} c_{S_1,j} H_j(x)$, when $t > S_1$, $\langle H_{2t}(x_1), H_{S_1}(x_1)^2\rangle = 0$. Similarly, when $t > S_1, T_1$, $\langle H_{2t}(x_1), H_{S_1}(x_1) H_{T_1}(x_1)\rangle = 0$. This shows that when $t > d_1 = \max_{S:a_S\neq 0} S_1$, $b_t = 0$.

For $t = d_1$, we will show that

$$\langle H_{2t,0,\ldots,0}(\vec{x}), \Delta_1\rangle = \sum_{S:S_1=d_1} a_S^2 \quad \text{and} \quad \langle H_{2t,0,\ldots,0}(\vec{x}), \Delta_2\rangle = 0\,.$$

Consider each term $b_S = a_S^2 \prod_{i=1}^n H_{S_i}(x_i)^2$ in Δ_1. By the above argument, if $S_1 < d_1$, then $\langle H_{2t}(x_1), b_S\rangle = 0$. If $S_1 = d_1$, by (2.1), we can write b_S as

$$b_S = a_S^2 \prod_{i=1}^n \left(\sum_{j=0}^{2S_i} c_{S_i,j} H_j(x_i)\right)\,.$$

We can expand the product and write b_S in the sum of product basis. We check the coefficient of the term $H_{2t,0,\ldots,0}(\vec{x})$ in the expansion. This coefficient is

precisely the product the coefficient of $H_{2t}(x_1)$ for x_1 and $H_0(x_i)$ for x_i where $2 \le i \le n$. By (2.1), it is exactly $c_{t,2t} = c_t$. Hence $\langle H_{2t,0,\ldots,0}(\vec{x}), b_S\rangle = c_t a_S^2$. Summing over all the S, we have $\langle H_{2t,0,\ldots,0}(\vec{x}), \Delta_1\rangle = c_t \sum_{S:S_i=t} a_S^2$.

Consider a cross term $b_{ST} = a_S a_T \prod_{i=1}^n H_{S_i}(x_i)H_{T_i}(x_i)$ in Δ_2.

$$\begin{aligned}
&\langle H_{2t,0,\ldots,0}(\vec{x}), b_{ST}\rangle \\
&= a_S a_T \langle H_{2t}(x_1), H_{S_1}(x_1)H_{T_1}(x_1)\rangle \prod_{i=2}^n \langle H_0(x_i), H_{S_i}(x_i)H_{T_i}(x_i)\rangle \\
&= a_S a_T \langle H_{2t}(x_1), H_{S_1}(x_1)H_{T_1}(x_1)\rangle \prod_{i=2}^n \langle H_{S_i}(x_i), H_{T_i}(x_i)\rangle\,.
\end{aligned}$$

Since $S \neq T$, there must exist i such that $S_i \neq T_i$. If $i \neq 1$, then $\langle H_{S_i}(x_i), H_{T_i}(x_i)\rangle = 0$, and consequently the above inner product is 0. If $i = 1$, since $t \ge S_1, T_1$, and $S_1 \neq T_1$, we have $2t > S_1 + T_1$. Since $H_{S_1}(x_1)H_{T_1}(x_1)$ has degree S_1+T_1, it can be written as the linear combination of $H_0(x_1), \ldots, H_{S_1+T_1}(x_1)$. As $S_1 + T_1 < 2t$, we must have

$$\langle H_{2t}(x_1), H_{S_1}(x_1)H_{T_1}(x_1)\rangle = 0\,.$$

Combining both cases, we have $\langle H_{2t,0,\ldots,0}(\vec{x}), \Delta_2\rangle = 0$. This completes the proof.

The above lemma can be used to detect d_1, the highest degree of x_1 present in one of the elements in the basis representation of $P(\vec{x})$. We can extend this analysis to finding a complete product basis iteratively. We denote by $\preceq$ the alphabetic order between two equal length integer sequence, i.e. $S_1 \ldots S_r \preceq T_1 \ldots T_r$ if there is $1 \le i \le r+1$ such that $S_j = T_j$ for $1 \le j < i$ and $S_i < T_i$. We define the r-maximal term in P as follows.

DEFINITION 2.1. *A basis polynomial H_T for $T = T_1 \ldots T_r 0 \ldots 0$ is called r-maximal for P, if there exists S such that $T \preceq S$ and $a_S \neq 0$; and if there is no S such that $T_1 \ldots T_r \prec S_1 \ldots S_r$ and $a_S \neq 0$.*

Lemma 2.2 amounts to saying that we can detect 1-maximal basis. Now we show the same argument can be used to detect the r-maximal basis for any $1 \le r \le n$. This is done iteratively. Suppose that we have already detected the r-maximal term $t_1 t_2 \cdots t_r$. We now extend it to the $r+1$-maximal term by checking the correlation between $H'_t := H_{2t_1,2t_2,\cdots,2t_r,2t,0,\cdots,0}$ and $P(\vec{x})^2$ for $t = d, d-1, \ldots, 0$.

