# Towards Better Understanding of Algorithms and Complexity of Some Learning Problems

Xin Yang

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2020

Reading Committee:

Paul Beame, Chair

Kevin Jamieson

Sewoong Oh

Program Authorized to Offer Degree:
Computer Science and Engineering

University of Washington

**Abstract**

Towards Better Understanding of
Algorithms and Complexity of Some Learning Problems

Xin Yang

Chair of the Supervisory Committee:
Professor Paul Beame
Computer Science and Engineering

We present several novel results on computational problems related to supervised learning. We focus on the computational resources required by algorithms to solve learning problems. The computational resources we consider are running time, memory usage and query complexity, which is the number of positions in the input that the algorithm needs to check. Some contributions include:

- Time-space tradeoff lower bounds for problems of learning from uniformly random labelled examples. With our methods we can obtain bounds for learning concept classes of finite functions from random evaluations even when the sample space of random inputs can be significantly smaller than the concept class of functions and the function values can be from an arbitrary finite set.

- A simple and efficient algorithm for approximating the John Ellipsoid of a symmetric polytope. Our algorithm is near optimal in the sense that our time complexity matches the current best verification algorithm. Experimental results suggest that our algorithm significantly outperforms existing algorithms.

- The first algorithm for the total least squares problem, a variant of the ordinary least

squares problem, that runs in time proportional to the sparsity of the input. The core to developing our algorithm involves recent advances in randomized linear algebra.

- A generic space efficient algorithm that is based on deterministic decision trees.
- The first algorithm for the linear bandits problem with prior constraints.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude to my advisors Paul Beame and Kevin Jamieson for all the precious advice, encouragement and support over the years. I would like to thank Paul for his patient guidance and tireless efforts. I hope that Paul's enthusiasm, creativity and views rub off on my work and life. I would like to thank Kevin for his optimism for research and incredible energy along the way, which greatly inspires me to dive into research areas that are new to me.

I would also like to thank Sewoong Oh for serving on my reading committee, and Yin Tat Lee and Mehran Mesbahi for serving on my general dissertation committee.

This dissertation is based on joint works with my other amazing collaborators, including Michael B Cohen, Ben Cousins, Huaian Diao, Rebecca Hoberg, Yin Tat Lee, Yingyu Liang, Vincent Liu, Lalit Jain, Harishchandra Ramadas, Thomas Rothvoss, Zhao Song, Mengdi Wang, David Woodruff, Lin F Yang, Qiao Zhang and Danyang Zhuo. I have learned a lot from you, and this dissertation would not exist without these collaborations. I would like to specifically thank Zhao for his close collaboration and guidance.

I was privileged to become a member of UW theory group. Our great faculty, especially Anna Karlin, James R. Lee and Anup Rao, have taught me a lot on brilliant research goals. I also enjoyed the great days with my friends: Alireza, Cyrus, Dorna, Ewin, Farzam, Haotian, Jeffrey, Jennifer, Kira, Kuikui, Makrand, Mert, Nathan, Oscar, Robbie, Ruoqi, Sally, Sami, Siddharth, Siva, Swati, Victor, Vincent and Yuqing. I would like to thank my fellow graduate students in the larger CSE community. You made my grad life much more enjoyable.

Thank you to Elise deGoede Dorough and Lindsay Michimoto. Studying abroad is

never an easy task, and it is your help that made me go through this process.

I am grateful to Chong Wang, my mentor at ByteDance. I was fortunate to work with you to see how industrial research works.

Going back to my undergraduate years at Tsinghua University, I would like to thank Periklis Papakonstantinou for welcoming me into the world of theoretical computer science. I would also like to thank Weihao Gao, Chengtao Li, Fu Li, Yuanzhi Li, Tianren Liu, Junxing Wang and Yi Wu for being amazing friends. It is so nice that we still keep in touch.

Finally, my family is perhaps the main source of my inspiration. I would like to thank my father, Qingwen, for all the guidance over the years. I would also like to thank my mother, Wenru, for always being there when I needed you. I cannot say enough how thankful I am for you.

# DEDICATION

To my parents, Qingwen and Wenru.

# Chapter 1

# INTRODUCTION

Machine learning is the field that studies how computer algorithms can perform complicated tasks via learning. Instead of pre-programmed rules, we feed the learning algorithms with training data, and expect them to make predictions and decisions. The more training data the learning algorithms receive, the more potential for accurate predictions and high quality decisions they have. This mimics how humans learn. Taking learning language as an example, babies are not born to know how to speak: It is the language environment of newborns, rather than genes or ethnicity, that determines which language they speak; babies learn to speak by receiving visual and sound information from the environment. Things are similar in machine learning: For example, in the task of email spam detection, while traditional algorithms may design many hard-coded complicated rules, machine learning algorithms take spam emails and non-spam emails as input, and train a classifier that can make good predictions on general unseen emails.

Learning systems, especially those built on large-scale data, have made significant impact in real-life applications. A far from exhaustive list includes autonomous vehicles, recommendation systems, face recognition and machine translation, which have already been involved in the daily lives of billions of people. On the other hand, it takes vast resources to train machine learning models. The computation power needed to train learning models increased 300,000 times from 2012 to 2018 [118]. Google's BERT-large [44], the state-of-art model across many natural language processing (NLP) tasks, was trained on 64 TPU chips for 4 days on 3 billion word-pieces. As a consequence, the carbon emissions of machine learning is becoming a problem: In a 2019 study [136], training a NLP model can cost 626,155 pounds of $CO_2$, which is about 17 times of the lifetime $CO_2$ consumption

of average American person.

This brings in the motivation of this thesis: we need a better understanding of the computation resources needed for machine learning. To do that, we use the techniques of theoretical computer science. From an algorithmic perspective, we would like to design more efficient algorithms for different learning tasks. From a hardness perspective, we would like to identify the computational complexity of learning problems in different settings in order to understand the bottleneck of various learning tasks which enables us to better allocate computational resources. There are multiple orthogonal computation resources, like float point operations and memory. We need to understand how the usage of learning algorithms on these resources correlates with each other.

This thesis mainly focuses on *supervised learning*. Let us start with an example of human learning of teaching concepts to children: When children see some new objects like cars or cats, we tell children names of these objects, and after gaining enough experience, children become able to recognize cars and cats, even if these cars or cats have not been seen by children. In this example, the names of concepts serve as "labels" of the visual information. Supervised learning is analogous to this procedure: Recall that in the aforementioned email spam detection problem, for the training data, besides the contents of the emails, the learner also sees whether each email is spam or not. We can think that there is an underlying ground truth mapping (unknown to the learner) that given contents of email as input, outputs whether the email is spam or not, and the goal of machine learning algorithms is to recover such a model based on training data. The spam information actually labels the training data, and roughly speaking, supervised learning is the setting where the learner receives data with labels. With labelled data, supervised learning is a comparatively simpler setting, and contains many widely used machine learning tasks in practice such as recommendation systems, object recognition and machine translation.

Supervised learning is also widely used in other research areas like materials science and medical science, where researchers want to find the best material/medicine from a pool of candidates via costly experiments. The problem of optimal experimental design

is that of finding the best way to conduct experiments with fewest trials as possible. We can regard the performances of each experiment as labels, so this problem falls into the category of supervised learning.

Overall, we resolve several learning problems in supervised learning, including time-space tradeoffs for online learning, optimal experimental design and regression problems.

## *1.1 Our Contributions*

More specifically, the main contributions of this thesis are summarized as follows.

### *1.1.1 Time-Space Tradeoffs in Online Supervised Learning*

We study the time-space tradeoffs in an online supervised learning setting. Traditionally, researchers care more about the training time and the size of training data required for good performance, while little is known about the space usage. However, as modern machine learning relies increasingly on the large scale of the data, it becomes unaffordable to put all training data in memory all at once. How much information from the training data needs to be kept in memory in order to learn successfully becomes an important question. To overcome the storage constraint, inspired by advances in streaming algorithms, an *online* learning scheme is used in practice. This means that the learning algorithms access the training data consecutively from a data stream, which makes it possible not to store training data all at once. Hence one may hope for a good learning algorithm that is both efficient in time measured by size of training data, and space measured by storage usage. In the breakthrough work of Ran Raz [113], it is shown that in this setting, even for a very simple task, any successful learning algorithm requires either essentially putting all the needed training data into memory, or seeing exponential number of training samples. Raz considered the problem of learning a boolean vector by seeing its inner products with other boolean vectors. We prove that this type of time-space tradeoffs holds for a more general class of learning problems. We give a property of learning problems that we call

*norm amplification curve* so that if one learning problem has a certain norm amplification curve, then the strong time-space tradeoff holds. Moreover, our results allow labels to be more than binary by introducing complex numbers into the analysis. This work is in Chapter 3 and appears in the following publication.

> Paul Beame, Shayan Oveis Gharan, and Xin Yang. Time-space tradeoffs for learning finite functions from random evaluations, with applications to polynomials. In *Conference On Learning Theory*, pages 843–856, 2018 [11].

Independently and contemporaneously with our preliminary version ([9]), Garg, Raz, and Tal [56] proved closely related results to ours.

The specific learning task Raz considered is over boolean variables, and the learning task is about inner product of boolean vectors, which can be considered as evaluation of linear boolean functions. A natural generalization is learning low degree polynomials from random evaluations. This is a very general class of learning problems, because when variables take discrete finite values, they can be modeled by finite fields, and all their behaviors can be modeled by polynomials over finite fields. In order to prove time-space tradeoff for learning polynomials, we need to bound the corresponding norm amplification curve, which is where the problem of estimating the weight distribution of Reed-Muller codes arises. Reed-Muller codes are among the oldest error correcting codes and are highly related to polynomials over finite fields. We resolve this problem by showing that the bias of Reed-Muller codes follows an exponential tail bound. These contributions are in Chapter 4 and appear in the following publication.

> Paul Beame, Shayan Oveis Gharan, and Xin Yang. On the bias of reed–muller codes over odd prime fields. *SIAM Journal on Discrete Mathematics*, 34(2):1232–1247, 2020 [8].

### 1.1.2 Decision Tree Based Generic Query Algorithm

Recently, Gonen, Lovett, and Moshkovitz [59] conjectured a criteria to determine whether a problem can be efficiently learned with bounded space usage. Their result is built on a reduction from the *statistical query model*. For more details on the statistical query model, see Section 2.1.1. We find a generic learning algorithm based on reduction from decision trees, which refutes Gonen et al.'s original conjecture. After communication from us, Gonen et al. proposed a modified conjecture in the conference version of their paper [60]. This is a joint work with Paul Beame and is in Chapter 5.

### 1.1.3 Faster Algorithm for Total Least Squares

*Least squares regression* is a method for fitting linear models. It was introduced by Carl Friedrich Gauss in the 18th century, and is among one of the oldest machine learning algorithms. In this setting, we have $n$ data points $(A_1, b_1), \cdots, (A_n, b_n)$ where $A_i$ is a $d$ dimensional vector, and $b_i$ is a real number. We want to use a linear model to fit these data, that is, we want to find a $d$ dimensional vector $x$ so that $b_i \approx \langle A_i, x \rangle$. Quantitatively, we want to choose $x$ to minimize the square loss $\sum_i (b_i - \langle A_i, x \rangle)^2$ (hence the name "least squares"). As one of the simplest models, linear models are widely used in many areas including statistics and finance, which in turns makes least squares regression a well-studied method.

One interpretation of least squares is that the labels $\{b_i\}$ are noisy, and we want to make the minimal shift on these labels so that the data fits into a linear model. A natural generalization is that not only the labels, but also the data $\{A_i\}$ can be noisy, so we can make changes to both $\{A_i\}$ and $\{b_i\}$ so that they fit into a linear model. This is called the *total least squares* problem. Traditional algebra solution for the total least squares problem require $O(nd^2)$ time. Such complexity is not efficient enough with large scale data, where sub-linear running time algorithms are preferred. Based on advances in randomized linear algebra, we develop a novel algorithm for the total least squares problem that achieves

running time proportional to the number of non-zero entries in the input. This is a huge advantage when the input is given in a sparse format. Large sparse data is quite common in applied machine learning, since information is not perfectly gained and there are many missing entries. These contributions are in Chapter 6 and appear in the following publication.

> Huaian Diao, Zhao Song, David Woodruff, and Xin Yang. Total least squares regression in input sparsity time. In *Advances in Neural Information Processing Systems*, pages 2482–2493, 2019 [48].

### *1.1.4 Efficient Algorithm for Optimal Experimental Design*

Previously we talked about optimal experimental design. One simple case is optimal experimental design for linear models. Linear models are the simplest models for experimental design, where the performance of the experiment depends linearly in each attribute. In this setting, the candidate pool is all $d$ dimensional vectors. Each possible experiment is also represented by a $d$ dimensional vector. The performance of an experiment $a$ at the candidate $x$ is given by their inner product $\langle a, x \rangle$. We want to design the experiments so that as much as possible information can be obtained.

There are multiple optimal criteria for the linear optimal experimental design, and *D-optimal* is one of the widely used optimality criteria. Roughly speaking, D-optimal design maximizes the determinant of some "information matrix" of the chosen experiments, and by doing so, the variance for the estimation is minimized. It turns out that computing the D-optimal design is equivalent to a computational geometry problem: the John Ellipsoid of convex polytopes. A classic result of Fritz John [72] stated that any convex polytope can be well sandwiched by two ellipsoids. These two ellipsoids share the same center and one is a scaled version of the other. The computational task is then to compute the maximal volume ellipsoid inscribed in the polytope. This problem also has a wealth of different applications, including sampling and integration [148, 28], linear bandits [23, 68], linear

programming [93], cutting plane methods [80] and differential privacy [108].

