Contents

1 Introduction

Learning polynomials has two areas of difficulty: 1) sample complexity and 2) computational complexity. This work will focus on analyzing algorithms for learning restricted classes of polynomials in order which achieve both good sample complexity and computational complexity which does not depend on p^d, where p is the dimension of the vectors in the space and d is the degree of the polynomial. We will proceed by first summarizing the work in the field in terms of their assumptions and their complexities in both the statistical sample regime as well as the computational complexity regime.

1.1 Motivation

Before we begin discussing the other background work, we motivate why we want to even learn polynomials. Recent work by Hazan et al. (2017) suggests that sparse polynomials are good function approximators for learning maps from *hyperparameters* in complicated machine learning models (for instance, neural nets) to eventual output.

Polynomials are also interesting in terms of discrete-time models, as there is a very natural interpretation. We can think of learning a d-degree polynomial model as a linear model over time chunks of with d. To make this more concrete, consider n-gram models in natural language processing (NLP) (here, we will take $n = d$: No relation to n, the number of data points). A common approach to featurizing sentences and documents is to build a sparse vector of counts lying in dimension $|\mathcal{V}|^d$, where $\mathcal{V}$ is the total vocabulary of the training and test corpuses. Each coordinate corresponds to a sequence of d words which appears in the corpus, known as a d-gram. The d-gram representation is simply the normalized counts of the d-grams in the document or sentence (whatever is being represented). If we concatenate the i-gram representations from $i = 1, \cdots, d$ and then learn a linear model on top of it, we are essentially learning a polynomial over the original space $|\mathcal{V}|$ where here, the monomials correspond to different i-grams.

2 A Greedy-ish Method for Learning Sparse Polynomials Exactly

2.1 Overview

Our work is directly inspired by the paper of Andoni et al. (2014), which manages to learn *sparse* polynomials (that is, polynomials with few coefficients) via a greedy algorithm approach in time which does not depend on p^d where p is the dimension and d is the degree of the polynomial. Here, they learn the polynomial exactly and do not invoke improper learning. Notably, we focus in this paper on learning polynomials over non-discrete domains (in particular, this leaves out the Boolean hypercube — in the noisy case, even learning monomials over random Rademacher variables is cryptographically hard). Therefore, for now, imagine that we are considering data generated by a standard Gaussian or a uniform distribution on a box in $\mathbb{R}^d$ with identity covariance, scaled appropriately. Notably, both of these distributions are *product distributions*, a fact whose importance will become apparent later.

The algorithm of Andoni et al. (2014) is called `Growing-Basis`. The authors assume a correlation oracle for $\langle f, f^* \rangle$ and $\langle f, (f^*)^2 \rangle$ where f^* is the polynomial we are trying to learn, and f is the input polynomial. The main idea is to consider polynomials in the orthogonal basis, and to use the trick of a) looking at the moments of the *squared* function and b) cleverly proceeding to eliminate high-degree terms of the polynomial first. Squaring the function allows us to retain interaction some terms in expectation (e.g., not everything in the expectation will be independent and become 0). By the orthogonality of the basis, we will be able to use correlation oracles in order to determine whether or not a particular monomial exists in the true function. If we find it exists, we keep it. If not, we remove the monomial from the set of monomials we consider.

There are essentially two settings of this algorithm. In one case, we assume we have access to the correlation oracles defined previously. In this setting, we require only $\mathcal{O}(kpd)$ calls to that oracle in order to learn the polynomial. In the other case, we assume that we see a sequence of (possibly noisy, possibly non-noisy) samples $(x, f^*(x))$. In this case, we need to *estimate* the oracle and we in fact have an associated sample complexity. For the sample-based algorithm, the `Growing-Basis` algorithm has sample complexity $\mathcal{O}(\text{poly}(p, k, 1/\epsilon, m))$, where $m = 2^d$ if the distribution is uniform over a box and $m = 2^{d \log d}$ if the distribution is a product Gaussian. The time complexity of the algorithm is $\mathcal{O}(pd \cdot \text{poly}(p, k, 1/\epsilon, m))$, which is just the number of samples times pd. In the noisy case, the sample complexity experiences an additional multiplicative factor of $\text{poly}(1 + \sigma)$, where the noise is assume to be from an independent Gaussian with variance σ^2.

However, the algorithm is not simple, and is not easy to implement. Furthermore, it seems very specific to the sparse, product-distribution setting. We would eventually like to be able to learn polynomials which are "low-dimensional" in other ways as well, or perhaps even learn other classes of functions. `Growing-Basis` does not lend itself to this kind of generalization easily. Thus, in this research project, we wish to discover and analyze more general and more simple algorithms which solve this problem, so that we may generalize the techniques found to other classes of problems.