Again we expand $P(\vec{x})^2$ into squared terms and cross terms. For a squared term

$$b_S = a_S^2 \prod_{i=1}^n H_{S_i}(x_i)^2 = a_S^2 \prod_{i=1}^n \left(\sum_{j=1}^{2S_i} (c_{S_i,j} H_j(x_i) + 1) \right)\,.$$

We can expand the product and check the coefficient of $H_{2t_1,2t_2,\cdots,2t_r,2t,0,\cdots,0}$ in the above expansion. Consider the first r variables, since $t_1 t_2 \cdots t_r$ is r-maximal, there are two cases 1) when there is $i \le r$ such that $S_i < t_i$. In this case, $H_{2t_i}(x_i)$ would not appear in the expansion so $\langle H'_t, b_S\rangle = 0$; 2) when $S_i = t_i$ for all $1 \le i \le r$. In this case, if $t > S_{r+1}$, again we have $\langle H'_t, b_S\rangle = 0$. When $t = S_{r+1}$, the coefficient of H'_t in the expansion is $a_S^2 \prod_{i=1}^{r+1} c_{S_i}$. Suppose that $t_1, \cdots, t_{r+1}$ is $r+1$-maximal. Then if $t > t_{r+1}$, $\langle H'_t, b_S\rangle = 0$, and if $t = t_{r+1}$,

$$\langle H'_t, b_S\rangle = \sum_{S:\forall 1\le i\le r+1 \quad S_i=t_i} a_S^2 \prod_{i=1}^{r+1} c_{S_i}\,.$$

Now for the cross term $b_{ST} = a_S a_T \prod_i H_{S_i}(x_i)H_{T_i}(x_i)$. Again since $S \neq T$, there exists i such that $S_i \neq T_i$. Consider the minimum of such i's. If $i > r+1$, clearly $\langle H'_t, b_{ST}\rangle = 0$. When $i \le r+1$, by the minimality of i, $t_j = S_j = T_j$ for $j < i$. By the maximality of $t_1 \cdots t_r t$ (since we only consider $t \ge t_{r+1}$), it must be that $t_i \ge S_i, T_i$. Since $S_i \neq T_i$, we have $2t_i > S_i + T_i$ and hence $\langle H'_t, b_{ST}\rangle = 0$.

From the above analysis, we now have the following generalization of Lemma 2.2

LEMMA 2.3. *Suppose that T is r-maximal in $P(\vec{x})$. Suppose that t_{r+1} is the maximum degree of x_{r+1} among all the basis polynomials that contain $\prod_{i=1}^r H_{t_i}(x_i)$. Let $b_t = \langle H_{2t_1,\cdots,2t_r,2t,0,\cdots,0}(\vec{x}), P(\vec{x})^2\rangle$. If $t > t_{r+1}$, then $b_t = 0$, and if $t = t_{r+1}$, then*

$$b_t = \prod_{i=1}^{r+1} c_{t_i} \cdot \sum_{S\,:\,S_i=t_i,\forall 1\le i\le r+1} a_S^2\,.$$

The above lemma gives an algorithm for finding one non-zero basis, or more precisely the maximal (in alphabetical order) non-zero basis, in the expansion of $P(\vec{x})$. Suppose that we already decide that P contains a non-zero basis which contains $H_{t_1}(x_1)\ldots H_{t_r}(x_r)$, we can then apply the above lemma to check the correlation between $H_{2t_1,\cdots,2t_r,2t,0,\cdots,0}(\vec{x})$ and $P(\vec{x})^2$ by setting t from d down to 0. We set t_{r+1} to the largest t such that the above correlation is positive. And repeat this process until $r = n$. Once we discover a basis and its coefficient, we can subtract it from P, and repeat the process until we obtain all the non-zero basis of P. This proves the correctness of Algorithm 1.

The query time of the algorithm is $O(dn)$ for discovering each non-zero basis. So in total it is $O(kdn)$ time.

2.2 Learning by random samples In the previous section, we showed how to learn a sparse polyno-

mial efficiently when we are given a correlation oracle. We now extend the algorithm to the case when we only obtain samples of the form $\vec{x}, P(\vec{x})$. We will use the algorithm from above but using empirical estimate of the correlations computed from the given samples. To be able to bound the number of required samples, we need to define the following two parameters: $M_P = \max_H E_{\vec{x}\sim D}[H(\vec{x})^2 P(\vec{x})^4]$ where H ranges over all the degree $2d$ basis with respect to the distribution D, and $\tau_d = \min_{t\le d} c_t$. Also, we define the error between P and $\widehat{P}$ as $\|\widehat{P} - P\| = \sqrt{\langle \widehat{P} - P, \widehat{P} - P\rangle} = \sqrt{\sum_S (a_S(\widehat{P}) - a_S(P))^2}$.

We prove the following theorem with the dependence on the values of M and τ. Later we show that we can bound M_P as a function of the distribution D, and that both parameters are at most exponential in d for standard input distributions such as the uniform and the Gaussian distribution.