We develop a simple fixed point iteration for this problem, which achieves the state-of-art time complexity. We give experiments showing that our algorithm outperforms all previous algorithms, which matches the theoretical calculation. This work is in Chapter 7 and appears in the following publication.

> Michael B Cohen, Ben Cousins, Yin Tat Lee, and Xin Yang. A near-optimal algorithm for approximating the john ellipsoid. In *COLT*, 2019 [33].

### *1.1.5 Experimental Design with Constraints*

We study the constrained optimal experimental design and apply it to the *linear bandits* problem. The linear bandits problem is a well studied online learning problem, and arises naturally in many applications such as content recommendation. This work generalizes the results of Fiez, Jain, Jamieson and Ratlif [54] in the unconstrained case. In the problem of linear bandits, there are examples where specific constraints can help reduce the sample complexity, but very little is known with general constraints. We obtained the first regret bound that takes advantage of the geometry properties of the constraints. This is a joint work with Lalit Jain and Kevin Jamieson, and is in Chapter 8.

# Chapter 2
# BACKGROUND, PRELIMINARY AND NOTATIONS

In this chapter we cover some topics and notations for the rest of this thesis.

## *2.1 Supervised Learning*

Supervised learning is a learning task where the learner wants to learn a function from labelled data. Formally, let $\mathcal{X}$ be a space of samples, and $\mathcal{C}$ be a space of concepts, where each concept is a mapping from $\mathcal{X}$ to some range $\mathcal{Y}$. Usually samples from $\mathcal{X}$ are represented by vectors over some field. Each observation is a pair $(x, y)$ where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. With training data $(x_1, y_1), (x_2, y_2), \cdots$, the learner produces an inferred concept $\widehat{c} \in \mathcal{C}$.

Many learning tasks fall into the category of supervised learning. For example, in linear regression, the concept class is all linear functions, and the labelled samples are noisy evaluations from the ground truth; in digit recognition using the MNIST dataset [92], the concept class is all mappings from $28 \times 28$ images to $\{0, 1, \cdots, 9\}$, and the labelled samples are digit drawings with correct labels.

### *2.1.1 Learning Models*

Supervised learning can be found in various computation models. In this section we introduce some common learning models for supervised learning.

#### *PAC Learning*

PAC learning, or Probably Approximately Correct learning, introduced by Leslie Valiant [146], enables us to apply computational complexity theory concepts in machine

learning. In the setting of PAC learning, we have a distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$. We stress that this distribution $\mathcal{D}$ is unknown to the learner. We also have a loss function $\ell$ mapping from $\mathcal{Y} \times \mathcal{Y}$ to $\mathbb{R}_{\geq 0}$. The input of the learning problem is a batch of $T$ samples $\{(x_i, y_i)\}_{i=1}^{T}$, where each sample $(x_i, y_i)$ is chosen independently identically from $\mathcal{D}$. The goal of the learner, is to report $\widehat{c} \in \mathcal{C}$ that has good performance. The performance of a concept $c \in \mathcal{C}$ is measured by $\mathbf{E}_{(x,y)\sim\mathcal{D}}[\ell(c(x), y)]$, which is called *risk*, or generalization error.

Let $c_*$ be the optimal concept, namely the one for which $\mathbf{E}_{(x,y)\sim\mathcal{D}}[\ell(c_*(x), y)]$ is minimized. Let $\epsilon \geq 0$ be the "Approximately" parameter and $\delta \in (0, 1)$ be the "Probably" parameter. PAC learning is interested in finding $\widehat{c} \in \mathcal{C}$ so that with probability at least $1 - \delta$,

$$\mathop{\mathbf{E}}_{(x,y)\sim\mathcal{D}}[\ell(\widehat{c}(x), y)] \leq \mathop{\mathbf{E}}_{(x,y)\sim\mathcal{D}}[\ell(c_*(x), y)] + \epsilon.$$

Here the randomness is taken over both the choice of the training set and the (potential) internal randomness of the learning algorithm.

Some readers may be more familiar with a variant definition of PAC learning, where instead of a joint distribution over $\mathcal{X} \times \mathcal{Y}$, there is a distribution $\mathcal{D}_{\mathcal{X}}$ over $\mathcal{X}$, which is known to the learner. Then, the training data is generated by $y = c'(x)$, where $c'$ is a one-to-many function from $\mathcal{X}$ to $\mathcal{Y}$ and $c'$ is unknown to the learner. $c'$ may have internal randomness, for example it may contain noise. If all possible $c'$ must come from $\mathcal{C}$, then this learning procedure is called *proper*.

One major goal of PAC learning is to identify the learnability of learning problems. The size of training data $T$, which is sometimes called *sample complexity*, is one of the most important aspects of PAC learning algorithm. For fixed parameters $\delta$ and $\epsilon$, we want to design learning algorithms with sample complexity as small as possible. To make learning algorithms applicable in practice, their computational complexity is another research focus.

In this thesis, we focus on using *empirical risk minimization* in PAC learning. At a high level, empirical risk minimization uses the concept that has the best performance over the

training data as its inference for the ground truth. Indeed, for each candidate concept $\widehat{c}$, although we cannot compute the risk of $\widehat{c}$ because the underlying distribution $\mathcal{D}$ is unknown, we can still measure its performance over the training data by $\sum_{i=1} \ell(\widehat{c}(x_i), y_i)$, which is called the *empirical loss*. Empirical risk minimization is the approach of reporting the concept with the smallest empirical loss. In some cases, perfect recovery is possible, namely the learner can return $\widehat{c}$ that is consistent over all observed samples; in other cases, due to noise or low complexity of the concept classes, the learner may not be able to find a consistent concept, but the learner can still return some concept that minimizes the empirical loss. Many theoretical results, including VC dimension and chaining methods, focus on the conditions under which empirical risk minimization can give a good estimator.

*Statistical Query Learning*

The statistical query model, introduced by Michael Kearns [77], is a generalization of the PAC learning model, aiming at developing learning algorithms robust to noise. In the statistical query model, we focus with the case of $\mathcal{Y} = \{-1, 1\}$. There is an unknown ground-truth $c \in \mathcal{C}$ where $c$ is chosen according to some distribution (usually uniform). There is a distribution $\mathcal{D}_{\mathcal{X}}$ over $\mathcal{X}$, which is known to the learner. The label of the data $x \in \mathcal{X}$ is then generated as $y = c(x)$. Unlike in the PAC learning model, the learner does not have direct access to labelled data $(x, y)$, e.g, sampling from $\mathcal{X}$ according to $\mathcal{D}_{\mathcal{X}}$; instead, there is an oracle $\mathsf{SQ}$. $\mathsf{SQ}$ takes a query $(\psi, \tau)$ where $\psi : \mathcal{X} \to \{-1, 1\}$ is some function over $\mathcal{X}$ and $\tau > 0$ is the tolerance parameter, and returns a value $\mathsf{SQ}(\psi, \tau)$ satisfying

$$|\mathsf{SQ}(\psi, \tau) - \mathop{\mathbf{E}}_{x \sim \mathcal{D}_{\mathcal{X}}}[\psi(x)c(x)]| \leq \tau.$$

It is relatively easy to prove lower bounds in the statistical query model, and the complexity measure in the statistical query model is closely related to other important quantities in learning theory including VC-dimension and sign-rank [121].

*Online Learning*

Online learning is another widely studied learning model. Online learning takes place in consecutive rounds, where in $t$-th round, there is a loss function $f_t : \mathcal{X} \to \mathbb{R}$. The learner selects a sample $x_t$ from $\mathcal{X}$ and suffers from a loss of $f_t(x_t)$. Then the learner receives some information about $f_t$, which can either be the whole function $f_t$, the gradient of $f_t$ at $x_t$, or just the value of the loss $f_t(x_t)$. The performance of the learner is measured by total loss, which is measured by $\sum_{t=1}^{T} f_t(x_t)$. One major goal of online learning is to design good strategy for the learner, so that the *regret* against optimal fixed choice, formally expressed as

$$\sum_{t=1}^{T} f_t(x_t) - \min_{x \in \mathcal{X}} \sum_{t=1}^{T} f_t(x),$$

is minimized. Another goal is to minimize the sample complexity $T$ so that the learner can make good choice after $T$ rounds.

If we regard $\{f_t\}$ as concepts and the $\{f_t(x_t)\}$ as labels, then online learning falls into the category of supervised learning. Unlike PAC learning, online learning does not have to be distributional. On one hand, the learning algorithm has the freedom to choose the sample that it would like to gain information on; on the other hand, we do not have to make assumptions on the distribution of the loss functions: they can be *stochastic*, i. e., i.i.d in each round; they can be *oblivious adversarial*, that is, they can be correlated, but not depend on the choice of the learner; they can also be *totally adaptively adversarial*, or in some sense, the worst case scenario where some adversary picks up the worst loss function for the learner based on previous history.

Let us see a concrete example of online learning. Suppose that we are a seller with many products and we want to sell something to one customer. In each round, we can recommend one product to the customer, and obtain the dissatisfaction index of the customer for this product, measured by a real number. Our target is to identify the product that the customer is most satisfied in as few rounds as possible. This problem can be modeled as a *multi-armed bandits* problem, which is a standard online learning problem.

In this problem, there is only one loss function, which is the dissatisfaction index of the customer; and the learner can only observe $f(x)$ for the chosen $x$, but cannot see other values.

In this thesis we also study a distributional version of online learning. In this setting, learning still takes place in rounds, but now the learner cannot choose the sample $x$. Instead, there is a distribution $\mathcal{D}_{\mathcal{X}}$ over $\mathcal{X}$ that is known to the learner, and in each round the learner receives i.i.d samples $(x, y)$ where $x \sim \mathcal{D}_{\mathcal{X}}$. We can also view this model as a streaming version of PAC learning, where the training data is not present to the learner all at once, but accessed as a stream of data. This is of great practical significance, because modern machine learning is largely built on big data, where the amount of training data is so large that it cannot be fully stored in the memory during the training process. Hence a natural work-around, is to read the training data from a data stream. From the perspective of streaming algorithms, this requires us to consider the following performance measurements: the sample complexity, the storage usage, and the number of passes we need to go through the data stream.

### *2.1.2 Loss Functions*

Here we give some concrete examples of loss functions.

- When $\mathcal{Y}$ is discrete, such as in the case of classification, $\ell$ can be the 0-1 loss. Namely, $\ell(y_1, y_2) = 1$ if $y_1 = y_2$, and $\ell(y_1, y_2) = 0$ otherwise.

- In regression problems, norm based loss functions are widely used. For example, the ordinary least squares regression uses $\ell_2$ loss $\ell(y_1, y_2) = (y_1 - y_2)^2$; ridge regression uses $\ell_1$ loss $\ell(y_1, y_2) = |y_1 - y_2|$.

The choice of loss functions is very important. In some cases, loss functions determine the computational complexity of the learning problem. For example, for the problem of binary classification, where $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{C}$ is all linear classifiers of the form $\mathbb{1}_{\langle x,w\rangle \geq 0}$, it is

known that with 0-1 loss, this problem is **NP**-hard. On the other hand, binary classification becomes tractable if we work with convex, continuous loss functions. Moreover, loss functions can affect the shape of the optimal solution. In the linear regression problem, it is well known that solution equipped with $\ell_1$ loss tends to be more sparse. This is the famous LASSO algorithm.

### *2.2 Algebra over Finite Fields*

Sometimes we study learning problems over finite fields. We say a set $\mathbb{F}$ is a finite field, if $\mathbb{F}$ satisfies the following conditions:

- $\mathbb{F}$ is finite.
- $\mathbb{F}$ is an abelian group under the operation $+$ with the addition identity $0$.
- $\mathbb{F}_*$, which is the set of all non-zero elements in $\mathbb{F}$, is an abelian group under the operation $\times$.
- $\times$ distributes over $+$.

We say that a finite field $\mathbb{F}$ has order $q$, if the number of elements in $\mathbb{F}$ is $q$. In order to emphasize the order of the field, we denote $\mathbb{F}_q$ the finite field of order $q$.

It is well known that if $q$ is the order of some finite field, then $q$ has to be a power of some prime number. Moreover, if $q$ is prime, then $\mathbb{F}_q$ as an addition group has to be cyclic. So for prime $q$, $\mathbb{F}_q$ is just $\{0, 1, \cdots, q-1\}$.

### *2.3 Gaussian Processes*

The Gaussian distribution, or normal distribution, is the most studied continuous probability distribution for real-valued random variables. For scalar random variables, we use

$\mathcal{N}(\mu, \sigma^2)$ to denote the Gaussian distribution with mean $\mu$ and variance $\sigma^2$. That is, if $X$ is distributed according to $\mathcal{N}(\mu, \sigma^2)$, then the probability density function of $X$ is

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

Similarly, for vector-valued random variables, we use $\mathcal{N}(\mu, \Sigma)$ to denote the multi-variate Gaussian distribution with mean $\mu$ and covariance matrix $\Sigma$.