2.2 Understanding `Growing-Basis`

The algorithm is given in Algorithm 1. In this section, we will give some deeper intuition for why this algorithm works and what its limitations are.

Algorithm 1 `Growing-Basis`

```
1: procedure Growing-Basis(degree $d$, $\langle \cdot, f^* \rangle$, $\langle \cdot, (f^*)^2 \rangle$)
2:     $\hat{f} := 0$                              ▷ The variable we will update until it equals $f^*$.
3:     while $\langle 1, (f^* - \hat{f})^2 \rangle > 0$ do     ▷ As long as correlation is non-zero, grow the basis.
4:         $H := 1, B := 1$ ▷ $H$ keeps track of square correlation, $B$ is the actual basis term.
5:         for $r = 1, \cdots, p$ do        ▷ Search in lexicographic order, going from large degree down to small degree.
6:             for $t = d, \cdots, 0$ do
7:                 if $\langle H \cdot H_{2t}(x_r), (f^* - \hat{f})^2 \rangle > 0$ then        ▷ The key correlation test.
8:                     $H := H \cdot H_{2t}(x_r)$, $B := B \cdot H_t(x_r)$
9:                     break out of double loop.
10:                end if
11:            end for
12:        end for
13:        $\hat{f} := \hat{f} + \langle B, f^* \rangle \cdot B$                ▷ Update the function.
14:    end while
15:    return $\hat{f}$
16: end procedure
```

We first explain the algorithm in the *oracle setting*. The key idea in the oracle setting is to greedily build a polynomial in an *orthonormal polyomial basis* which depends on the distribution, one basis function at a time. Because of the properties of the basis, we can identify first the existence of variable x_i using correlation, and then find its degree in the basis function. We always consider the leftover correlation, finding the next basis term and its coefficient in the polynomial until the amount of leftover correlation is zero.

This strategy will work for the following reasons:

- We can work in an orthonormal basis and pay a factor 2^d increase in the sparsity of the representation.
- An orthonormal polynomial basis for a product distribution decomposes into terms of different degress with *no overlapping variables*. That is, we can construct an orthonormal basis for a product distribution by taking a product of the 1-dimensional bases in each variable.
- We can identify the degree of a variable in a particular basis function by examining the correlation of several basis functions with $(f^* - \hat{f})^2$ in an iterative fashion. Note that $f^* - \hat{f}$ can be thought of in terms of "remaining correlation". This search procedure takes time $\mathcal{O}(nd)$.

To make the intuitions precise, let us give some definitions.

Definition 2.1. Inner product $\langle h_1, h_2\rangle$ is defined with respect to a distribution D over the data X as $\mathbb{E}_D\left[h_1(x)h_2(x)\right]$. We also have $\|h\|^2 = \langle h, h\rangle$.

Definition 2.2. A *correlation oracle pair* calculates $\langle f^*, f\rangle$ and $\langle (f^*)^2, f\rangle$ where f^* is the true polynomial.

Definition 2.3. Consider inner product space $\langle \cdot, \cdot\rangle_D$ for distribution D, where $D = \mu^{\otimes p}$ is a product measure over $\mathbb{R}^p$. For any coordinate, we can find an orthogonal basis of polynomials depending on distribution D by Gram-Schmidt. Let $H_t(x_i)$ be the degree t basis function for variable x_i. Then for $T = (t_1, \cdots, t_p)$ such that $\sum_i t_i = d$, $H_T(x) = \prod_i H_{t_i}(x_i)$ defines the orthogonal basis function parametrized by T in the product basis.

Therefore, we can write

$$f^*(x) := \sum_T \alpha_T H_T(x)$$

for any polynomial f^*.

2.2.1 Controlling Sparsity Blowup

We now describe how we control the blowup in monomial-sparsity that happens if we shift to a different (orthonormal) basis.