THEOREM 2.2. *Suppose we are given a polynomial P on n variables, of degree d and sparsity k. Let $\vec{x} \in \mathbb{R}^n$ be drawn according to a product distribution μ^n, where μ has parameters M and τ_d. Then, given $E = O(M_P \operatorname{poly}(n, k, (1/\tau_d)^d, 1/\epsilon))$ samples, we can learn P within error ϵ with high probability. The running time is bounded by $O(End)$. When the sampled function value has independent noise, e.g. $\widetilde{f}(\vec{x}) = f(\vec{x}) + g$ where g is independent Gaussian noise $\mathcal{N}(0, \sigma^2)$, the same bounds apply with a multiplicative factor of $\operatorname{poly}(1+\sigma)$.*

Proof. In order to emulate the correlation oracle, we estimate $C_H = \langle H(\vec{x}), P(\vec{x})^2\rangle = \int H(\vec{x})P(\vec{x})^2\mu(\vec{x})d\vec{x}$ empirically. For m random samples $(\vec{x}_i, P(\vec{x}_i))$, the estimate becomes:

$$\widehat{C}_H = \sum_{i=1}^{m} H(\vec{x}_i)P(\vec{x}_i)^2/m\,.$$

$\widehat{C}_H$ approximates C_H within additive error ϵ_1 with constant probability, as long as we have $m = O\left(\frac{\mathbb{E}[H^2P^4]}{\epsilon_1^2}\right)$ samples (using Chebyshev inequality). To boost the probability of success to $1-\delta$, we repeat the estimator $O(\log 1/\delta)$ times and take the median of the obtained values.

By Lemma 2.3, $\langle H_{2t_1,\ldots,2t_n}(\vec{x}), P(\vec{x})^2\rangle = \prod_i c_{t_i} a^2_{t_1,\ldots,t_n}$. Since $c_0 = 1$, we have that $\prod_i c_{t_i} \ge \tau_d^d$, where d is the degree of $P(\vec{x})$. In addition, by Theorem 2.1, we only make $O(knd)$ correlation queries, so to achieve the error of ϵ, it suffices to set $\epsilon_1 = \epsilon^2/(kn(1/\tau_d)^d)$ and hence take $O(M_P \operatorname{poly}(n, k, (1/\tau_d)^d, 1/\epsilon))$ samples. When we run algorithm in Lemma 2.3 with respect to the simulated correlation oracle, we set a cutoff $\kappa = \epsilon/(kn(1/\tau_d)^d)$ in steps 2 and 8: namely, if the correlation is below κ, we treat it as 0. The error now may come from several sources: 1) some small terms missed due to the cutoff; 2) accumulated error by the subtracting process; 3) the estimation error of the coefficient of a non-zero term. By the choice of ϵ_1 and κ and a simple union bound, we can see the total error is bounded by ϵ with high probability.

The computational time requires the extra factor to evaluate $H(\vec{x})$, which is bounded by $O(nd)$ per evaluation.

When the sampled function value has independent noise, since we compute the correlation up to the fourth moment, by standard concentration bound we can obtain the claimed bounds.

The above theorem is stated in terms of M_P and τ_d. Now we show specific bounds for these two values for the uniform and Gaussian distributions. For the uniform distribution over $[-1,1]^n$, the corresponding orthogonal polynomial basis is the (normalized) Legendre polynomials; and for Gaussian distribution, the (normalized) Hermite polynomials [Sze89]. From the standard representation of both polynomials, it is easily seen, by checking the coefficient of the leading monomial of H_d, that $c_d = \Omega(1)$ for both families. To bound M_P, we assume that $P(\vec{x})$ is normalized, i.e $\langle P(\vec{x}), P(\vec{x})\rangle = 1$. Denote by $M_{d,k} = \max_{P,H} E[H^2P^4]$ where the maximum is over degree-d, k-sparse polynomial P with unit norm and degree-$2d$ basis H. We first consider the case when D is uniform over $[-1,1]^n$, and H is the product of Legendre polynomials. For the normalized Legendre polynomial, we have that $|H_{d_i}(x_i)| \le \sqrt{2d_i+1}$ for $x \in [-1,1]$. Hence, for $\vec{x} \in [-1,1]^n$,

$$|H_S(\vec{x})| = \prod_i |H_{S_i}(x_i)| \le \prod_i \sqrt{2S_i+1} \le \prod_i 2^{S_i} = 2^d\,.$$

Hence $|P(\vec{x})| = |\sum_S a_S H_S(\vec{x})| \le 2^d \sum_S |a_S|$ for $\vec{x} \in [-1,1]^n$. By Parseval identity, $\sum_S a_S^2 = 1$, and by P is k-sparse, $\sum_S |a_S| \le \sqrt{k}$. Therefore $|P(\vec{x})| \le 2^d\sqrt{k}$ for $\vec{x} \in [-1,1]^n$. Hence $H(\vec{x})^2P(\vec{x})^4 \le 2^{6d}k^2$ if $\deg(P) \le d$, and H is a degree $2d$ basis. That is $M_{d,k} = O(\operatorname{poly}(2^d, k))$. Similarly, we can bound $M_{d,k}$ for Gaussian distribution and Hermite polynomials. Since the range of Gaussian distribution is $(-\infty,\infty)^n$, we can no longer bound the uniform norm of $H(\vec{x})$ and $P(\vec{x})$. But it suffices to consider the range where $|x_i| = O(\sqrt{d\log d})$. By examining the coefficients of Hermite polynomial, we can apply the same reasoning as above to obtain a bound of $M_{d,k} = O(\operatorname{poly}(2^{d\log d}, k))$. We emphasize that the dependence of sampling complexity on d is $2^{O(d)}$ for the uniform distribution and $2^{O(d\log d)}$

for Gaussian distribution, which is constant for constant d.