### *2.4 Least squares and pseudoinverse of matrices*

In this thesis we sometimes need to deal with best solution to a overdetermined linear system, that is, given a matrix $A \in \mathbb{R}^{n \times d}$ and a vector $y \in \mathbb{R}^n$ where $n \geq d$, we want to find $x \in \mathbb{R}^d$ so that $Ax = y$. This is not always possible since $n \geq d$. Therefore, we instead consider the optimal $x$ that minimizes $\|Ax - y\|_2^2 = \sum_{i=1}^n (\langle A_i, x \rangle - y_i)^2$. This is the aforementioned least squares problem in Section 1.1.3, and it has a closed form solution. Indeed, by setting the derivative of $\|Ax - y\|_2^2$ to be 0, we have

$$x = (A^\top A)^{-1} A^\top y.$$

The matrix $(A^\top A)^{-1} A^\top \in \mathbb{R}^{d \times d}$ is the famous *Moore-Penrose pseudoinverse* of $A$.

# Chapter 3

# TIME-SPACE TRADEOFFS FOR LEARNING FINITE FUNCTIONS FROM RANDOM EVALUATIONS

## *3.1 Introduction*

In supervised learning from labelled examples, the question of the sample complexity required to obtain good generalization has been of considerable interest and research. However, another important parameter is how much information from these samples needs to be kept in memory in order to learn successfully. There has been a line of work improving the memory efficiency of learning algorithms, and the question of the limits of such improvement has begun to be tackled relatively recently. Shamir [119] and Steinhardt, Valiant, and Wager [134] both obtained constraints on the space required for certain learning problems and in the latter paper, the authors asked whether one could obtain strong tradeoffs for learning from random samples that yields a superlinear threshold for the space required for efficient learning. Raz [113] showed that even given exact information, if the space of a learning algorithm is bounded by a sufficiently small quadratic function of the number of input bits, then the problem of online of learning parity functions given exact answers on random samples requires an exponential number of samples even to learn these parity functions approximately.

More precisely, we consider problems of online learning from uniform random samples, in which an unknown concept $c$ is chosen uniformly from a set $\mathcal{C}$ of (multivalued) concepts and a learner is given a stream of samples $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots$ where each $x^{(t)}$ is chosen uniformly at random from $\mathcal{X}$ and $y^{(t)} = L(x^{(t)}, c)$ for labelling function $L$ which maps each pair $(x, c)$ to the outcome (or label) of the value of concept $c \in \mathcal{C}$ when given $x \in \mathcal{X}$. The learner's goal is either that of identification "find $c$" or pre-

diction "predict $L(x, c)$ for randomly chosen $x$ with significant advantage over random guessing." In the case of learning parities, $\mathcal{C} = \mathcal{X} = \{0,1\}^n$ and $L(x, c) = x \cdot c \pmod 2$. With high probability $n + 1$ uniformly random samples suffice to span $\{0,1\}^n$ and one can learn parities using Gaussian elimination with $(n+1)^2$ space. Alternatively, an algorithm with only $O(n)$ space can wait for a specific basis of vectors $x$ to appear (for example the standard basis) and store the resulting values; however, this takes $\Omega(2^n)$ time. Raz [113] showed that either $\Omega(n^2)$ space or $2^{\Omega(n)}$ time is essential: even if the space is bounded by $n^2/25$, $2^{\Omega(n)}$ queries are required to learn $c$ correctly with any probability that is $2^{-o(n)}$. In follow-on work, Kol, Raz, and Tal [84] showed that the same lower bound applies even if the input $c$ is sparse.

We can view $c$ as a linear function over $\mathbb{F}_2$, and, from this perspective, parity learning identifies a linear function from evaluations over uniformly random inputs. A natural generalization asks if a similar lower bound exists when we learn higher degree polynomials with bounded space. As a motivating example, consider homogeneous quadratic functions over $\mathbb{F}_2$. Let $m = \binom{n+1}{2}$ and $\mathcal{C} = \{0,1\}^m$, which we identify with the space of homogeneous quadratic polynomials in $\mathbb{F}_2[z_1, \ldots, z_n]$ or, equivalently, the space of upper triangular Boolean matrices. This learning algorithm has $\mathcal{X} = \{0,1\}^n$ and $\mathcal{C} = \{0,1\}^m$, and the learning function $L(x, c) = c(x) = \sum_{i \leqslant j} c_{ij} x_i x_j \bmod 2$, or equivalently $L(x, c) = x^\top c x$ when $c$ is viewed as an upper triangular matrix.

Given $x \in \{0,1\}^n$ and $c \in \{0,1\}^m$, we can view evaluating $c(x)$ as computing $xx^\top \cdot c \bmod 2$ where we interpret $xx^\top$ as an element of $\{0,1\}^m$. For $O(m)$ randomly chosen $x \in \{0,1\}^n$, the vectors $xx^\top$ almost surely span $\{0,1\}^m$ and hence we can store $m$ samples of the form $(x, y)$ and apply Gaussian elimination to determine $c$. This time, we only need $n + 1$ bits to store each sample for a total space bound of $O(nm)$. An alternative algorithm using $O(m)$ space and time $2^{O(n)}$ would be to look for a specific basis such as the basis consisting of the upper triangular parts of $\{e_i e_i^\top \mid 1 \leqslant i \leqslant n\} \cup \{(e_i + e_j)(e_i + e_j)^\top \mid 1 \leqslant$

$i < j \leqslant n\}$. Previous lower bounds for learning do not apply to this problem[1] because $|\mathcal{X}| \ll |\mathcal{C}|$. Our results imply that this tradeoff between $\Omega(nm)$ space or $2^{\Omega(n)}$ time is inherently required to learn $c$ with probability $2^{-o(n)}$ or predict its output with at least $2^{-o(n)}$ advantage.

The techniques in [113] and [84] were based on fairly ad-hoc simulations of the original space-bounded learning algorithm by a restricted form of linear branching program for which one can measure progress at learning $c$ using the dimension of the consistent subspace. More recent papers, by Moshkovitz and Moshkovitz [102] using graph mixing properties and by Raz [114] using an analytic approach, considered more general tests and used a measure of progress based on 2-norms. While the method of [102] was not strong enough to reproduce the bound in [113] for the case of parity learning, the methods of Raz [114] and the later improvement [103] by Moshkovitz and Moshkovitz of [102] reproduced the parity learning bound and more.

In particular, Raz [114] defined a $\pm 1$ matrix $M$ that is indexed by $\mathcal{X} \times \mathcal{C}$. It is natural to see $M(x, c)$ as $(-1)^{L(x,c)}$ for a labelling function $L$ that has labels in $\{0, 1\}$. The lower bound is governed by the (expectation) matrix norm of $M$, which is a function of the largest singular value of $M$, and the progress is analyzed by bounding the impact of applying the matrix to probability distributions with small expectation 2-norm. This method works if $|\mathcal{X}| \geqslant |\mathcal{C}|$ - i. e., the sample space of inputs is at least as large as the concept class - but it fails completely if $|\mathcal{X}| \ll |\mathcal{C}|$, which is precisely the situation for learning quadratic functions. Indeed, none of the prior approaches works in this case.

In our work we extend the analytic approach to capture *general* discrete problems of learning from uniform random samples in which (1) the sample space of inputs can be much smaller than the concept class and (2) members of the concept class can have values from an arbitrary finite set, which we identify with $\{0, 1, \ldots, r\}$ for convenience. Our extensions come from two different directions.

---

[1]Note that in [84] the lower bound applies in a dual case when the unknown $c$ is sparse, and hence $|\mathcal{C}| \ll |\mathcal{X}|$.

We define a property of matrices $M$ that allows us to refine the notion of the largest singular value and extend the method of Raz [114] to the cases that $|\mathcal{X}| \ll |\mathcal{C}|$. This property, which we call the *norm amplification curve* of the matrix on the positive orthant, analyzes more precisely how $\|M \cdot p\|_2$ grows as a function of $\|p\|_2$ for probability vectors $p$ on $\mathcal{C}$. The key reason that this is not simply governed by the singular values is that the interior of the positive orthant can contain at most one singular vector. We give a simple condition on the 2-norm amplification curve of $M$ that is sufficient to ensure that there is a time-space tradeoff showing that any learning algorithm for $M$ with success probability at least $2^{-\varepsilon n}$ for some $\varepsilon > 0$ either requires space $\Omega(mn)$ or time $2^{\Omega(n)}$.

For any fixed learning problem given by a matrix $M$, the natural way to express the amplification curve at any particular value of $\|p\|_2$ yields an optimization problem given by a quadratic program with constraints on $\|p\|_2^2$, $\|p\|_1$ and $p \geqslant 0$, and with objective function $\|Mp\|_2^2 = \langle M^\top M, pp^\top \rangle$ that seems difficult to solve. Instead, we relax the quadratic program to a semi-definite program where we replace $pp^\top$ by a positive semidefinite matrix $U$ with the analogous constraints. We can then obtain an upper bound on the amplification curve by moving to the SDP dual and evaluating the dual objective at a particular Laplacian determined by the properties of $M^\top M$.

In order to handle concepts that are more than binary-valued[2], we move to matrices whose entries are complex $r$-th roots of unity. Indeed, a single matrix $M$ does not suffice for $r > 3$ and we instead work with a family of complex matrices $M^{(1)}, \ldots, M^{(r-1)}$, each associated with a different root of unity. We use the natural generalization of the norm amplification curve to complex matrices and also generalize the semi-definite relaxation method to bound these curves using $(M^{(j)})^* M^{(j)}$ instead of $M^\top M$. We then show how the overall analytic approach can be carried through with a modest number of changes from the binary-valued case.

[2]The formalization of Moshkovitz and Moshovitz [102, 103] does include the case of multivalued outcomes, though they do not apply it to any examples and their mixing condition does not hold in the case of small input sample spaces

Our lower bound shows that if the 2-norm amplification curve for $M$ has (or, in the case of $r$-valued labels, matrices $M^{(1)}, \ldots, M^{(r-1)}$ have) the required property, then to achieve learning success probability for $M$ of at least $|\mathcal{X}|^{-\varepsilon}$ for some $\varepsilon > 0$, either space $\Omega(\log |\mathcal{X}| \cdot \log_r |\mathcal{C}|)$ or time $|\mathcal{X}|^{\Omega(1)}$ is required. This matches the natural upper bounds asymptotically over a wide range of learning problems.

As applications, we focus on problems of learning polynomials of varying degrees over finite fields. For matrices $M$ associated with polynomials over $\mathbb{F}_2$, the property of the matrices $M^\top M$ required to bound the amplication curves for $M$ correspond precisely to properties of the weight distribution of Reed-Muller codes over $\mathbb{F}_2$. In the case of quadratic polynomials, this weight distribution is known exactly. In the case of higher degree polynomials, bounds on the weight distribution of such codes, or more precisely on the bounds on the bias of random degree $d$ polynomials over $\mathbb{F}_2$ of Ben-Eliezer, Hod, and Lovett [15] are sufficient to let us show that learning polynomials of degree at most $d$ over $\mathbb{F}_2^n$ from random inputs with probability $2^{-\Omega(n/d)}$ either requires space $\Omega(nm/d)$ or time $2^{\Omega(n/d)}$.

We also analyze learning problems for the case of prime fields $\mathbb{F}_p$ for $p$ odd using our multivalued techniques involving complex matrices. For $\mathbb{F}_p$, even the cases of linear and affine polynomials are new. We relate the norm amplification curves of the associated matrices to bounds on the bias of random degree $d$ polynomials over $\mathbb{F}_p$. We also give a precise analysis of the bias of the set of quadratic polynomials over $\mathbb{F}_p^n$ to derive tight time-space tradeoff lower bounds for learning them. For larger degrees we apply bounds on the bias that we proved in Chapter 4.

Independent of the specific applications to learning from random examples that we obtain, the measures of matrices that we introduce, the 2-norm amplification curve on the positive orthant, and semi-definite relaxation approach seem likely to have significant applications in other contexts outside of learning.

**Related work:** Independently and contemporaneously with our preliminary version ([9]), Garg, Raz, and Tal [57] proved closely related results to ours for the case of binary labels. The fundamental techniques are similarly grounded in the approach of [114]. At the very high-level, they prove very similar structural properties of the matrix $M$, namely, they show that it is an "$L_2$ two-source extractor" which can be seen to be equivalent to bounding our norm amplification curve for learning matrices. More precisely, their "almost orthogonality property" essentially corresponds to upper bounding $W_\kappa(M^*M)$ for some threshold $\kappa$ (see Definition 3.6.1 and Lemma 3.6.2). However, since we use duality explicitly, our proof seems more amenable to extensions, particularly, when we have more structure in the learning matrix $M$. Subsequently ([56]), they were also able to allow multivalued labels by extending the extractor approach to permit correlations between the sample inputs and the concept.

### *3.2 Branching programs for learning*

In order to be able to solve the learning problem given concept class $\mathcal{C}$, sample space of inputs $\mathcal{X}$ and labelling function $L$ on $\mathcal{X} \times \mathcal{C}$ exactly we require that the learning function $L$ have the property that for all $c \neq c' \in \mathcal{C}$ there exists an $x \in \mathcal{X}$ such that $L(x, c) \neq L(x, c')$. Note that the set $\{0, 1, \ldots, r-1\}$ of labels allows us to model any learning situation in which $r$ different labels are possible.