Lemma 2.4. *Suppose f^* is k-sparse in product basis H_1. Then it is ($k2^d$)-sparse in product basis H_2.*

Proof. Write each term $H^{(1)}_{t_i}(x_i)$ of f^* in basis H_1 in basis H_2: each will have t_i terms. Since each monomial term in H_1 is a product of such $H_{t_i}(x_i)$, there will be $\prod_i (t_i + 1) \leq 2^{\sum_i t_i} \leq 2^d$ terms for each monomial. Since there are k monomials, there are at most $k2^d$ terms when expressed in H_2. Said another way, that means that there are at most $k2^d$ terms in the sum $\sum_T \alpha_T H_T(x)$. □

2.2.2 Detecting Degrees with Correlation Tests in the Oracle Case

We will now see the importance of the properties of orthogonal polynomial bases over product distributions to `Growing-Basis`. There are two key properties which will allow us to design an algorithm to detect the presence of a variable in a basis term as well as find its degree, namely

- Orthogonal basis elements have zero correlation.
- Orthogonal basis elements can be factored into orthogonal basis terms in one variable.

We now give a lemma which suggests the correctness of the search procedure used in `Growing-Basis`.

Lemma 2.5. *Let d_1 denote the maximum degree of variable x_1 in f^*. Then, $\langle H_{2t}(x_1), (f^*)^2(x)\rangle > 0$ iff $t \leq d_1$.*

Proof. We have

$$(f^*)^2(x) = \sum_T \alpha_T^2 \prod_{i=1}^{p} H_{t_i}(x_i)^2 + \sum_{T \neq U} \alpha_T \alpha_U \prod_{i=1}^{p} H_{t_i}(x_i) H_{u_i}(x_i)$$

Note that if $t > t_1$, $H_{t_1}(x_1)^2$ will only be supported on basis functions $H_0, \cdots, H_{2t_1}$. This set does not include H_{2t} since $2t > 2t_1$, so $\langle H_{2t}(x_1), H_{t_1}(x_1)^2\rangle = 0$. Likewise for second term if $t > u_1$, thus, if $t > d_1$, correlation is zero. If $t = d_1$, the correlation is nonzero for the first term, but zero for the second term. Let's look at

$$\left\langle H_{2t}(x_1), \prod_{i=1}^{p} H_{t_i}(x_i)^2 \right\rangle = \left\langle H_{2t}(x_1), \prod_{i=1}^{p} \left(1 + \sum_{j=1}^{2t_i} c_{t,j} H_j(x_i)\right)\right\rangle$$

Since $t_i = t$ (for T such that $t_1 = d_1$), the coefficient of the term $H_{2t}(x_1)\prod_{i=2}^{p} H_0(x_i)$ is the only thing that remains since everything else will get zeroed out. Then just sum over T such that $t_1 = d_1$.

The second term does not contribute since either $i \neq 1$ or $t_i + u_i < 2t$ since $u_i \neq t_i$. Note that for the first case ($i \neq 1$), we are using the fact that x_1 will not show up in another orthgonal basis for a different variable. Then, we will get

$$\left\langle H_{2t}(x_1), \prod_{i=1}^{n} H_{t_i}(x_i) H_{u_i}(x_i) \right\rangle = 0$$

□

This lemma shows that if we proceed from the largest degree possible, we will be able to detect the degree of x_1 in one of the basis functions in the representation of f^*. With some more analysis of a similar flavor, we can extend this to finding a complete product basis representation. The key idea is to proceed through degree lists T in lexicographic order.

Definition 2.6. Lexicographic order.
We say $T \succeq U$ if T is lexicographically superior to U. For instance, $1544300 \succeq 1544000$ since $0 < 3$.

One can check that if $T = T_1 \cdots T_r 0 \cdots 0$ and there is no S such that H_S has a nonzero coefficient while $S_1 \cdots S_r \succeq T_1 \cdots T_r$, and there is some other U such that $U \succeq T_1 \cdots T_r 0 \cdots 0$, then we are done with the variables $x_1, \cdots, x_r$ and it is time to move on to the next variable, x_{r+1}. Essentially, we move from checking correlations of $H_{2t_1,\cdots,2t_{r-1},2t,0,\cdots,0}$ and $(f^*)^2$ to checking correlations of $H_{2t_1,\cdots,2t_{r-1},2t_r,2t,\cdots,0}$ and $(f^*)^2$.

The lemma proved above shows that this general procedure works to detect the correct H_T for the case $r = 1$. It is not so hard to generalize this to arbitrary r by induction – the analysis is almost the same. It thus follows that we can detect basis elements H_T in descending lexicographic order, where the lexicographic order is on the T. Thus, we find the maximal T first, followed by the second most maximal, and so on.