According to Lemma 2.1, changing the product basis, for example to the standard polynomial basis, only incurs a multiplicative factor of $2^{O(d)}$. Hence, we have

COROLLARY 2.1. *If $P(\vec{x})$ is k-sparse in Legendre or standard polynomial basis, it can be learned within ϵ error with $O(\text{poly}(n, k, 2^d, 1/\epsilon))$ samples uniformly from $[-1, 1]^n$. Similar bound, replacing 2^d by $2^{d \log d}$, holds for Hermite or standard polynomial basis with respect to the Gaussian distribution.*

3 Statistical Queries and Real Functions

The *statistical query* learning model captures a large class of noise-robust learning algorithms [Kea98]; informally, it is the class of algorithms whose only interaction with the function to be learned is via noisy estimates of certain statistics of the example-label pairs $(x, f(x))$. In this section we briefly mention one of the complications of considering learning real-valued functions in the statistical query model.

DEFINITION 3.1. *Given a distribution D over a set S, and a function $f : S \to \mathbb{R}$, a* statistical query oracle *of tolerance τ takes, as input, an arbitrary function $g : S \times \mathbb{R} \to \mathbb{R}$, and outputs some value v satisfying $|v - E_{x \leftarrow D}[g(x, f(x))]| \leq \tau$. A* statistical query algorithm *for learning a concept class $\mathcal{F} = \{f\}$ is an algorithm whose only access to the function to be learned $f \in \mathcal{F}$ is via (possibly adaptive) queries to a statistical query oracle.*

The most well-known impossibility result for the statistical query learning model is the problem of learning parity with noise [BFJ+94]. Specifically, given the concept class consisting of the 2^n parity functions over the Boolean cube, any statistical query algorithm that, with high probability, can weakly learn this concept class must either ask $2^{\Omega(n)}$ queries, or ask queries with inverse exponential tolerance. The intuition for the difficulty of learning this concept class via statistical queries is that for any pair of distinct parity functions, f, f' the distributions $f(x), f'(x)$ induced by $x \leftarrow \{0, 1\}^n$ are independent. In the following example, we describe a natural real-valued analog of this function, and show that it *is* statistically query learnable.

EXAMPLE 3.1. *Consider the class of functions $\{f_S\}$, with $f_S : [0, 2\pi]^n \to [-1, 1]$, indexed by each subset $S \subset [n]$. Define $f_S(x) = \cos(\sum_{i \in S} x_i)$. Note that each such function is $\sqrt{n}$-Lipschitz, and that for $S \neq S'$, the distributions of $f_S(x), f_{S'}(x)$ over $x \leftarrow Unif([0, 2\pi]^n)$ are independent. Nevertheless, it is easy to construct a set of n statistical queries which, when queried with tolerance $\tau < 1/2$, allow one to learn f_S: let $g_i(x, y) = \pm 1$ according to whether there exists a set $T \subset [n]$ such that $i \in T$ and $\cos\left(\sum_{j \in T} x_j\right) = y$. Note that $E[g_i(x, y)]$ is the indicator of whether $i \in S$.*

The above example illustrates the potential complications that arise from considering real-valued functions in the statistical query model. This is not strictly an issue with the infinite dimensionality of the space of functions from $\mathbb{R} \to \mathbb{R}$; if we restrict the domain and range of the functions to be an extremely fine finite grid, the above example still persists. The following theorem quantifies this relationship between the statistical query learnability and the granularity of the range of the functions in the class.

THEOREM 3.1. *Given an arbitrary finite set S, a finite set $T \subset [0, 1]$, a concept class $\mathcal{F} \subset \{f : S \to T\}$ and a distribution D over S, if there exist d functions $f_1, \ldots, f_d \in \mathcal{F}$ with identical marginal distributions of $f_i(x)$ for $x \leftarrow D$, with the property that for all $h_1, h_2 : T \to \mathbb{R}$ for which $E_{x \leftarrow S}[h_\cdot(f_i(x))^2] \leq 1$, for $i \neq j$,*

$$|E_{x \leftarrow D}[h_1(f_i(x)) \cdot h_2(f_j(x))] - E_{x \leftarrow D}[h_1(f_i(x))] \cdot E_{x \leftarrow D}[h_2(f_j(x))]| \leq \frac{1}{d^3},$$

then any statistical query algorithm that asks queries with tolerance $\tau \geq 2\sqrt{2|T|}/d^{2.5}$ will require at least $\frac{d\tau^2}{8|T|}$ statistical queries in order to (weakly) learn concept class $\mathcal{F}$ over D.