Following [113], the time and space of a learner are modelled simultaneously by expressing the learner's computation as a layered branching program: a finite rooted directed acyclic multigraph with every non-sink node having outdegree $r|\mathcal{X}|$, with one outedge for each $(x, y)$ with $x \in \mathcal{X}$ and $b \in \{0, 1, \ldots, r-1\}$ that leads to a node in the next layer. Each sink node $v$ is labelled by some $c' \in \mathcal{C}$ which is the learner's guess of the value of the concept $c$. (In the case of prediction we allow the sink label to be an arbitrary function from $\mathcal{X}$ to $\{0, 1, \ldots, r-1\}$ denoting the best prediction of the algorithm for each $x \in \mathcal{X}$.)

The space $S$ used by the learning branching program is the $\log_2$ of the maximum num-

ber of nodes in any layer and the time $T$ is the length of the longest path from the root to a sink.

The samples given to the learner $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots$ based on uniformly randomly chosen $x^{(1)}, x^{(2)}, \ldots \in \mathcal{X}$ and a concept $c \in \mathcal{C}$ determines a (randomly chosen) *computation* path in the branching program. When we consider computation paths we include the concept $c$ in their description. The (expected) success probability of the learner is the probability for a uniformly random concept $c \in \mathcal{C}$ that a random computation path given concept $c$ reaches a sink node $v$ with label $c' = c$ (or with sufficiently good predictions on randomly chosen $x \in \mathcal{X}$).

**Progress towards identification** Following [102] and [114] we measure progress towards identifying $c \in \mathcal{C}$ using the "expectation 2-norm" over the uniform distribution: For any set $S$, and $f : S \to \mathbb{R}$, define $\|f\|_2 = \left(\mathbb{E}_{s \in_R S} f^2(s)\right)^{1/2} = \left(\sum_{s \in S} f^2(s)/|S|\right)^{1/2}$. Define $\Delta_{\mathcal{C}}$ to be the space of probability distributions on $\mathcal{C}$. Consider the two extremes for the expectation 2-norm of elements of $\Delta_{\mathcal{C}}$: If $\mathbb{P}$ is the uniform distribution on $\mathcal{C}$, then $\|\mathbb{P}\|_2 = |\mathcal{C}|^{-1}$. This distribution represents the learner's knowledge of the concept $c$ at the start of the branching program. On the other hand if $\mathbb{P}$ is point distribution on any $c'$, then $\|\mathbb{P}\|_2 = |\mathcal{C}|^{-1/2}$.

For each node $v$ in the branching program, there is an induced probability distribution on $\mathcal{C}$, $\mathbb{P}'_{c|v}$ which represents the distribution on $\mathcal{C}$ conditioned on the fact that the computation path passes through $v$. It represents the learner's knowledge of $c$ at the time that the computation path has reached $v$. Intuitively, the learner has made significant progress towards identifying the concept $c$ if $\|\mathbb{P}'_{c|v}\|_2$ is much larger than $|\mathcal{C}|^{-1}$, say $\|\mathbb{P}'_{c|v}\|_2 \geqslant |\mathcal{C}|^{\delta/2} \cdot |\mathcal{C}|^{-1} = |\mathcal{C}|^{-(1-\delta/2)}$.

The general idea will be to argue that for any fixed node $v$ in the branching program that is at a layer $t$ that is $|\mathcal{X}|^{o(1)}$, the probability over a randomly chosen computation path that $v$ is the first node on the path for which the learner has made significant progress is $|\mathcal{C}|^{-\Omega(\log_r |\mathcal{X}|)}$. Since by assumption of correctness the learner makes significant progress

with at least $|\mathcal{C}|^{-\varepsilon}$ probability, there must be at least $|\mathcal{C}|^{\Omega(\log_r |\mathcal{X}|)}$ such nodes and hence the space must be $\Omega(\log |\mathcal{C}| \log_r |\mathcal{X}|)$.

Given that we want to consider the first vertex on a computation path at which significant progress has been made, it is natural to truncate a computation path at $v$ if significant progress has been already been made at $v$ (and then one should not count any path through $v$ towards the progress at some subsequent node $w$). Following [114], for technical reasons we will also truncate the computation path in other circumstances: if the concept $c$ has too high probability at $v$, or if the next edge is labelled by a pair $(x, y)$ for which the value on input $x$ of random concepts whose computation path reaches $v$ is significantly biased away from the uniform distribution on $\{0, 1 \ldots, r-1\}$.

Like Raz [114], we use an analytic approach to understanding the progress and the bias. In [114], only binary feedback is possible and progress is analyzed in terms of the matrix properties of a learning matrix $M$ given by $M(x, c) = (-1)^{L(x,c)}$, which is viewed as the learning problem specification. This form is particularly convenient since it allows one to represent the predictability of outcomes under a distribution $\mathbb{P}$ on $\mathcal{C}$ in terms of a matrix vector product. That is, $(M \cdot \mathbb{P})(x) = \mathbf{E}_{c \sim \mathbb{P}}[(-1)^{L(x,c)}]$ is the expected bias of a concept distributed according to $\mathbb{P}$ on input $a$.

It would be natural to try to extend this analytic approach for $r > 2$ by replacing $(-1)^{L(x,c)}$ by $\omega^{L(x,c)}$ where $\omega = e^{2\pi i/r}$ is a primitive $r$-th root of unity. However, for $r > 3$, simply having $\mathbf{E}_{c \in_R \mathcal{C}}[\omega^{f(c)}]$ small does not imply that $f$ is close to uniformly distributed on $\{0, 1, \ldots, r-1\}$. Nonetheless, by setting $g_k = \mathbf{Pr}_{c \in_R \mathcal{C}}[f(c) = k]$ we can apply the following proposition, which is a standard method using exponential sums, to show that bounding $|\mathbf{E}_{c \in_R \mathcal{C}}[\omega^{j \cdot f(c)}]|$ for all $j \in \{1, \ldots, r-1\}$ is sufficient to show that $f$ is close to uniformly distributed. We include its proof for completeness.

**Proposition 3.2.1.** *Suppose that* $\sum_{k=0}^{r-1} g_k = 1$ *and define* $g(z) = \sum_{k=0}^{r-1} g_k z^k$. *If* $|g(\omega^j)| < \varepsilon$ *for all* $j \in \{1, \ldots, r-1\}$ *then for all* $k \in \{0, 1, \ldots, r-1\}$, $|g_k - \frac{1}{r}| \leqslant \varepsilon$.

*Proof.* Write $s(z) = \sum_{k=0}^{r-1} z^k$ and observe that $s(1) = r$ but $s(\omega^j) = 0$ for all $j \in \{1, \ldots, r-$

1}. Define $h(z) = g(z) - \frac{1}{r} \cdot s(z)$. Observe that $h(z) = \sum_{k=0}^{r-1} h_k z^k$ where $h_k = g_k - \frac{1}{r}$ so it suffices to prove that $|h_k| \leqslant \varepsilon$ for all $k \in \{0, 1, \ldots, r\}$. Note that $h(1) = 0$, and $h(\omega^j) = g(\omega^j)$ for all $j \in \{1, \ldots, r-1\}$. Therefore $|h(\omega^j)| \leqslant \varepsilon$ for all $j \in \{0, 1 \ldots, r-1\}$. If we let $\mathbf{h} = (h_0, \ldots, h_{r-1})^T$ be the vector of coefficients and $\mathbf{v} = (h(1), h(\omega), \ldots, h(\omega^{r-1}))^T$ be the vector of values of $h$, we have $V(\omega) \cdot \mathbf{h} = \mathbf{v}$ where $V(\omega)$ is the usual $r \times r$ Vandermonde matrix for the powers of $\omega$. Now $V(\omega)^{-1} = V(\omega^{-1})/r$, the matrix of the discrete Fourier transform; so in particular, for every $k \in \{0, 1, \ldots, r-1\}$,

$$h_k = \frac{1}{r} \cdot \sum_{j=0}^{r-1} \omega^{-jk} v_j.$$

Hence $|h_k| \leqslant \frac{1}{r} \cdot \sum_{j=0}^{r-1} |v_j| \leqslant \varepsilon$ since every $|v_j| \leqslant \varepsilon$, which suffices to prove the proposition. □

Therefore, instead of the single $\pm$ matrix $M$ given by $M(x, c) = (-1)^{L(x,c)}$, we will analyze the learning problem given by $L$ using $r - 1$ different[3] complex matrices $M^{(j)} \in \mathbb{C}^{\mathcal{X} \times \mathcal{C}}$ for $j \in \{1, \ldots, r-1\}$ given by $M^{(j)}(x, c) = \omega^{j \cdot L(x,c)}$. We now define the probability distributions and truncation process for computation paths inductively as follows:

**Definition 3.2.2.** *We define probability distributions $\mathbb{P}_{c|v} \in \Delta_{\mathcal{C}}$ and the $(\delta, \alpha, \gamma)$-truncation of the computation paths inductively as follows:*

- *If $v$ is the root, then $\mathbb{P}_{c|v}$ is the uniform distribution on $\mathcal{C}$.*
- *(Significant Progress) If $\|\mathbb{P}_{c|v}\|_2 \geqslant |\mathcal{C}|^{-(1-\delta/2)}$ then truncate all computation paths at $v$. We call vertex $v$* significant *in this case.*
- *(High Probability) Truncate the computation paths at $v$ for all concepts $c'$ for which $\mathbb{P}_{c|v}(c') \geqslant |\mathcal{C}|^{-\alpha}$. Let* $\mathrm{High}(v)$ *be the set of such concepts.*

[3] In Proposition 3.2.1 one can observe that $|g(\omega^j)| = |g(\omega^{r-j})|$ so $\lceil (r-1)/2 \rceil$ matrices suffice, but we find it convenient to argue using all $r - 1$ matrices; however, this does imply that a single matrix suffices when $r = 3$.

- *(High Bias) Truncate any computation path at $v$ if it follows an outedge $e$ of $v$ with label $(x, y)$ for which $|(M^{(j)} \cdot \mathbb{P}_{c|v})(x)| \geqslant |\mathcal{X}|^{-\gamma}$ for some $j \in \{1, \ldots, r-1\}$. That is, we truncate the paths at $v$ if the label outcome for the next sample for $x \in \mathcal{X}$ is too predictable given the knowledge that the path was not truncated previously and arrived at $v$.*
- *If $v$ is not the root then define $\mathbb{P}_{c|v}$ to be the conditional probability distribution on $c$ over all computation paths that have not previously been truncated and arrive at $v$.*

*For an edge $e = (v, w)$ of the branching program, we also define a probability distribution $\mathbb{P}_{c|e} \in \Delta_{\mathcal{C}}$, which is the conditional probability distribution on $\mathcal{C}$ induced by the truncated computation paths that pass through edge $e$.*

With this definition, it is no longer immediate from the assumption of correctness that the truncated path reaches a significant node with at least $|\mathcal{X}|^{-\varepsilon}$ probability. However, we will see that a single assumption about the matrices $M^{(j)}$ will be sufficient to prove both that this holds and that the probability is $|\mathcal{C}|^{-\log_r |\mathcal{X}|}$ that the path reaches any specific node $v$ at which significant progress has been made.

### *3.3 Norm amplification by matrices on the positive orthant*

By definition, for $\mathbb{P} \in \Delta_{\mathcal{C}}$, and $M \in \mathbb{C}^{\mathcal{X} \times \mathcal{C}}$, $\|M \cdot \mathbb{P}\|_2^2 = \mathbf{E}_{x \in_R \mathcal{X}}[|(M \cdot \mathbb{P})(x)|^2]$. Observe that for $\mathbb{P} = \mathbb{P}_{c|v}$ and $M = M^{(j)}$ for $j \in \{1, \ldots, r-1\}$, the values $|(M^{(j)} \cdot \mathbb{P}_{c|v})(x)|$ are the quantities that we test to determine whether an edge labelled $a$ is a high bias edge that causes the truncation of the computation path. Therefore $\|M^{(j)} \cdot \mathbb{P}_{c|v}\|_2^2$ is the expected square of this bias value for uniformly random inputs at $v$.

If we have not learned the concept $x$, we would not expect to be able to predict its value on a random input; moreover, since any path that would follow a high bias input is truncated, it is essential to argue that $\|M^{(j)} \cdot \mathbb{P}_{c|v}\|_2$ remains small at any node $v$ where there has not been significant progress.

In [114] there is a single $\pm 1$ matrix $M$ and $\|M \cdot \mathbb{P}_{c|v}\|_2$ is bounded using the matrix

norm $\|M\|_2$ given by $\|M\|_2 = \sup_{\substack{f:\mathcal{C}\to\mathbb{R} \\ f\neq 0}} \|M \cdot f\|_2 / \|f\|_2$, where the numerator is an expectation 2-norm over $\mathcal{X}$ and the denominator is an expectation 2-norm over $\mathcal{C}$. Thus $\|M\|_2 = \sqrt{|\mathcal{C}|/|\mathcal{X}|} \cdot \sigma_{\max}(M)$, where $\sigma_{\max}(M)$ is the largest singular value of $M$ and $\sqrt{|\mathcal{C}|/|\mathcal{X}|}$ is a normalization factor.

In the case of the matrix $M$ associated with parity learning, $|\mathcal{X}| = |\mathcal{C}| = 2^n$ and all the singular values are equal to $\sqrt{|\mathcal{C}|}$ so $\|M\|_2 = \sqrt{|\mathcal{C}|} = 2^{n/2}$. With this bound, if $v$ is not a node of significant progress then $\|\mathbb{P}_{c|v}\|_2 \leqslant 2^{-(1-\delta/2)n}$ and hence $\|M \cdot \mathbb{P}_{c|v}\|_2 \leqslant 2^{-(1-\delta)n/2}$ which is $1/|\mathcal{X}|^{(1-\delta)/2}$ and hence small.