2.2.3 Learning from Samples Instead of Oracles

In the sampling case, we no longer receive a correlation oracle pair. Instead, we receive true samples, which may also be corrupted by noise. In this case, we must *simulate* the correlation oracle, using the following estimator:

Definition 2.7. Correlation oracle estimator.
We wish to estimate $C_H = \langle H, (f^*)^2 \rangle$. For n random samples, we estimate

$$\hat{C}_H = \frac{1}{n}\sum_{i=1}^{n} H(x_i)(f^*)(x_i)^2$$

By Chebyshev, one can get a constant probability estimate with $\mathcal{O}\left(\frac{\mathbb{E}\left[H^2(f^*)^4\right]}{\epsilon^2}\right)$ samples, where the top term is just the variance. To increase the probability of successfully estimating arbitrarily large to $1 - \delta$, just repeat the estimator $\log(1/\delta)$ times and take the median, a standard trick in approximation algorithms.

Since we have approximation error, an additional issue arises: when the correlation is truly 0, we may not be able to recognize it due to the approximation! Thus, in the sampling version of the algorithm, a cutoff is necessary below which values are declared 0.

The sample complexity therefore depends on $\max_{H,f^*} \mathbb{E}\left[H^2(f^*)^4\right]$, as we saw from the Chebyshev bound, which is dependent on the distribution. Here, H is a degree $2d$ basis function and f^* is a degree d, k-sparse polynomial, as we see from Algorithm 1. It turns out that the Legendre polynomials (the orthonormal polynomial basis which arises when we work over $D = [-1, 1]^n$ uniform) have bounded basis monomials: $|H_{d_i}(x_i)| \le \sqrt{2d_i + 1}$. Therefore,

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

and thus

$$|f^*(x)| = |\sum_S \alpha_S H_S(x)| \le 2^d \sum_S |\alpha_S| \le 2^d \sqrt{k}$$

by Parseval's identity, which is essentially the Pythagorean theorem for inner product spaces and the fact that there are only k monomials since f^* is k-sparse. Therefore,

$$\max_{H, f^*} \mathbb{E}\left[H^2 (f^*)^4\right] \le 2^{6d} k^2$$

It turns out that a similar fact holds for Hermite polynomials, which arise when D is a Gaussian product distribution. In this case, the constant depending on d will be $2^{d \log d}$ instead.

2.3 Why Does `Growing-Basis` Not Generalize to Non-Product Distributions?

The methodology in the algorithm and proof in the previous sections has the following properties:

- It relies heavily on orthogonal properties of polynomials.
- It is "term-by-term": we examine and find each basis function one at a time.
- There is a 2^d dependence in the sample complexity because
  - Transforming to an orthogonal basis only causes 2^d blow-up in sparsity
  - A fact about the structure of the coefficients of Legendre/Hermite polynomials.
- It has a key weakness, relying heavily upon the product distribution assumption in order to construct orthogonal polynomial bases over p variables.

What could we do to generalize to non-product distributions? First of all, it is possible to create orthonormal polynomial bases for *non-product distributions* by using Gram-Schmidt orthogonalization procedure. In fact, efficient methods for finding these representations are studied in the context of *polynomial chaos expansions* in the simulation of various differential equations in science Navarro et al. (2014). However, a key difficulty remains: There is no guarantee (and it is in fact extremely unlikely) that each of the basis functions contains only one variable. We saw in the proofs above that we needed to use this fact at one point. The fact that this is probably not true prevents us from simply proceeding in lexicographic order, detecting each variable and its degree one-by-one for each monomial basis term. Instead, we would somehow have to use facts about the particular orthogonal basis derived: Namely, which variables are in which term and so on, probably some properties about the factorization as well. We would then also have to invent some method of detecting clusters of monomials at a time.

This approach seems difficult but perhaps possible on a case-by-case basis. It might be that for some particular product distributions it is possible to find a nice form for the orthogonal basis by which means one might develop an analagous algorithm to `Growing-Basis`. Therefore, one might pursue this direction as one method of generalizing. However, it also seems brittle.

3 Other Related Work

There are several categories for papers in the literature related to learning polynomials efficiently. Roughly, they can be broken down into categories by the following dichotomies:

(a) exact or improper learning;

(b) statistical and/or computational speed-up;

(c) dimension reduction/sketching in p, the dimension of data, or n, the number of data points

(d) Boolean domain or not

We can also allow improper learning: In other words, we may not necessarily care about learning the exact function, but will be happy enough with a predictor which does about as well as the true function. Our goal is also to consider both statistical and computational efficiency, and hopefully find cases in which one can speed both up.

There are so far two aspects of our work: the actual sketching/dimension-reduction of the regression, and also the method of analysis.

We describe work related to sketching first.