The proof of the above theorem follows the general outline of the proof of the statistical query lower bound of [BFJ+94].

We begin the proof by noting that the set of functions that map S to $\mathbb{R}$ is a $|S|$-dimensional inner product space, under the inner product $\langle f, f' \rangle = E_{x \leftarrow D}[f(x) \cdot f'(x)]$. Let $sp(f) \subset \{f' : S \to \mathbb{R}\}$ denote the span of $h_1(f(x)) - E_{x \leftarrow D}[h_1(f(x))], \ldots, h_{|T|}(f(x)) - E_{x \leftarrow D}[h_{|T|}(f(x))]$, where $h_i : T \to \mathbb{R}$ is defined so that for $T = \{x_1, \ldots, x_{|T|}\}$, $h_i(x_i) = 1/\sqrt{|T|}$ and for $j \neq i$, $h_i(x_j) = 0$. Note that these functions form an orthonormal basis for maps from $T \to \mathbb{R}$ under the inner product $\langle h, h' \rangle = E_{x \leftarrow Unif[T]}[h(x) \cdot h'(x)]$.

For each function f_i, let $\alpha_{i,1}, \ldots, \alpha_{i,|dim(sp(f_i))|}$ denote an orthonormal basis for $sp(f_i)$. We now extend the set $\{\alpha_{i,j}\}$ into a basis for functions that map $S \to \mathbb{R}$, by adding functions $\alpha'_1, \ldots, \alpha'_s$, with the properties that $||\alpha'_i|| = 1$, $\langle \alpha'_k, \alpha_{i,j} \rangle = 0$, and $\langle \alpha'_i, \alpha'_j \rangle = 0$.

We extend the distribution D over S to the distribution D' over $S \times T$ defined by $D' = D \times unif(S)$. For each $\alpha_{i,j}$, define $\beta_{i,j,k} : S \times T \to \mathbb{R}$ by $\beta_{i,j,k}(x,y) = h_k(y)\alpha_{i,j}(x)$, for h_k as defined above. Similarly, we define $\beta'_{i,k} = h_k(y)\alpha'_i(x)$. Note that the space of functions from $S \times T \to R$ is a finite dimensional inner product space (over the reals) with respect to the inner product defined by $\langle t, t' \rangle = E_{(x,y)\leftarrow D'}[t(x,y) \cdot t'(x,y)]$, spanned by the set $\{\beta_{i,j,k}\} \cup \{\beta'_{i,k}\}$.

We first claim that the $\alpha_{i,j}$, and hence the $\beta_{i,j,k}$ are linearly independent.

LEMMA 3.1. *The set $\{\alpha_{i,j}\} \cup \{\alpha'_i\}$ are linearly independent.*

Proof. Note that since, by definition, the span of $\{\alpha_{i,j,k}\}$ is orthogonal to the span of $\{\alpha'_i\}$, it suffices to prove that it cannot happen that (without loss of generality) $\alpha_{1,1} = \sum_{(i,j)\neq(,1)} c_{i,j}\alpha_{i,j}$, for constants $c_{i,j}$. Assume, for the sake of contradiction, that such an identity holds. Thus we have the following:

$$\begin{aligned} 0 &= ||\alpha_{1,1} - \sum c_{i,j}\alpha_{i,j}|| \\ &= 1 - 2\sum_{i>1,j} c_{i,j}\langle \alpha_{1,1}, \alpha_{i,j}\rangle \\ &\quad + \sum_{i,j,i',j'} c_{i,j}c_{i',j'}\langle \alpha_{i,j}, \alpha_{i',j'}\rangle \\ &\geq 1 + \sum_{i,j} c_{i,j}^2 - 2\sum_{i>1}\left|\langle \alpha_{1,1}, \sum_j c_{i,j}\alpha_{i,j}\rangle\right| \\ &\quad - \sum_{i\neq i'}\left|\langle \sum_j c_{i,j}\alpha_{i,j}, \sum_j c_{i',j}\alpha_{i',j}\rangle\right| \end{aligned} \tag{3.1}$$

Defining $m = \max_{i\in[d]}\sqrt{\sum_{j\in|T|} c_{i,j}^2}$ and using our assumption that for h, h' with $E[h(f_i(x))] = 0 = E[h'(f_{i'}(x))]$ and $||h(f_i)|| = ||h'(f_{i'})|| = 1$, for $i \neq i'$, $\langle h(f_i), h'(f_{i'})\rangle \leq 1/d^3$, we have that

$$2\sum_{i>1}|\langle \alpha_{1,1}, \sum_j c_{i,j}\alpha_{i,j}\rangle| \leq \frac{2}{d^3}\sum_{i>1}\sqrt{\sum_j c_{i,j}^2} \leq \frac{2m}{d^2},$$

and the final term in Equation 3.1 is similarly upper bounded by $\frac{m^2}{d}$, whereas, trivially, the second term in Equation 3.1 is bounded below by m^2. Hence Equation 3.1 yields the following contradiction:

$$0 \geq 1 + m^2(1 - \frac{1}{d}) - \frac{2m}{d^2} > 0,$$

where the final inequality holds for any $d \geq 2$.