However, even in the case of learning quadratic functions over $\mathbb{F}_2$, the largest singular value of the matrix $M$ is still $\sqrt{|\mathcal{C}|}$ (the uniform distribution on $\mathcal{C}$ is a singular vector) and so $\|M\|_2 = |\mathcal{C}|/\sqrt{|\mathcal{X}|}$. But in that case, when $\|\mathbb{P}_{c|v}\|_2$ is $|\mathcal{C}|^{-(1-\delta/2)}$ we conclude that $\|M\|_2 \cdot \|\mathbb{P}_{c|v}\|_2$ is at most $|\mathcal{C}|^{\delta/2}/\sqrt{|\mathcal{X}|}$ which is much larger than 1 and hence a useless bound on $\|M \cdot \mathbb{P}_{c|v}\|_2$.

Indeed, the same kind of problem occurs in using the method of Raz [114] for any learning problem for which $|\mathcal{X}|$ is $|\mathcal{C}|^{o(1)}$: If $v$ is a child of the root of the branching program at which the more likely outcome $b$ of a single randomly chosen input $x \in \mathcal{X}$ is remembered, then $\|\mathbb{P}_{c|v}\|_2 \leqslant \sqrt{2}/|\mathcal{C}|$. However, in this case $|(M \cdot \mathbb{P}_{c|v})(x)| = 1$ and so $\|(M \cdot \mathbb{P}_{c|v})\|_2 \geqslant |\mathcal{X}|^{-1/2}$. It follows that $\|M\|_2 \geqslant |\mathcal{C}|/(2|\mathcal{X}|)^{1/2}$ and when $|\mathcal{X}|$ is $|\mathcal{C}|^{o(1)}$ the derived upper bound on $\|M \cdot \mathbb{P}_{c|v'}\|_2$ at nodes $v'$ where $\|\mathbb{P}_{c|v'}\|_2 \geqslant 1/|\mathcal{C}|^{1-\delta/2}$ will be larger than 1 and therefore useless.

We need a more precise way to bound $\|M \cdot \mathbb{P}\|_2$ as a function of $\|\mathbb{P}\|_2$ than using the single number $\|M\|_2$. To do this we will need to use the fact that $\mathbb{P} \in \Delta_{\mathcal{C}}$ – it has a fixed $\ell_1$ norm and (more importantly) it is non-negative and therefore lies in the positive orthant.

**Definition 3.3.1.** *For $M \in \mathbb{C}^{\mathcal{X}\times\mathcal{C}}$ the* 2-norm amplification curve *of $M$, $\tau_M : [0,1] \to \mathbb{R}$ is given by*

$$\tau_M(\delta) = \sup_{\substack{\mathbb{P}\in\Delta_{\mathcal{C}} \\ \|\mathbb{P}\|_2 \leqslant 1/|\mathcal{C}|^{1-\delta/2}}} \log_{|\mathcal{X}|}(\|M \cdot \mathbb{P}\|_2).$$

In other words, whenever $\|\mathbb{P}\|_2$ is at most $|\mathcal{C}|^{-(1-\delta/2)}$, $\|M \cdot \mathbb{P}\|_2$ is at most $|\mathcal{X}|^{\tau_M(\delta)}$. To prove our lower bounds we will bound the norm amplification curves $\tau_{M^{(j)}}$ for all $j \in \{1, \ldots, r-1\}$.

### *3.4 Theorems*

Our general lower bound for learning problems over arbitrary finite label sets is given by following theorem.

**Theorem 3.4.1.** *There are constants $c_1, c_2, c_3 > 0$ such that the follow holds. Let $L : \mathcal{X} \times \mathcal{C} \to \{0, 1, \ldots, r-1\}$ be a labelling function and for $j = 1, \cdots, r-1$ define the matix $M^{(j)} \in \mathbb{C}^{\mathcal{X} \times \mathcal{C}}$ by $M^{(j)}(x, c) = \omega^{j \cdot L(x,c)}$ where $\omega = e^{2\pi i/r}$ and assume[4] that $|\mathcal{X}| \leqslant |\mathcal{C}|$. Suppose that for $0 < \delta' < 1$ we have $\tau_{M^{(j)}}(\delta') \leqslant -\gamma' < 0$ for all $j \in \{1, \cdots, r-1\}$. Then, for $\varepsilon \geqslant c_1 \min(\delta', \gamma') > 0$, $\beta \geqslant c_2 \min(\delta', \gamma') > 0$, and $\eta \geqslant c_3 \, \delta' \, \gamma' > 0$, any algorithm that solves the learning problem for $L$ with success probability at least $|\mathcal{X}|^{-\varepsilon}$ or advantage $\geqslant |\mathcal{X}|^{-\varepsilon/2}$ either requires space at least $\eta \log_2 |\mathcal{X}| \log_r |\mathcal{C}|$ or time at least $|\mathcal{X}|^{\beta}$.*

**Applications to learning polynomials** There are many potential applications of the above theorem but for this work we focus on learning polynomials from their evaluations over finite fields of various sizes. The bounds are derived using the semidefinite programming approach given in Section 3.6 together with analyses for polynomials given in Section 3.7.

**Learning polynomials over $\mathbb{F}_2$** We first consider the case of polynomials over $\mathbb{F}_2$ which yield a binary labelling set. In this case $\omega = -1$ and there is only one matrix $M$, whose entries are $M(x, c) = (-1)^{L(x,c)}$ as in [114].

---

[4] We could write the statement of the theorem to apply to all $\mathcal{X}$ and $\mathcal{C}$ by replacing each occurrence of $|\mathcal{X}|$ in the lower bounds with $\min(|\mathcal{X}|, |\mathcal{C}|)$. When $|\mathcal{X}| \geqslant |\mathcal{C}|$ and $r = 2$, we can use $\|M\|_2$ to bound $\tau_M(\delta')$ which yields the bound given in [114]

The case of linear functions over $\mathbb{F}_2$ is just the parity learning problem. For learning higher degree polynomials over $\mathbb{F}_2$ we obtain the following bounds on the norm amplification curves of their associated matrices:

**Theorem 3.4.2.** *The following norm amplification bounds hold:*

*(a) For all $\delta \in [0,1]$, the matrix $M$ for learning quadratic functions over $\mathbb{F}_2^n$ satisfies*
$$\tau_M(\delta) \leqslant \tfrac{-(1-\delta)}{8} + \tfrac{5+\delta}{8n}.$$

*(b) For any $\zeta > 0$, there are constants $\delta, \gamma$ with $0 < \delta < 1/2$ and $\gamma > 0$ such that for $d \leqslant (1-\zeta)n$ the matrix $M$ for learning functions of degree $\leqslant d$ over $\mathbb{F}_2^n$ satisfies $\tau_M(\delta) \leqslant -\gamma/d$.*

Theorem 3.4.2 is proved in Section 3.7. The case for quadratic polynomials over $\mathbb{F}_2$ follows from properties of the weight distribution of Reed-Muller codes $RM(n,2)$ shown by [123] and [100]. The case for higher degree polynomials over $\mathbb{F}_2$ follows from tail bounds on the bias of $\mathbb{F}_2$ polynomials given by [15].

Using these bounds together with Theorem 3.4.1 yields the following:

**Theorem 3.4.3.** *There are constants $\varepsilon, \zeta > 0$ such that the following hold:*

*(a) Let $m = \binom{n+1}{2}$ for positive integer $n$. Any algorithm for learning quadratic functions over $\mathbb{F}_2^n$ that succeeds with probability at least $2^{-\varepsilon n}$ requires space $\Omega(nm)$ or time $2^{\Omega(n)}$.*

*(b) Let $n > 0$ and $d > 0$ be integers such that $d \leqslant (1-\zeta) \cdot n$ and let $m = \sum_{i=0}^{d} \binom{n}{i}$. Any algorithm for learning polynomial functions of degree at most $d$ over $\mathbb{F}_2^n$ that succeeds with probability at least $2^{-\varepsilon n/d}$ requires space $\Omega(nm/d)$ or time $2^{\Omega(n/d)}$.*

These bounds are tight for constant $d$ since they match the resources used by the natural learning algorithms described in the introduction up to constant factors in the space bound and in the exponent of the time bound.

**Learning polynomials over $\mathbb{F}_p$ for odd prime $p$.** The following theorem bounds the norm amplification curves for polynomials of various degrees over odd prime fields.

**Theorem 3.4.4.** *Let $p$ be an odd prime. For all $\delta \in (0,1)$ and for all $j \in \mathbb{F}_p^*$,*

*(a) the matrices $M^{(j)}$ for learning linear functions over $\mathbb{F}_p^n$ satisfy $\tau_{M^{(j)}}(\delta) \leqslant -\frac{1-\delta}{2}$,*

*(b) the matrices $M^{(j)}$ for learning affine functions over $\mathbb{F}_p^n$ satisfy $\tau_{M^{(j)}}(\delta) \leqslant -\frac{1-\delta}{2} + \frac{\delta}{2n}$,*

*(c) the matrices $M^{(j)}$ for learning quadratic functions over $\mathbb{F}_p^n$ satisfy $\tau_{M^{(j)}}(\delta) \leqslant \frac{-(1-\delta)}{4} + \frac{2}{n}$, and*

*(d) for any $0 < \zeta < 1/2$, there are $\delta, \gamma$ with $0 < \delta < 1/2$ and $\gamma > 0$ such that for $d \leqslant \zeta n$, the matrices $M^{(j)}$ for learning functions of degree $\leqslant d$ over $\mathbb{F}_p^n$ satisfy $\tau_{M^{(j)}}(\delta) \leqslant -\gamma/d$.*

The proof of Theorem 3.4.4 is in Section 3.7. Parts (a) and (b) are almost immediate. The proof of part (c) involves a tight structural characterization of quadratic polynomials over $\mathbb{F}_p$. The proof of part (d) for $d \geqslant 3$ uses tail bounds on the bias of polynomials of degree at most $d$ over $\mathbb{F}_p$, which is proven in Chapter 4.

Using the bounds on the norm amplification curves of Theorem 3.4.4 together with Theorem 3.4.1, we immediately obtain the time-space tradeoff lower bounds in following theorem.

**Theorem 3.4.5.** *Let $p$ be an odd prime. There is an $\varepsilon > 0$ such that the following hold:*

*(a) Any algorithm for learning linear or affine functions over $\mathbb{F}_p^n$ from their evaluations that succeeds with probability at least $p^{-\varepsilon n}$ requires time $p^{\Omega(n)}$ or space $\Omega(n^2 \log p)$.*

*(b) Let $m = \binom{n+2}{2}$. Any algorithm for learning quadratic functions over $\mathbb{F}_p^n$ that succeeds with probability at least $p^{-\varepsilon n}$ requires space $\Omega(nm \log p)$ or time $p^{\Omega(n)}$.*

*(c) There are constants $\zeta, \varepsilon > 0$ such that for $3 \leqslant d \leqslant (1-\zeta) \cdot n$ and for $m$ equal to the number of monomials of degree at most $d$ over $\mathbb{F}_p^n$, any algorithm for learning polynomial*

*functions of degree at most $d$ over $\mathbb{F}_p^n$ that succeeds with probability at least $p^{-\varepsilon n/d}$ requires space $\Omega(\log p \cdot nm/d)$ or time $p^{\Omega(n/d)}$.*

### *3.5 Lower Bounds for Learning Finite Functions from Random Samples*

In this section we prove Theorem 3.4.1. Let $0 < \delta' < 1$ be the value given in the statement of the theorem. To do this we define several positive quantities based on $\delta'$ that will be useful:

- $\delta = \delta'/6$,
- $\alpha = 1 - 2\delta$,
- $\gamma = \min_j\{-\tau_{M^{(j)}}(\delta')/2\}$,
- $\beta = \min(\gamma, \delta)/8$, and
- $\varepsilon = \beta/2$.

Let $B$ be a learning branching program for $L$ with length at most $|\mathcal{X}|^\beta - 1$ and success probability at least $|\mathcal{X}|^{-\varepsilon}$ of identifying the concept (or producing a prediction advantage of more than $|\mathcal{X}|^{-\varepsilon/2}$).

We will prove that $B$ must have width $|\mathcal{C}|^{\Omega(\delta\gamma \log_r |\mathcal{X}|)}$. We first apply the $(\delta, \alpha, \gamma)$-truncation procedure given in Definition 3.2.2 to yield $\mathbb{P}_{c|v}$ and $\mathbb{P}_{e|v}$ for all vertices $v$ in $B$.

The following simple technical lemmas are analogues of ones proved in [114], though we structure our argument somewhat differently. The first uses the bound on the amplification curve of the matrices $M^{(j)}$ for $j \in [r-1]$ in place of its matrix norm.