Pham & Pagh (2013) describes a method for applying sketches of tensors to the polynomial regression problem. The idea is to use random dimension reduction (Johnson-Lindenstrauss, or fast Johnson-Lindenstrauss) and to learn on the dimension-reduced space. This allows the authors to provide an extra term in the bound which is due to the error of approximation due to the dimension reduction.

The work of Maillard & Munos (2009) and Maillard & Munos (2012) introduces the notion of *compressed least squares*: Essentially, they perform Johnson-Lindenstrauss linear dimension reduction on the data points, and then run regular linear regression. They analyze this setup and get an error bound. Kabán (2014) improves the analysis to generalize to a broader class than just JL linear dimension reduction, and specializes the bound.

However, recent work by Slawski (2017a) and Slawski (2017b) demonstrate that the conclusions reached in these earlier papers are not entirely accurate. In particular, in the generalization bounds from the prior work, there are terms which in some settings of the optimal linear regression coefficients implies a worse statistical bound than regular ordinary least squares regression. These papers connect compressed least squares to the analysis of *principal components regression* (PCR) and show that the JL approach, as well as the column subset selection approach to dimension reduction, which are both computationally cheaper than using PCA, require a few more samples than PCR does.

Another aspect of the literature is computationally focused, and focuses on sketching the kernel matrix of kernel ridge regression using a bounded number of queries to the kernel matrix. Our focus will likely not be in this direction during this project.

Finally, we mention that Avron et al. (2014) provides methods for applying **oblivious subspace embeddings** (OSE) to kernel learning, in particular, they discuss the case where the kernel is polynomial. In the past, the main issue was that OSEs only applied to situations where one had an explicitly provided matrix, which one does not have in the case that one implicits performs regression (via kernel trick). They also apply their techniques to principal component regression (PCR).

3.1 Bounds from Principal Components Regression and Johnson-Lindenstrauss

We will give more specifics for two methods in this domain, both in the realm of improper learning: That is, we will not try to learn the exact polynomial, only a good predictor. First, we will consider kernel polynomial regression as an efficient approach to regression in the high-dimensional space. We will then focus on the dimension reduction paradigm: Namely, perform a regression in a lower-dimensional space created by a dimension reduction procedure on the data. The hope is then that either a) the dimension reduction reduced the data into its "correct space" and we can procede with regression without losing anything, or b) the bias induced by the dimension reduction is small, and that we can tolerate it as an approximation error. It seems necessary to therefore make assumptions about the initial space so that it is in some sense "lying on a low-dimensional manifold". Indeed this seems to be the case.

3.1.1 Polynomial Kernel Regression

Mohri et al. (2012) gives basic bounds for linear and kernel regression. For kernel ridge regression, the sample complexity bound holds with probability $1-\delta$:

$$R(h) \le \hat{R}(h) + \frac{8r^2\Lambda^2}{\sqrt{m}}\left(\sqrt{\frac{\mathrm{Tr}[K]}{mr^2}} + \frac{3}{4}\sqrt{\frac{\log\frac{2}{\delta}}{2}}\right)$$

where K is the kernel matrix which satisifes $K(x,x) \le r$, the hypothesis h is a linear function of the kernel embedding, and the true function f is bounded: $|f(x)| \le r\Lambda$. This bound follows from a standard Rademacher complexity bound on the regression task. The application to kernel regression is achieved by bounding the data-dependent Rademacher complexity under the constraints imposed upon the kernel. This sample complexity bound is bad (essentially, $\mathcal{O}(p^d)$ unless strong assumptions are made upon the degredation of the spectrum of the kernel matrix: Eigenvalues need to become exponentially small.)

Regular kernel ridge regression takes time proportional to $\mathcal{O}(n^3)$ or $\mathcal{O}((p^d)^3)$, depending on which formulation of the kernel regression problem one uses (one can write it in terms of XX^T or X^TX). Recall n is the number of data points and p is the dimension while d is the degree of the polynomial in the kernel. This approach can be very expensive, and we see that unless the data matrix X has a quickly decaying spectrum, the actual number of samples to learn is large and depends on p^d – so this is very bad.