The above establishes that, although the vectors $\{\beta_{i,j,k}, \beta'_{i,k}\}$ are not orthogonal, any function $g : S \times T \to \mathbb{R}$ has a unique representation in this basis. The following lemma shows that if $||g|| = 1$, the sum of the squares of the coefficients in such a representation is bounded by 2.

LEMMA 3.2. *For a function $g : S \times T \to \mathbb{R}$ with $E_{(x,y)\leftarrow D'}[g(x,y)^2] = 1$, and $g = \sum_{i,j,k} c_{i,j,k}^2\beta_{i,j,k} + \sum_{i,k} c_{i,k}\beta'_{i,k}$, it holds that $\sum_{i,j,k} c_{i,j,k}^2 + \sum_{i,k} c_{i,k}^2 \leq 2$.*

Proof. As the spans of the βs and the β's are orthogonal, and the $\beta'_{i,k}$ form an orthonormal basis for the space they span, it suffices to prove the claim for a function $g \in span(\{\beta_{i,j,k}\})$.

Note that for any $(j,k) \neq (j',k')$, $\langle \beta_{i,j,k}, \beta_{i,'j',k'}\rangle = 0$, hence $\sum_{j,k} c_{i,j,k}^2 = ||\sum_{j,k} c_{i,j,k}\beta_{i,j,k}||^2$. Let $c_i = \sqrt{\sum_{j,k} c_{i,j,k}^2}$ and $v_i = \frac{1}{c_i}\sum_{j,k} c_{i,j,k}\beta_{i,j,k}$. Hence we have $g = \sum_i c_i v_i$. Recall that $||v_i|| = 1$, and for $i \neq j$, $|\langle v_i, v_j\rangle| \leq \frac{1}{d^3}$.

The proof now follows easily. First, assuming without loss of generality that $\max_i c_i = c_1$, we show that $c_1 < 2$. If this were not the case, then projecting g onto v_1 would yield a vector of length at least $c_1 - 2\sum_{i\geq 2} c_i\langle v_1, v_i\rangle \geq c_1 - 2c_1/d^2 > c_1/2$, (for $d \geq 2$) contradicting the fact that $||g|| = 1$, and hence the norm of its projection onto any unit vector can be at most 1.

Given this bound on the maximum c_i, and the fact that $1 = ||g|| = \sum_i c_i^2 + \sum_{i\neq jj} c_i c_j\langle v_i, \langle v_j\rangle$, we have that $\sum_i c_i^2 \leq 1 + d^2\frac{4}{d^3} < 2$ for any $d > 4$.

We now complete our proof of Theorem 3.1.

Proof. [Proof of Theorem 3.1] Given a statistical query $g(x,y) = \sum_{i,j,k} c_{i,j,k}\beta_{i,j,k} + \sum_{i,k} c_{i,k}\beta'_{i,k}$ expressed in the basis $\{\beta_{i,j,k}\} \cup \{\beta'_{i,k}\}$, consider $E_{x\leftarrow D}[g(x, f_r(x))]$ for some $r \in [d]$. We first consider the contribution from the $c_{i,k}\beta'_{i,k}$ terms:

$$\begin{aligned} &E_{x\leftarrow D}[c_{i,k}\beta'_{i,k}(x, f_r(x)] \\ &\quad = c_{i,k}E_{x\leftarrow D}[h_k(f_r(x))\alpha'_i(x)] \\ &\quad = c_{i,k}\left(E_{x\leftarrow D}\left[\left(h_k(f_r(x)) - E[h_k(f_r(x))]\right)\alpha'_i(x)\right]\right. \\ &\qquad\qquad \left. + E_{x\leftarrow D}[h_k(f_r(x))]E_{x\leftarrow D}[\alpha'_i(x)]\right) \\ &\quad = c_{i,k}E_{x\leftarrow D}[h_k(f_r(x))]E_{x\leftarrow D}[\alpha'_i(x)]. \end{aligned}$$

Where the last line above follows since α'_i is orthogonal to $sp(f_r)$. Now, note that by our assumption that the marginal distribution of $f_i(x)$ are identical, the quantity above is independent of f_r.

We now consider the contributions from the $c_{i,j,k}\beta_{i,j,k}$ terms. Note that for any i, $\sum_{j,k} c_{i,j,k}\beta_{i,j,k}(x,y) = \sum_k h_k(y)\sqrt{\sum_j c_{i,j,k}^2}t_k(x)$, for some $t_k \in sp(f_i)$ with $||t|| = 1$. Taking the expectation of this with $y = f_r(x)$ for a single $i \neq r$, has magnitude

bounded above by $\frac{\sqrt{|T|}}{d^3}\sqrt{\sum_{j,k} c_{i,j,k}^2}$. For $i = r$, this expression is similarly bounded by $\sqrt{|T|}\sqrt{\sum_{j,k} c_{i,j,k}^2}$. Because $||g|| = 1$, from Lemma 3.2 there can be at most w indices $i \in [d]$ for which $\sum_{j,k} c_{i,j,k}^2 \geq \frac{2}{w}$. Given that $\sum_{j,k} c_{r,j,k}^2 \leq \frac{2}{w}$ a statistical query oracle can respond with a value that is independent of the choice of the true function f_r provided the tolerance of the oracle, τ, satisfies: $\tau \geq \sqrt{2|T|/w} + \frac{\sqrt{|T|}}{d^3}\sqrt{2d}$. Hence for r chosen uniformly from $[d]$, there is a constant $c < 1$ such that no algorithm when given fewer than $\frac{d}{w}$ will be able to return a function f such that with probability at least c, $\langle f, f_r \rangle > O(1/\sqrt{d})$.