**Lemma 3.5.1.** *Suppose that vertex $v$ in $B$ is not significant. Then*

$$\mathbf{Pr}_{x \in_R \mathcal{X}}[\exists j \in \{1, \cdots, r-1\}, |(M^{(j)} \cdot \mathbb{P}_{c|v})(x)| \geqslant |\mathcal{X}|^{-\gamma}] \leqslant (r-1) \cdot |\mathcal{X}|^{-2\gamma}.$$

*Proof.* Since $v$ is not significant $\|\mathbb{P}_{c|v}\|_2 \leqslant |\mathcal{C}|^{-(1-\delta/2)}$. For fixed $j \in \{1, \cdots, r-1\}$, by definition of $\tau_{M^{(j)}}$,

$$\mathop{\mathbf{E}}_{x \in_R \mathcal{X}}[|(M^{(j)} \cdot \mathbb{P}_{c|v})(x)|^2] = \|M^{(j)} \cdot \mathbb{P}_{c|v}\|_2^2 \leqslant |\mathcal{X}|^{2\tau_{M^{(j)}}(\delta)} \leqslant |\mathcal{X}|^{2\tau_{M^{(j)}}(\delta')} \leqslant |\mathcal{X}|^{-4\gamma}.$$

Therefore, by Markov's inequality,

$$\mathbf{Pr}_{x \in_R \mathcal{X}}[|(M^{(j)} \cdot \mathbb{P}_{c|v})(x)| \geqslant |\mathcal{X}|^{-\gamma}] = \mathbf{Pr}_{x \in_R \mathcal{X}}[|(M^{(j)} \cdot \mathbb{P}_{c|v})(x)|^2 \geqslant |\mathcal{X}|^{-2\gamma}] \leqslant |\mathcal{X}|^{-2\gamma}.$$

Hence by a union bound,

$$\mathbf{Pr}_{x \in_R \mathcal{X}}[\exists j \in \{1, \cdots, r-1\}, |(M^{(j)} \cdot \mathbb{P}_{c|v})(x)| \geqslant |\mathcal{X}|^{-\gamma}] \leqslant (r-1) \cdot |\mathcal{X}|^{-2\gamma}.$$

□

The second is trivial in the case that $r = 2$ but requires a proof for larger $r$.

**Lemma 3.5.2.** *Suppose that vertex $v$ in $B$ is not significant and that $x \in \mathcal{X}$ has the property that for all $j \in [r-1]$, $|(M^{(j)} \cdot \mathbb{P}_{c|v})(x)| < |\mathcal{X}|^{-\gamma}$. Then for all $y \in \{0, 1, \ldots, r-1\}$,*

$$\left|\mathbf{Pr}_{c' \sim \mathbb{P}_{c|v}}[L(x, c') = y] - \tfrac{1}{r}\right| \leqslant |\mathcal{X}|^{-\gamma}.$$

*Proof.* We apply Proposition 3.2.1: For $y \in \{0, 1, \ldots, r-1\}$, write $g_y = \mathbf{Pr}_{c' \sim \mathbb{P}_{c|v}}[L(x, c') = y]$ and define $g(z) = \sum_{y=0}^{r-1} g_y z^y$. Observe that for $j \in \{1, \ldots, r-1\}$,

$$\begin{aligned}
g(\omega^j) &= \sum_{y=0}^{r-1} \mathbf{Pr}_{c' \sim \mathbb{P}_{c|v}}[L(x, c') = y] \cdot \omega^{jy} \\
&= \sum_{y=0}^{r-1} \sum_{c' \in \mathcal{C}} \mathbb{P}_{c|v}(c') \cdot \mathbf{1}_{L(x,c')=y} \cdot \omega^{jy} \\
&= \sum_{c' \in \mathcal{C}} \mathbb{P}_{c|v}(c') \cdot \sum_{y=0}^{r-1} \mathbf{1}_{L(x,c')=y} \cdot \omega^{jy} \\
&= \sum_{c' \in \mathcal{C}} \mathbb{P}_{c|v}(c') \cdot \omega^{j \cdot L(x,c')} \\
&= (M^{(j)} \cdot \mathbb{P}_{c|v})(x).
\end{aligned}$$

Therefore $|g(\omega^j)| = |(M^{(j)} \cdot \mathbb{P}_{c|v})(x)| < \mathcal{X}^{-\gamma}$. Applying Proposition 3.2.1 immediately yields the lemma. □

**Lemma 3.5.3.** *Suppose that vertex $v$ in $B$ is not significant. Then*

$$\mathbf{Pr}_{c' \sim \mathbb{P}_{c|v}}[c' \in \mathrm{High}(v)] \leqslant |\mathcal{C}|^{-\delta}.$$

*Proof.* Since $v$ is not significant,

$$\mathop{\mathbf{E}}_{c' \sim \mathbb{P}_{c|v}}[\mathbb{P}_{c|v}(c')] = \sum_{c' \in \mathcal{C}} (\mathbb{P}_{c|v}(c'))^2 = |\mathcal{C}| \cdot \|\mathbb{P}_{c|v}\|_2^2 \leqslant |\mathcal{C}|^{-(1-\delta)} = |\mathcal{C}|^{-(\alpha+\delta)}.$$

Therefore since $\alpha = 1 - 2\delta$, by Markov's inequality,

$$\mathbf{Pr}_{c' \sim \mathbb{P}_{c|v}}[c' \in \mathrm{High}(v)] = \mathbf{Pr}_{c' \sim \mathbb{P}_{c|v}}[\mathbb{P}_{c|v}(c') \geqslant |\mathcal{C}|^{-\alpha}] \leqslant |\mathcal{C}|^{-\delta}.$$

□

**Lemma 3.5.4.** *The probability, over uniformly random $c' \in \mathcal{C}$ and uniformly random computation path $C$ in $B$ given concept $c'$, that the truncated version $T$ of $C$ reaches a significant vertex of $B$ is at least $\frac{1}{2}|\mathcal{X}|^{-\beta/2}$.*

*Proof.* Let $c'$ be chosen uniformly at random from $\mathcal{C}$ and consider the truncated path $T$. $T$ will not reach a significant vertex of $B$ only if one of the following holds:

1. $T$ is truncated at a vertex $v$ where $\mathbb{P}_{c|v}(c') \geqslant |\mathcal{C}|^{-\alpha}$.

2. $T$ is truncated at a vertex $v$ because the next edge of $C$ is labelled by $(x, y)$ where $|(M^{(j)} \cdot \mathbb{P}_{c|v})(x)| \geqslant |\mathcal{X}|^{-\gamma}$ for some $j \in \{1, \cdots, r-1\}$.

3. $T$ ends at a leaf that is not significant.

By Lemma 3.5.3, for each vertex $v$ on $C$, conditioned on the truncated path reaching $v$, the probability that $\mathbb{P}_{c|v}(c') \geqslant |\mathcal{C}|^{-\alpha}$ is at most $|\mathcal{C}|^{-\delta}$. Similarly, by Lemma 3.5.1, for each $v$ on the path, conditioned on the truncated path reaching $v$, the probability that $|(M^{(j)} \cdot \mathbb{P}_{c|v})(x)| \geqslant |\mathcal{X}|^{-\gamma}$ for any $j \in [r-1]$ is at most $(r-1) \cdot |\mathcal{X}|^{-2\gamma}$. Therefore, since $T$ has length at most $|\mathcal{X}|^{\beta}$, the probability that $T$ is truncated at $v$ for either reason is at most $|\mathcal{X}|^{\beta}((r-1) \cdot |\mathcal{X}|^{-2\gamma} + |\mathcal{C}|^{-\delta}) < r \cdot |\mathcal{X}|^{-\beta}$ since $|\mathcal{X}| \leqslant |\mathcal{C}|$ and $\beta < \min(\gamma, \delta/2)$.

(*Readers who wish to focus on identification may find it easier on first reading to skip to the alternative proof at the end of this argument*.) For any sink node $v$ of $B$, let $\text{Trunc}(v)$ denote the probability that the random computation path $C$ for a random concept $c'$ chosen uniformly from $\mathcal{C}$ is truncated, conditioned on the computation on $c'$ ending at $v$. By Markov's inequality, the probability that the computation path for a random concept $c'$ ends at vertex $v$ with $\text{Trunc}(v) > 2r \cdot |\mathcal{X}|^{-\beta/2}$ is less than $\frac{1}{2}|\mathcal{X}|^{-\beta/2}$.

Let $f_v : \mathcal{X} \to \{0, 1, \ldots, r-1\}$ denote the function labelling node $v$ which encapsulates the best prediction of the algorithm for the label of each point in $\mathcal{X}$. (This will simply be some $c'' \in \mathcal{C}$ in the case of identification rather than prediction.) We argue that if $v$ is not significant and $\text{Trunc}(v) \leqslant 2r|\mathcal{X}|^{-\beta/2}$ then $f_v$ provides little advantage in predicting $L(\cdot, c')$.

By Lemmas 3.5.1 and 3.5.2, if $v$ is not significant then

$$\mathbf{Pr}_{c' \sim \mathbb{P}_{c|v},\, x \in_R \mathcal{X}}[L(x, c') = f_v(x)] \leqslant \frac{1}{r} + |\mathcal{X}|^{-\gamma} + (r-1) \cdot |\mathcal{X}|^{-2\gamma}.$$

Now $\mathbb{P}_{c|v}$ is the distribution on concepts $c' \in \mathcal{C}$ conditioned on their (randomly chosen) computation path reaching $v$ and not being truncated. On the other hand, correctness is defined with respect to randomly chosen computation paths independent of truncation. However, if $\text{Trunc}(v) \leqslant 2r \cdot |\mathcal{X}|^{-\beta/2}$, then over the distribution independent of truncation we obtain that the probability conditioned on $c'$ reaching $v$ of predicting $L(x, c')$ is at most

$$\frac{1}{r} + |\mathcal{X}|^{-\gamma} + (r-1) \cdot |\mathcal{X}|^{-2\gamma} + 2r \cdot |\mathcal{X}|^{-\beta/2} \leqslant \frac{1}{r} + |\mathcal{X}|^{-\varepsilon/2}$$

which is a prediction advantage of at most $|\mathcal{X}|^{-\varepsilon/2}$. Therefore none of the nodes non-significant nodes $v$ with small $\text{Trunc}(v)$ can contribute to the $|\mathcal{X}|^{-\varepsilon}$ success probability and hence the probability that the computation reaches a significant node must be at least the success probability $|\mathcal{X}|^{-\varepsilon}$ of having a large advantage minus the probability that the computation on $c'$ reaches a sink vertex $v$ with $\text{Trunc}(v) > 2r \cdot |\mathcal{X}|^{-\beta/2}$, which is $|\mathcal{X}|^{-\varepsilon} - \frac{1}{2}|\mathcal{X}|^{-\beta/2} = \frac{1}{2}|\mathcal{X}|^{-\beta/2}$ as required.

(*An alternative simpler argument that may be a bit more intuitive for the case of identification*.) If $T$ reaches a leaf $v$ that is not significant then, conditioned on arriving at $v$, the probability

that the concept $c'$ equals the label of $v$ is at most $\max_{c'' \in \mathcal{C}} \mathbb{P}_{c|v}(c'')$. Now

$$\frac{\max_{c'' \in \mathcal{C}} \mathbb{P}_{c|v}(c'')}{|\mathcal{C}|^{1/2}} \leqslant \|\mathbb{P}_{c|v}\|_2 < |\mathcal{C}|^{-(1-\delta/2)}$$

since $v$ is not significant, so we have $\max_{c'' \in \mathcal{C}} \mathbb{P}_{c|v}(c'') < |\mathcal{C}|^{-(1-\delta)/2} = |\mathcal{C}|^{-(\alpha+\delta)/2}$ and the probability that $B$ is correct conditioned on the truncated path reaching a leaf vertex that is not significant is less than $|\mathcal{C}|^{-(\alpha+\delta)/2} \leqslant |\mathcal{C}|^{-\beta} \leqslant |\mathcal{X}|^{-\beta}$ since $|\mathcal{X}| \leqslant |\mathcal{C}|$.

Since $B$ is correct with probability at least $|\mathcal{X}|^{-\varepsilon} = |\mathcal{X}|^{-\beta/2}$ and these three cases in which $T$ does not reach a significant vertex account for correctness at most $(r+1) \cdot |\mathcal{X}|^{-\beta}$, which is much less than $\frac{1}{2} \cdot |\mathcal{X}|^{-\beta/2}$, $T$ must reach a significant vertex with probability at least $\frac{1}{2}|\mathcal{X}|^{-\beta/2}$. □

The following lemma is the the key to the proof of the theorem.

**Lemma 3.5.5.** *Let $s$ be any significant vertex of $B$. There is an $\eta = \delta\gamma/2 > 0$ such that for a uniformly random $x$ chosen from $\mathcal{C}$ and a uniformly random computation path $C$, the probability that its truncation $T$ ends at $s$ is at most $|\mathcal{C}|^{-\eta \log_r |\mathcal{X}|}$.*

The proof of Lemma 3.5.5 requires a delicate progress argument and is deferred to the next subsection. We first show how Lemmas 3.5.4 and 3.5.5 immediately imply Theorem 3.4.1.