3.1.2 Dimension Reduction and Regression

In the dimension reduction setting, Slawski (2017a) gives the following theorem:

Theorem 3.1. *(Slawski (2017a)).*
For $r \in [\min(p, n)]$, *let dimension reduction (from* $\mathbb{R}^p \to \mathbb{R}^k$*) matrix* $R \in \mathbb{R}^{p\times k}$ *be* $(2nr, \epsilon_1/\sqrt{r}, \delta_1)$*-JL (this just means it satisfies JL with error parameter* $\epsilon_1/\sqrt{r}$ *with probability* $1-\delta_1$ *for* $2nr$ *points). Let* R *also be* $(r, \epsilon_2, \delta_2)$*-restricted isometry (the RIP property) from sparse recovery. Then with probability* $1-\delta_1-\delta_2$,

$$\|(I - P_{X_R})X\|_F^2 \leq \left(1 - \frac{\epsilon_1^2}{(1-\epsilon_2)^4}\right)\|\Delta_r\|_F^2$$

where P_{X_R} *denotes the projection onto the top* r *principal components of* X *and* Δ_r *is everything but the top* r *principal components of* X*. Conditioned on this event holding, we have*

$$\min_{v\in\mathbb{R}^k} \mathbb{E}\left[\|Xw^* - XRv\|_2^2/n\right] \leq \left(1 - \frac{\epsilon_1^2}{(1-\epsilon_2)^4}\right)\|w^*\|_2^2\frac{\|\Delta_r\|_F^2}{n} + \sigma^2\frac{k}{n}$$

where σ^2 *is the variance of the noise of the labels.*

This theorem suggests that if we want to apply dimension reduction to the p^d-dimensional space, the data matrix in p^d space (i.e., the original data matrix which has been tensored d times) must either

- Obey a strong decay on the spectrum (this is the Δ term), in which case principal components regression (PCR) is optimal, or
- Obey a restricted isometry property (RIP) like condition, which may be unlikely to hold for $X^{\otimes d}$.

Thus neither of these methods are satisfactory. It may still be interesting to look into characterizing the structure of $X^{\otimes d}$, however. There are several conditions in the compressed sensing literature which are weaker than RIP (for instance, nullspace property) which may hold on $X^{\otimes d}$ for some assumptions on X. As Andoni et al. (2014) noted, there is also a literature on making compressed sensing methods *computationally efficient* as well, via sketching methods, though these tend to depend on the structure of the design matrix. Nevertheless, it is a possible future direction.

4 Appendix: Considering the Single Monomial Case

In this section, we will outline some progress on the goal of estimating a *single* monomial. We will outline the sample complexity of the approach and mention the computational complexity. This section is a part of ongoing work with Rishabh Dudeja, Alex Andoni, and Daniel Hsu.

In order to create a more general algorithm, we can consider a simple case first: The case where the polynomial we are learning is a single monomial. If we develop some techniques in this simpler regime, we might be able to see a way to generalize the analysis beyond product distributions. Note that this problem is essentially an identification problem if the polynomial is multilinear: We want to identify which variables are in the monomial. If it is not multilinear, we also must learn the degrees.

Our learning task is to learn $f^*(x) = \prod_{i \in S} x_i^{d_i}$, for some set $S \subseteq [p]$ of the variables. We also have $d_i \in \mathbb{Z}^+$ and $\sum_{i=1}^{p} d_i = d$, since the degree of f^* is d. Let us begin by assuming there is no noise. We are also placing ourselves in the *random design* setting, where we assume something about the distribution of x (as in Andoni et al. (2014)).

4.1 Analyzing the Sample Complexity of ERM

First, we can consider the computationally inefficient empirical risk minimization (ERM) algorithm, which literally searches over all possible choices. In our case, since we consider monomials over $\mathbb{R}^p$ with integer degrees with total degree bounded by d, there are only a finite number of hypotheses in the hypothesis class $\mathcal{H}$. There are

$$|\mathcal{H}| = \sum_{i=1}^{\min(p,d)} \binom{p}{i} \cdot \#(\text{ partitions of length } i \text{ summing up to } d)$$

Assuming that $d << p$, we get that $\binom{p}{i} \leq \mathcal{O}(p^d)$. By a theorem of Hardy & Ramanujan (1918), we get that the partition number of d is bounded by $\mathcal{O}(\frac{1}{d}e^{\sqrt{d}})$. Thus, $\log |\mathcal{H}| \leq \mathcal{O}\left(d \log p + \log(1/d) + \sqrt{d}\right)$.

Then, by Mohri et al. (2012), we have the following theorem:

Theorem 4.1. *Let L be a loss function bounded by M. Assume that hypothesis class $\mathcal{H}$ is finite. Then for any $\delta > 0$ with probability at least $1 - \delta$, for all $h \in \mathcal{H}$, we have*

$$R(h) - \hat{R}(h) \leq M\sqrt{\frac{\log |\mathcal{H}| + \log(1/\delta)}{2n}}$$

where n is the number of samples, R is the risk function (taken over the expectation of the data) and $\hat{R}$ is the empirical risk function (taken over the sample dataset).