4 Learning Arbitrary Functions

We now discuss learning arbitrary real functions. While this may be a hard question in general, we hope to gain some ground when the functions have additional properties. Below, we show that one can learn L-Lipshitz function in time $n^{O(L/\epsilon)^2}$, up to ϵ error.

4.1 Lipschitz Functions We start by noting that if f is L-Lipschitz then there is a polynomial time algorithm for learning f approximately, within an additive error ϵ. We accomplish this by computing the low-level Fourier coefficients of f (i.e., correlation of f with the Fourier basis). It turns out that one only needs to look at Fourier basis functions of degree at most $O(L/\epsilon)^2$ to approximate within ℓ_2 error ϵ. Hence we can compute all of them in time $n^{O(L/\epsilon)^2}$.

THEOREM 4.1. *Let f be a function over $[-1,1]^n$ that is square-integrable. If f is L-lipschitz, then f can be approximated to within additive ℓ_2 error $\epsilon\|f\|$ using Fourier basis functions of degree at most $O(L/\epsilon)^2$.*

Hence, f can be learned from random examples in time $n^{O(L/\epsilon)^2}$. Furthermore, this number of examples is optimal as there is a matching lower bound.

Proof. Suppose $\|f\| = 1$ and consider the decomposition of f into Fourier basis:

$$f(\vec{x}) = \sum_{m=(m_1,\ldots m_n)\in\mathbb{Z}^n} \hat{f}_m e^{j(m\vec{x})},$$

where j is imaginary and $m\vec{x}$ is a dot-product. The gradient of f with respect to variable x_i is:

$$\frac{\partial f}{\partial x_i} = \sum_m m_i \cdot \hat{f}_m e^{jm\vec{x}}$$

Note that the ℓ_2 norm of the gradient must be at most L. Integrating over all $\vec{x}$ gives us:

$$L^2 \geq \int \sum_i \left(\frac{\partial f}{\partial x_i}\right)^2 \mathrm{d}\vec{x} = \sum_i \sum_m \hat{f}_m^2 m_i^2 = \sum_m \hat{f}_m^2 \|m\|^2.$$

Hence, $\sum_{m:\|m\|>L/\epsilon} \hat{f}_m^2 \leq \epsilon^2$.

Now the algorithm for learning the function f will just learn each Fourier coefficient $\hat{f}_m$ for $\|m\| \leq L/\epsilon$ directly as the empirical estimate of $\hat{f}_m = \mathbb{E}[f(\vec{x}) \cdot e^{jm\vec{x}}]$.

Note that, in constrast, a boolean function over the unit cube $g : \{-1,+1\}^n \to \{-1,+1\}$ is very hard to approximate. Even for $\epsilon = 0.1$, one requires $\Omega(2^n)$ time and sample complexity, as there are $exp(\Omega(2^n))$ functions that are all at least ϵ-far (that is at ℓ_2-distance ϵ) from each other. For example if we take a class of $exp(O(2^n))$ randomly chosen boolean functions, they are all ϵ-far from each other. Whereas for real valued functions over the real cube, we get a much smaller family of functions that are ϵ-far – only $exp(n^{(L/\epsilon)^2})$. The above theorem leads to the following observation.

OBSERVATION 4.1. *One cannot interpolate (extend) a boolean function on the unit cube $\{-1,+1\}^n$ to obtain an L-Lipschitz functions over $[-1,1]^n$ (for a constant L), that preserve distances between any pair of functions upto constant factors. By "preserving distance" we mean that if two boolean functions differ on at least ϵ fraction of the inputs, then their interpolations should also be at least $\Omega(\epsilon)$-far in ℓ_2 distance.*

4.2 Thresholded Polynomials Another natural and important class of functions for learning are thresholded polynomials functions that are boolean functions given by $f(\vec{x}) = P(\vec{x}) > \theta$ where $P(\vec{x})$ is a polynomial. This seems to be hard class in the worst case. Indeed it is at least as hard as parity with noise. Perhaps it becomes easier under some kind of smoothed analysis there θ is chosen from a distribution.