*Proof of Theorem 3.4.1.* By Lemma 3.5.4, for $c$ chosen uniformly at random from $\mathcal{C}$ and $T$ the truncation of a uniformly random computation path given concept $c$, $T$ ends at a significant vertex with probability at least $|\mathcal{X}|^{-\beta/2}/2$. On the other hand, by Lemma 3.5.5, for any significant vertex $s$, the probability that $T$ ends at $s$ is at most $|\mathcal{C}|^{-\eta \log_r |\mathcal{X}|}$. Therefore the number of significant vertices must be at least $2|\mathcal{C}|^{\eta \log_r |\mathcal{X}|}/|\mathcal{X}|^{\beta/2}$ and since $B$ has length at most $|\mathcal{X}|^{\beta}$, there must be at least $2|\mathcal{C}|^{\eta \log_r |\mathcal{X}|}/|\mathcal{X}|^{3\beta/2}$ significant vertices in some layer. Hence $B$ requires space $\Omega(\delta\gamma \log_2 |\mathcal{C}| \log_r |\mathcal{X}|)$. □

### 3.5.1 *Progress towards significance*

In this section we prove Lemma 3.5.5 showing that for any particular significant vertex $s$ a random truncated path reaches $s$ only with probability $|\mathcal{C}|^{-\Omega(\delta\gamma \log_r |\mathcal{X}|)}$. For each vertex $v$ in $B$ let $\mathbf{Pr}[v]$ denote the probability over a random concept $c$, that the truncation of a random computation path in $B$ given concept $c$ visits $v$ and for each edge $e$ in $B$ let $\mathbf{Pr}[e]$ denote the probability over a random concept $c$, that the truncation of a random computation path in $B$ for $c$ traverses $e$.

Since $B$ is a leveled branching program, the vertices of $B$ may be divided into disjoint sets $V_t$ for $t = 0, 1, \ldots, T$ where $T$ is the length of $B$ and $V_t$ is the set of vertices at distance $t$ from the root, and disjoint sets of edges $E_t$ for $t = 1, \ldots, T$ where $E_t$ consists of the edges from $V_{t-1}$ to $V_t$. For each vertex $v \in V_{t-1}$, note that by definition we only have

$$\mathbf{Pr}[v] \geqslant \sum_{(v,w)\in E_t} \mathbf{Pr}[(v,w)]$$

since some truncated paths may terminate at $v$.

For each $t$, since the truncated computation path visits at most one vertex and at most one edge at level $t$, we obtain a sub-distribution on $V_t$ in which the probability of $v \in V_t$ is $\mathbf{Pr}[v]$ and a corresponding sub-distribution on $E_t$ in which the probability of $e \in E_t$ is $\mathbf{Pr}[e]$. We write $v \sim V_t$ and $e \sim E_t$ to denote random selection from these sub-distributions, where the outcome $\perp$ corresponds to the case that no vertex (respectively no edge) is selected.

Fix some significant vertex $s$. We consider the progress that a truncated path makes as it moves from the start vertex to $s$. We measure the progress at a vertex $v$ as

$$\rho(v) = \frac{\langle \mathbb{P}_{c|v}, \mathbb{P}_{c|s} \rangle}{\langle \mathbb{P}_{c|s}, \mathbb{P}_{c|s} \rangle}.$$

Clearly $\rho(s) = 1$. We first see that $\rho$ starts out at a tiny value.

**Lemma 3.5.6.** *If $v_0$ is the start vertex of $B$ then $\rho(v_0) \leqslant |\mathcal{C}|^{-\delta}$.*

*Proof.* By definition, $\mathbb{P}_{c|v_0}$ is the uniform distribution on $\mathcal{C}$. Therefore

$$\langle \mathbb{P}_{c|v_0}, \mathbb{P}_{c|s} \rangle = \mathop{\mathbf{E}}_{c' \in \mathcal{C}}[|\mathcal{C}|^{-1} \cdot \mathbb{P}_{c|s}(c')] = |\mathcal{C}|^{-2} \cdot \sum_{c' \in \mathcal{C}} \mathbb{P}_{c|v_0}(c') = |\mathcal{C}|^{-2}$$

since $\mathbb{P}_{c|s}$ is a probability distribution on $\mathcal{C}$. On the other hand, since $s$ is significant, $\langle \mathbb{P}_{c|s}, \mathbb{P}_{c|s} \rangle = \|\mathbb{P}_{c|s}\|_2^2 \geqslant |\mathcal{C}|^{\delta} \cdot |\mathcal{C}|^{-2}$. The lemma follows immediately. □

Since the truncated path is randomly chosen, the progress towards $s$ after $t$ steps is a random variable. We show that not only is the increase in this expected value of this random variable in each step very small, its higher moments also increase at a very small rate. Define

$$\Phi_t = \mathop{\mathbf{E}}_{v \sim V_t}[(\rho(v))^{\gamma \log_r |\mathcal{X}|}]$$

where we extend $\rho$ and define $\rho(\perp) = 0$. We will show that for $s \in V_t$, $\Phi_t$ is still $|\mathcal{C}|^{-\Omega(\delta\gamma \log_r |\mathcal{X}|)}$, which will be sufficient to prove Lemma 3.5.5.

Therefore, Lemma 3.5.5, and hence Theorem 3.4.1, will follow from the following lemma.

**Lemma 3.5.7.** *For every $t$ with $1 \leqslant t \leqslant |\mathcal{X}|^{\beta} - 1$,*

$$\Phi_t \leqslant \Phi_{t-1} \cdot (1 + |\mathcal{X}|^{-2\beta}) + |\mathcal{C}|^{-\gamma \log_r |\mathcal{X}|}.$$

*Proof of Lemma 3.5.5 from Lemma 3.5.7.* By definition of $\Phi_t$ and Lemma 3.5.6 we have $\Phi_0 \leqslant |\mathcal{C}|^{-\delta\gamma \log_r |\mathcal{X}|}$. By Lemma 3.5.7, for every $t$ with $1 \leqslant t \leqslant |\mathcal{X}|^{\beta} - 1$,

$$\Phi_t \leqslant \sum_{j=0}^{t} (1 + |\mathcal{X}|^{-2\beta})^j \cdot |\mathcal{C}|^{-\delta\gamma \log_r |\mathcal{X}|} < (t+1) \cdot (1 + |\mathcal{X}|^{-2\beta})^t \cdot |\mathcal{C}|^{-\delta\gamma \log_r |\mathcal{X}|}.$$

In particular, for every $t \leqslant |\mathcal{X}|^{\beta} - 1$,

$$\Phi_t \leqslant |\mathcal{X}|^{\beta} \cdot (1 + |\mathcal{X}|^{-2\beta})^{|\mathcal{X}|^{\beta}} \cdot |\mathcal{C}|^{-\delta\gamma \log_r |\mathcal{X}|} \leqslant e^{1/|\mathcal{X}|^{\beta}} \cdot |\mathcal{X}|^{\beta} \cdot |\mathcal{C}|^{-\delta\gamma \log_r |\mathcal{X}|}.$$

Now fix $t^*$ to be the level of the significant node $s$. Every truncated path that reaches $s$ will have contribution to $\Phi_{t^*}$ of $(\rho(s))^{\gamma \log_r |\mathcal{X}|} = 1$ times its probability of occurring. Therefore the truncation of a random computation path reaches $s$ with probability at most $|\mathcal{C}|^{-\eta \log_r |\mathcal{X}|}$ for $\eta = \delta\gamma/2$ and $|\mathcal{X}|, |\mathcal{C}|$ sufficiently large, which proves the lemma. □

We now focus on the proof of Lemma 3.5.7. Because $\Phi_t$ depends on the sub-distribution over $V_t$ and $\Phi_{t-1}$ depends on the sub-distribution over $V_{t-1}$, it is natural to consider the analogous quantities based on the sub-distribution over the set $E_t$ of edges that join $V_{t-1}$ and $V_t$. We can extend the definition of $\rho$ to edges of $B$, where we write

$$\rho(e) = \frac{\langle \mathbb{P}_{c|e}, \mathbb{P}_{c|s} \rangle}{\langle \mathbb{P}_{c|s}, \mathbb{P}_{c|s} \rangle}.$$

Then define

$$\Phi'_t = \mathop{\mathbf{E}}_{e \sim E_t}[(\rho(e))^{\gamma \log_r |\mathcal{X}|}].$$

Intuitively, there is no gain of information in moving from elements $E_t$ to elements of $V_t$. More precisely, we have the following lemma:

**Lemma 3.5.8.** *For all $t$, $\Phi_t \leqslant \Phi'_t$.*

*Proof.* Note that for $v \in V_t$, since the truncated paths that follow some edge $(u, v) \in E_t$ are precisely those that reach $v$, by definition, $\mathbf{Pr}[v] = \sum_{(u,v)\in E_t} \mathbf{Pr}[(u, v)]$. Since the same applies separately to the set of truncated paths for each concept $c' \in \mathcal{C}$ that reach $v$, for each $c' \in \mathcal{C}$ we have

$$\mathbf{Pr}[v] \cdot \mathbb{P}_{c|v}(c') = \sum_{(u,v)\in E_t} \mathbf{Pr}[(u, v)] \cdot \mathbb{P}_{c|(u,v)}(c').$$

Therefore,

$$\mathbf{Pr}[v] \cdot \frac{\langle \mathbb{P}_{c|v}, \mathbb{P}_{c|s} \rangle}{\langle \mathbb{P}_{c|s}, \mathbb{P}_{c|s} \rangle} = \sum_{(u,v)\in E_t} \mathbf{Pr}[(u, v)] \cdot \frac{\langle \mathbb{P}_{c|(u,v)}, \mathbb{P}_{c|s} \rangle}{\langle \mathbb{P}_{c|s}, \mathbb{P}_{c|s} \rangle};$$

i. e., $\mathbf{Pr}[v] \cdot \rho(v) = \sum_{(u,v)\in E_t} \mathbf{Pr}[(u, v)] \cdot \rho((u, v))$. Since $\mathbf{Pr}[v] = \sum_{(u,v)\in E_t} \mathbf{Pr}[(u, v)]$, by the convexity of the map $s \mapsto s^{\gamma \log_r |\mathcal{X}|}$ we have

$$\mathbf{Pr}[v] \cdot (\rho(v))^{\gamma \log_r |\mathcal{X}|} \leqslant \sum_{(u,v)\in E_t} \mathbf{Pr}[(u, v)] \cdot (\rho((u, v))^{\gamma \log_r |\mathcal{X}|}.$$

Therefore

$$\begin{aligned}\Phi_t = \sum_{v\in V_t} \mathbf{Pr}[v] \cdot (\rho(v))^{\gamma \log_r |\mathcal{X}|} &\leqslant \sum_{v\in V_t} \sum_{(u,v)\in E_t} \mathbf{Pr}[(u, v)] \cdot (\rho((u, v)))^{\gamma \log_r |\mathcal{X}|} \\ &= \sum_{e\in E_t} \mathbf{Pr}[e] \cdot (\rho(e))^{\gamma \log_r |\mathcal{X}|} = \Phi'_t.\end{aligned}$$

□

Therefore, to prove Lemma 3.5.7 it suffices to prove that the same statement holds with $\Phi_t$ replaced by $\Phi'_t$; that is,

$$\mathop{\mathbf{E}}_{e\in E_t}[(\rho(e))^{\gamma\log_r|\mathcal{X}|}] \leqslant (1+|\mathcal{X}|^{-2\beta})\cdot \mathop{\mathbf{E}}_{v\in V_{t-1}}[(\rho(v))^{\gamma\log_r|\mathcal{X}|}] + |\mathcal{C}|^{-\gamma\log_r|\mathcal{X}|}$$

$E_t$ is the disjoint union of the out-edges $\Gamma_{out}(v)$ for vertices $v \in V_{t-1}$, so it suffices to show that for each $v \in V_{t-1}$,

$$\sum_{e\in\Gamma_{out}(v)} \mathbf{Pr}[e]\cdot(\rho(e))^{\gamma\log_r|\mathcal{X}|} \leqslant (1+|\mathcal{X}|^{-2\beta})\cdot\mathbf{Pr}[v]\cdot(\rho(v))^{\gamma\log_r|\mathcal{X}|} + |\mathcal{C}|^{-\gamma\log_r|\mathcal{X}|}\cdot\mathbf{Pr}[v]. \tag{3.1}$$

Since any truncated path that follows $e$ must also visit $v$, we can write $\mathbf{Pr}[e|v] = \mathbf{Pr}[e]/\mathbf{Pr}[v]$. Moreover, both $\rho(v)$ and $\rho(e)$ have the same denominator $\langle \mathbb{P}_{c|s}, \mathbb{P}_{c|s}\rangle$ and therefore, by definition, inequality (3.1), and hence Lemma 3.5.7, follows from the following lemma.

**Lemma 3.5.9.** *For $v \in V_{t-1}$,*

$$\sum_{e\in\Gamma_{out}(v)} \mathbf{Pr}[e|v]\cdot\langle\mathbb{P}_{c|e},\mathbb{P}_{c|s}\rangle^{\gamma\log_r|\mathcal{X}|} \leqslant (1+|\mathcal{X}|^{-2\beta})\cdot\langle\mathbb{P}_{c|v},\mathbb{P}_{c|s}\rangle^{\gamma\log_r|\mathcal{X}|} + |\mathcal{C}|^{-\gamma\log_r|\mathcal{X}|}.$$

Before we prove Lemma 3.5.9, we first prove some technical lemmas, the first relating the distributions for $v \in V_{t-1}$ and edges $e \in E_t$ and the last upper bounding $\|\mathbb{P}_{c|s}\|_2$.