Thus, we only need to bound M to be done. For data x on a finite support, this depends on the chosen loss function (possibly ℓ_2 loss). The bound is then $M \leq L(\max_x f^*(x) - \min_x f^*(x))$. For other assumptions we can get different bounds.

Sample complexity wise, this bound seems like a good result, assuming L is reasonable. However, the algorithm is completely intractable, and we want both good sample complexity *and* good computational complexity.

4.2 A Log-Space Transformation

ERM is not computationally efficient. In light of that, suppose we consider the transformation $(x, f^*(x)) \to (\log |x|, \log |f^*(x)|)$. Then, we would have

$$\log f^*(x) = \sum_{i \in S} d_i \log(|x_i|)$$

and we would would now have a linear problem in p-dimensional space. In particular, this problem can be solved efficiently with ℓ_1 minimization techniques like the Lasso. Note that after we take log transforms, we are considering the relaxed case of linear hypothesis classes with $\|w\|_1 \leq d$. Our relaxation is essentially allowing the powers of the variables in the monomial to be fractional.

There is a key issue to watch out for with this method. By taking the log transform, we convert the space that x is supported on to something which is unbounded, if the support of x contains 0. It may have been that the support of x was already unbounded in the first place: For instance, if $x \sim \mathcal{N}(0, I_p)$, $x \in (-\infty, \infty)^p$. However, in this case, the probability of x being at the extremes goes to zero quickly. When we take the log-transform, we mess up the locations at which the value of x diverges. So, we need to find a way to deal with this.

Another issue is that by transforming the data, after solving the problem in the transformed space, we would need some approach to revert back to the original space. In the case that the polynomial had all even degrees, there would be no issue with respect to the $|\cdot|$. There would also be no issues in the multilinear case. In other scenarios however, this detail would present an issue. However, we would still need to control the error in the linear case after reverting to the original space by taking exponents: Namely, we would get a multiplicative error e^ϵ, where ϵ was the error incurred in the transformed space, since $\log f^*(x) = \log \hat{f}(x) + \epsilon$ implies

$$f^*(x) = e^\epsilon \hat{f}(x)$$

Noise is also difficult to consider in this setting. Supposing we had multiplicative noise $1 + \eta$ at the beginning, we would get a noise term of $\log(1 + \eta)$, which will not have zero mean. Thus, there is an inherent bias which presents a difficulty. For now we ignore noise.

4.2.1 A Truncated Log

We can present a partial fix for the issue of the log transformation by considering a "truncated" log term.

Definition 4.2. Truncated Log.
Let $\tilde{\log}_\lambda(x) := \begin{cases} \log(x) & \text{if } x \geq \lambda \\ \log(\lambda) & \text{if } x < \lambda \end{cases}$.

This prevents log from going to infinity, but also amounts to being a threshold at which we throw out data. We will also need to figure out how to undo this process, or how to view this procedure as a kind of rejection sampling.

4.3 A Rademacher Analysis of ℓ_1-Minimization

We now present the beginning of a Rademacher analysis of the sample complexity of this approach, and outline the issues which still need to be resolved. First, we have the following theorem bounding the sample complexity of a regression problem with respect to the Rademacher complexity of the hypothesis class, from Mohri et al. (2012):

Theorem 4.3. *Assume that $\|f - f^*\|_\infty \leq M$ for all $h \in \mathcal{H}$. Then for any $\delta > 0$ with probability at least $1-\delta$ over a sample S of size n, the following holds for all $h \in \mathcal{H}$:*

$$\mathbb{E}\left[(f(x) - f^*(x))^2\right] \leq \frac{1}{n}\sum_{i=1}^{n}(f(x_i) - f^*(x_i))^2 + 4M\mathcal{R}_n(\mathcal{H}) + M^2\sqrt{\frac{\log(1/\delta)}{2n}}$$

Then, a theorem from Kakade et al. (2009) bounds the empirical Rademacher complexity of a linear function class with ℓ_1-norm bounded by d. Call this function class $\mathcal{G}$.

Theorem 4.4.

$$\hat{\mathcal{R}}_n(\mathcal{G}) \leq \frac{d * \left(\sup_{i\in[n]} \|x_i\|_\infty\right)\sqrt{2\log(2p)}}{\sqrt{n}}$$

where $\hat{\mathcal{R}}_n$ is the empirical Rademacher complexity for a dataset of n points.