OBSERVATION 4.2. *The parity with noise problem is reducible to the problem of learning thresholded polynomial functions.*

Proof. Suppose we are given a parity function $f(\vec{x}) = \prod_{i\in S} x_i$ for some set $S \subset [n]$ of size k, and we get noisy samples $\hat{f}(\vec{x}) = f(\vec{x}) \cdot (-1)^{b_\eta}$ where b_η is a random η-biased Bernoulli variable

Then consider the threshold polynomial $sign(\hat{f}(\vec{x}))$. It can also be seen as (has same distribution as) the noisy threshold polynomial $\text{sign}(\prod_{i\in S} \cdot (-1)^{b_{\eta'}})$ for some η'. Furthermore, the distribution of x (for the learning of thresholded polynomials) is immaterial as long as it is symmetric.

References

[ABW10] Benny Applebaum, Boaz Barak, and Avi Wigderson. Public-key cryptosystem from different assumptions. In *Proceedings of the Symposium on Theory of Computing (STOC)*, 2010.

[Ale03] Michael Alekhnovich. More on average case vs approximation complexity. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, 2003.

[BFJ+94] Avrim Blum, Merrick Furst, Jeffrey Jackson, Michael Kearns, Yishay Mansour, and Steven Rudich. Weakly learning dnf and characterizing statistical query learning using fourier analysis. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 253–262, 1994.

[BK97] Avrim Blum and Ravi Kannan. Learning an intersection of a constant number of halfspaces under a uniform distribution. *Journal of Computer and System Sciences*, 54(2):371–380, 1997.

[BKW03] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, 2003.

[CM06] Graham Cormode and S. Muthukrishnan. Combinatorial algorithms for compressed sensing. In *Ann. Conf. Information Sciences and Systems*, 2006.

[CRT06] E. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52:489 – 509, 2006.

[CT06] E. Candes and T. Tao. Near optimal signal recovery from random projections: Universal encoding strategies. *IEEE Transactions on Information Theory*, 2006.

[DLM+07] Ilias Diakonikolas, Homin K. Lee, Kevin Matulef, Krzysztof Onak, Ronitt Rubinfeld, Rocco A. Servedio, and Andrew Wan. Testing for concise representations. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 549–558, 2007.

[Don06] D. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289 – 1306, 2006.

[FGRW09] Vitaly Feldman, Venkatesan Guruswami, Prasad Raghavendra, and Yi Wu. Agnostic learning of monomials by halfspaces is hard. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 385–394, 2009.

[GLPS12] Anna C. Gilbert, Yi Li, Ely Porat, and Martin J. Strauss. Approximate sparse recovery: Optimizing time and measurements. *SIAM J. Comput.*, 41(2):436–453, 2012.

[GSTV06] Anna Gilbert, Martin Strauss, Joel Tropp, and Roman Vershynin. Algorithmic linear dimension reduction in the ℓ_1 norm for sparse vectors. In *Allerton*, 2006.

[GSTV07] Anna Gilbert, Martin Strauss, Joel Tropp, and Roman Vershynin. One sketch for all: fast algorithms for compressed sensing. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 237–246, 2007.

[HKM12] Prahladh Harsha, Adam Klivans, and Raghu Meka. An invariance principle for polytopes. *J. ACM*, 59(29), 2012. Previously in STOC'10.

[IR08] Piotr Indyk and Milan Ruzic. Near-optimal sparse recovery in the l1 norm. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 199–207, 2008.

[Kea98] Michael Kearns. Efficient noise-tolerant learning from statistical queries. *J. ACM*, 45(6):983–1006, 1998. Previously in STOC'93.

[KKMS05] Adam Kalai, Adam Klivans, Yishay Mansour, and Rocco Servedio. Agnostically learning halfspaces. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 11–20, 2005.

[KOS08] Adam R. Klivans, Ryan O'Donnell, and Rocco A. Servedio. Learning geometric concepts via gaussian surface area. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, 2008.

[KST09] Adam Tauman Kalai, Alex Samorodnitsky, and Shang-Hua Teng. Learning and smoothed analysis. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 395–404, 2009.

[LLS+13] Roi Livni, David Lehavi, Sagi Schein, Hila Nachliely, Shai Shalev-Shwartz, and Amir Globerson. Vanishing component analysis. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 597–605, 2013.

[Lon95] P. Long. On the sample complexity of pac learning halfspaces against the uniform distribution. *IEEE Transactions on Neural Networks*, 6(6):1556–1559, 1995.

[LSS13] Roi Livni, Shai Shalev-Shwartz, and Ohad Shamir. A provably efficient algorithm for training deep networks. *arXiv:1304.7045, available at* `http://arxiv.org/abs/1304.7045`, 2013.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *Proceedings of the Symposium on Theory of Computing (STOC)*, 2009.

[PS12] Ely Porat and Martin J. Strauss. Sublinear time, measurement-optimal, sparse recovery for all. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1215–1227, 2012.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 84–93, New York, NY, USA, 2005. ACM.

[SS02] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[Sze89] Gabor Szegö. *Orthogonal Polynomials*. A.M.S., 1989.

[Val84] Leslie Valiant. A theory of the learnable. *Communications of the ACM*, 27, 1984.

[Val12] Gregory Valiant. Finding correlations in subquadratic time, with applications to learning parities and juntas with noise. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, 2012.

[Vem97] S. Vempala. A random sampling based algorithm for learning the intersection of halfspaces. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 508–513, 1997.