To get the non-empirical Rademacher complexity used in the bound, we have to take an expectation over the data distribution:

$$\mathcal{R}_n(\mathcal{G}) = \mathbb{E}_{x_k \sim D^n_{k=1}}\left[\hat{\mathcal{R}}_n(\mathcal{G})\right]$$

Essentially, this means that we must bound

$$\mathbb{E}_{x_k \sim D^n_{k=1}}\left[\sup_{i\in[n]} \|x_i\|_\infty\right]$$

If we bound this quantity with respect to some distribution D (which may perhaps be non-product), we will be able to get a non-trivial bound on the Rademacher complexity. We are in the process of developing approximations to this bound first for $\mathcal{N}(0, \sigma^2 I)$, where we can recognize that this becomes a problem of finding

$$\mathbb{E}_{\{y_k \sim \mathcal{N}(0,\sigma^2)\}^{np}_{k=1}}\left[\sup_{k\in[np]} |\tilde{\log}(y_k)|\right]$$

We can calculate this term by way of analogy to calculating the expected value of the maximum of the absolute value of a normal variable, and can handle the truncated log term by splitting up the expectation into two integrals.

We can then plug this bound into the formula from Mohri et al. (2012) to get a sample complexity bound. On to this sample complexity bound we will need to determine how many samples from the original distribution were rejected, and add that upon to the bound. We also will still need to figure out how to return from the transformed space to the original space.

References

Alexandr Andoni, Rina Panigrahy, Gregory Valiant, and Li Zhang. Learning sparse polynomial functions. *SODA '14 Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 500–510, 2014. URL `http://www.mit.edu/~andoni/papers/learnpoly.pdf`.

Haim Avron, Huy L. Nguyen, and David P. Woodruff. Subspace embeddings for the polynomial kernel. *NIPS '14 Proceedings of the 27th International Conference on Neural Information Processing Systems*, 2:2258–2266, 2014. URL `http://www.cs.cmu.edu/afs/cs/user/dwoodruf/www/anw14.pdf`.

G. H. Hardy and S. Ramanujan. Asymptotic formulae in combinatory analysis. *Proceedings of the London Mathematical Society*, 17:75–115, 1918. URL `http://ramanujan.sirinudi.org/Volumes/published/ram36.pdf`.

Elad Hazan, Adam Klivans, and Yang Yuan. Hyperparameter optimization: A spectral approach. *arXiv preprint*, 2017. URL `https://arxiv.org/pdf/1706.00764.pdf`.

Ata Kabán. New bounds on compressive linear least squares regression. *AISTATS 2014 The 17th International Conference on Artificial Intelligence and Statistics, Journal of Machine Learning Research–Proceedings Track*, 33:448–456, 2014. URL `http://proceedings.mlr.press/v33/kaban14.pdf`.

Sham M. Kakade, Karthik Sridharan, and Ambuj Tewari. On the complexity of linear prediction: Risk bounds, margin bounds, and regularization. *Advances in Neural Information Processing Systems*, pp. 793–800, 2009. URL `http://ttic.uchicago.edu/~karthik/rad-paper.pdf`.

Odalric-Ambrym Maillard and Rémi Munos. Compressed least-squares regression. *NIPS 2009 Advances in Neural Information Processing Systems 22*, 2009. URL `https://papers.nips.cc/paper/3698-compressed-least-squares-regression`.

Odalric-Ambrym Maillard and Rémi Munos. Linear regression with random projections. *Journal of Machine Learning Research*, 13:2735–2772, 2012. URL `http://www.jmlr.org/papers/volume13/maillard12a/maillard12a.pdf`.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*, chapter 10: Regression, pp. 237–266. The MIT Press, 2012.

Maria Navarro, Jeroen Witteveen, and Joke Blom. Polynomial chaos expansion for general multivariate distributions with correlated variables. 2014. URL `https://arxiv.org/pdf/1406.5483.pdf`. arXiv preprint.

Ninh Pham and Rasmus Pagh. Fast and scalable polynomial kernels via explicit feature maps. *KDD '13 Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 239–247, 2013. URL `http://www.itu.dk/people/pagh/papers/tensorsketch.pdf`.

Martin Slawski. Compressed least squares regression revisited. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, Proceedings of Machine Learning Research*, 54:1207–1215, 2017a. URL `http://proceedings.mlr.press/v54/slawski17a.html`.

Martin Slawski. On principal components regression, random projections, and column subsampling. *arXiv preprint*, 2017b. URL `https://arxiv.org/pdf/1709.08104.pdf`.