**Lemma 3.5.10.** *Suppose that $v \in V_{t-1}$ is not significant and $e = (v, w) \in E_t$ has $\mathbf{Pr}[e] > 0$ and label $(x, y)$. Then for $c' \in \mathcal{C}$, $\mathbb{P}_{c|e}(c') > 0$ only if $c' \notin \mathrm{High}(v)$ and $L(x, c') = y$, in which case*

$$\mathbb{P}_{c|e}(c') = c_e^{-1}\cdot\mathbb{P}_{c|v}(c')$$

*where $c_e \geqslant \frac{1}{r} - |\mathcal{X}|^{-\gamma} - |\mathcal{C}|^{-\delta}$.*

*Proof.* If there exists $j \in \{1, \cdots, r-1\}$ such that $|(M^{(j)}\cdot\mathbb{P}_{c|v})(x)| \geqslant |\mathcal{X}|^{-\gamma}$ then by definition of truncation we also will have $\mathbf{Pr}[e] = 0$. Therefore, since $\mathbf{Pr}[e] > 0$, $e$ is not a high

bias edge – that is, $\forall j \in [r-1]$, $|(M^{(j)} \cdot \mathbb{P}_{c|v})(x)| < |\mathcal{X}|^{-\gamma}$. We now use Lemma 3.5.2 to derive that

$$\mathbf{Pr}_{c' \sim \mathbb{P}_{c|v}}[L(x, c') = y] \geqslant \frac{1}{r} - |\mathcal{X}|^{-\gamma}.$$

Let $\mathcal{E}_e(c')$ be the event that both $L(x, c') = y$ and $c' \notin \text{High}(v)$ and define

$$c_e = \mathbf{Pr}_{c' \sim \mathbb{P}_{c|v}}[\mathcal{E}_e(c')].$$

If $\mathcal{E}_e(c')$ fails to hold for all $c'$, i. e., $c' \in \text{High}(v)$ or $L(x, c') \neq y$, then any truncated path for concept $c'$ that reaches $v$ will not continue along $e$ and hence $\mathbf{Pr}[e] = 0$. On the other hand, since $\mathbf{Pr}[e] > 0$, if $\mathcal{E}_e(c')$ holds for some concept $c'$ then any truncated path for $c'$ that reaches $v$ will continue precisely if the input chosen at $v$ is $a$, which happens with probability $|\mathcal{X}|^{-1}$ for each such $c'$. The total probability over $c' \in \mathcal{C}$, conditioned that the truncated path on $c'$ reaches $v$ and that the path continues along $e$ is then $|\mathcal{X}|^{-1} \cdot c_e$. Therefore, if $c' \in \mathcal{E}_e$ then $\mathbb{P}_{c|e}(c') = \frac{|\mathcal{X}|^{-1} \cdot \mathbb{P}_{c|v}(c')}{|\mathcal{X}|^{-1} \cdot c_e} = c_e^{-1} \cdot \mathbb{P}_{c|v}(c')$. Now by Lemma 3.5.3,

$$\mathbf{Pr}_{c' \sim \mathbb{P}_{c|v}}[c' \in \text{High}(v)] \leqslant |\mathcal{C}|^{-\delta}$$

and so

$$c_e = \mathbf{Pr}_{c' \sim \mathbb{P}_{c|v}}[L(x, c') = y \text{ and } c' \notin \text{High}(v)] > \frac{1}{r} - |\mathcal{X}|^{-\gamma} - |\mathcal{C}|^{-\delta}$$

as required. □

We use this lemma together with an argument similar to that of Lemma 3.5.8 to upper bound $\|\mathbb{P}_{c|s}\|_2$ for our significant vertex $s$.

**Lemma 3.5.11.** $\|\mathbb{P}_{c|s}\|_2 \leqslant 2r \cdot |\mathcal{C}|^{-(1-\delta/2)}$.

*Proof.* The main observation is that $s$ is the first significant vertex of any truncated path that reaches it and so the probability distributions of each of the immediate predecessors $v$ of $s$ must have bounded expectation 2-norm and, by Lemma 3.5.10 and the proof idea from Lemma 3.5.8, the 2-norm of the distribution at $s$ cannot grow too much larger than those at its immediate predecessors.

By Lemma 3.5.10, if $e = (v, s)$ and $\mathbf{Pr}[e] > 0$, then

$$\|\mathbb{P}_{c|e}\|_2 \leqslant c_e^{-1} \cdot \|\mathbb{P}_{c|v}\| \leqslant c_e^{-1}|\mathcal{C}|^{-(1-\delta/2)} \leqslant 2r \cdot |\mathcal{C}|^{-(1-\delta/2)}$$

since $v$ is not significant and $c_e \geqslant \frac{1}{r} - |\mathcal{X}|^{-\gamma} - |\mathcal{C}|^{-\delta} > \frac{1}{2r}$ for $|\mathcal{X}|$ and $|\mathcal{C}|$ sufficiently large. Let $\Gamma_{in}(s)$ be the set of edges $(v, s)$ in $B$. $\mathbf{Pr}[s] = \sum_{e=(v,s)\in\Gamma_{in}(s)} \mathbf{Pr}[e]$ and for each $c' \in \mathcal{C}$,

$$\mathbf{Pr}[s] \cdot \mathbb{P}_{c|s}(c') = \sum_{e=(v,s)\in\Gamma_{in}(s)} \mathbf{Pr}[e] \cdot \mathbb{P}_{c|e}(c').$$

Since $\mathbf{Pr}[s] = \sum_{e=(v,s)\in\Gamma_{in}(s)} \mathbf{Pr}[e]$, by convexity of the map $r \mapsto r^2$, we have

$$\mathbf{Pr}[s] \cdot (\mathbb{P}_{c|s}(c'))^2 \leqslant \sum_{e=(v,s)\in\Gamma_{in}(s)} \mathbf{Pr}[e] \cdot (\mathbb{P}_{c|e}(c'))^2.$$

Summing over $c' \in \mathcal{C}$ we have

$$\begin{aligned}\mathbf{Pr}[s] \cdot \|\mathbb{P}_{c|s}\|_2^2 &\leqslant \sum_{e=(v,s)\in\Gamma_{in}(s)} \mathbf{Pr}[e] \cdot \|\mathbb{P}_{c|e}\|_2^2 \\ &\leqslant \sum_{e=(v,s)\in\Gamma_{in}(s)} \mathbf{Pr}[e] \cdot (2r \cdot |\mathcal{C}|^{-(1-\delta/2)})^2 = \mathbf{Pr}[s] \cdot (2r \cdot |\mathcal{C}|^{-(1-\delta/2)})^2.\end{aligned}$$

Therefore $\|\mathbb{P}_{c|s}\| \leqslant 2r \cdot |\mathcal{C}|^{-(1-\delta/2)}$ as required since $\mathbf{Pr}[s] > 0$. □

To complete the proof of Lemma 3.5.7, and hence Lemma 3.5.5, it only remains to prove Lemma 3.5.9.

*Proof of Lemma 3.5.9*

Since we know that if $v \in V_{t-1}$ is significant then any edge $e \in \Gamma_{out}(v)$ has $\mathbf{Pr}[e] = 0$, we can assume without loss of generality that $v$ is not significant.

Define $g : \mathcal{C} \to \mathbb{R}$ by

$$g(c') = \mathbb{P}_{c|v}(c') \cdot \mathbb{P}_{c|s}(c')$$

and note that $\langle \mathbb{P}_{c|v}, \mathbb{P}_{c|s} \rangle = \mathbf{E}_{c'\in\mathcal{C}}[g(c')]$. For $c' \in \mathcal{C}$ define

$$f(c') = \begin{cases} g(c') & c' \notin \mathrm{High}(v) \\ 0 & \text{otherwise} \end{cases}$$

and let $F = \sum_{c' \in \mathcal{C}} f(c')$. For every edge $e$ where $\langle \mathbb{P}_{c|e}, \mathbb{P}_{c|s} \rangle > 0$, we have $F > 0$.

The function $f$ induces a new probability distribution on $\mathcal{C}$, $\mathbb{P}_f$, given by $\mathbb{P}_f(c') = f(c')/\sum_{x \in \mathcal{C}} f(x) = f(c')/F$ in which each point $c' \in \mathcal{C} \setminus \mathrm{High}(v)$ is chosen with probability proportional to its contribution to $\langle \mathbb{P}_{c|v}, \mathbb{P}_{c|s} \rangle$ and each $c' \in \mathrm{High}(v)$ has probability 0.

CLAIM: Let $(x, y)$ be the label on an edge $e$, then

$$\begin{aligned}\langle \mathbb{P}_{c|e}, \mathbb{P}_{c|s} \rangle &\leqslant (rc_e)^{-1} \cdot (1 + \sum_{j=1}^{r-1} |(M^{(j)} \cdot \mathbb{P}_f)(x)|) \cdot F/|\mathcal{C}| \\ &\leqslant (rc_e)^{-1} \cdot (1 + \sum_{j=1}^{r-1} |(M^{(j)} \cdot \mathbb{P}_f)(x)|) \cdot \langle \mathbb{P}_{c|v}, \mathbb{P}_{c|s} \rangle\end{aligned}$$

where $c_e$ is given by Lemma 3.5.10.

We first prove the claim. By Lemma 3.5.10 and the definition of $f$,

$$\mathbb{P}_{c|e}(c') \cdot \mathbb{P}_{c|s}(c') = \begin{cases} c_e^{-1} \cdot f(c') & \text{if } L(x, c') = y \\ 0 & \text{otherwise.} \end{cases}$$

Therefore

$$\langle \mathbb{P}_{c|e}, \mathbb{P}_{c|s} \rangle = \mathop{\mathbf{E}}_{c' \in_R \mathcal{C}}[\mathbb{P}_{c|e}(c') \cdot \mathbb{P}_{c|s}(c')] = \mathop{\mathbf{E}}_{c' \in_R \mathcal{C}}[c_e^{-1} f(c') \cdot \mathbf{1}_{L(x,c')=y}]$$

Let $z = M^{(1)}(x, c') \cdot \omega^{-b}$. Then $z \in \{1, \omega, \cdots, \omega^{r-1}\}$. The indicator function $\mathbf{1}_{L(x,c')=y}$ is 1 when $z = 1$, and $\mathbf{1}_{L(x,c')=y}$ is 0 when $z = \omega, \cdots, \omega^{r-1}$. By interpolation we have

$$\mathbf{1}_{L(x,c')=y} = \frac{1}{r} \sum_{j=0}^{r-1} z^j$$

Notice that $z^j = \omega^{-bj} \cdot M^{(j)}(x, c')$ for $j = 1, \cdots, r-1$, so we have

$$\begin{aligned}
\langle \mathbb{P}_{c|e}, \mathbb{P}_{c|s} \rangle &= \mathop{\mathbf{E}}_{c' \in_R \mathcal{C}} \left[ c_e^{-1} f(c') \cdot (1 + \sum_{j=1}^{r-1} \omega^{-y \cdot j} \cdot M^{(j)}(x, c'))/r \right] \\
&= (rc_e)^{-1} \cdot \left( \mathop{\mathbf{E}}_{c' \in_R \mathcal{C}}[f(c')] + \sum_{j=1}^{r-1} \omega^{-y \cdot j} \cdot \mathop{\mathbf{E}}_{c' \in_R \mathcal{C}}[M^{(j)}(x, c') \cdot f(c')] \right) \\
&\leqslant (rc_e)^{-1} \cdot \left( \mathop{\mathbf{E}}_{c' \in_R \mathcal{C}}[f(c')] + \sum_{j=1}^{r-1} \left| \mathop{\mathbf{E}}_{c' \in_R \mathcal{C}}[M^{(j)}(x, c') \cdot f(c')] \right| \right) \\
&= (rc_e)^{-1} \cdot |\mathcal{C}|^{-1} \cdot F \cdot \left( 1 + \frac{\sum_{j=1}^{r-1} \left| \mathbf{E}_{c' \in_R \mathcal{C}}[M^{(j)}(x, c') \cdot f(c')] \right|}{F} ] \right) \\
&= (rc_e)^{-1} \cdot |\mathcal{C}|^{-1} \cdot F \cdot (1 + \sum_{j=1}^{r-1} |(M^{(j)} \cdot \mathbb{P}_f)(x)|) \\
&\leqslant (rc_e)^{-1} \cdot (1 + \sum_{j=1}^{r-1} |(M^{(j)} \cdot \mathbb{P}_f)(x)|) \cdot \langle \mathbb{P}_{c|v}, \mathbb{P}_{c|s} \rangle
\end{aligned}$$

since $|\mathcal{C}|^{-1} \cdot F = \mathbf{E}_{c' \in_R \mathcal{C}}[f(c')] \leqslant \mathbf{E}_{c' \in_R \mathcal{C}}[g(c')] = \langle \mathbb{P}_{c|v}, \mathbb{P}_{c|s} \rangle$, which proves the claim.

By Lemma 3.5.10, $rc_e \geqslant 1 - r \cdot |\mathcal{X}|^{-\gamma} - r \cdot |\mathcal{C}|^{-\delta}$ and so $(rc_e)^{-1} \leqslant 1 + |\mathcal{X}|^{-\sigma} \leqslant 2$ for $\sigma = \min(\gamma, \delta)/2$ and sufficiently large $|\mathcal{X}|$ since $|\mathcal{X}| \leqslant |\mathcal{C}|$. We consider two cases:

CASE $F \leqslant |\mathcal{C}|^{-1}$: In this case, since $\mathbb{P}_f$ is a probability distribution, for every $x \in \mathcal{X}$ and $j \in [r-1]$, we have $|(M^{(j)} \cdot \mathbb{P}_f)(x)| \leqslant \max_{c' \in \mathcal{C}} |M^{(j)}(x, c')| = 1$ and from the claim we obtain for every edge $e \in \Gamma_{out}(v)$, $\langle \mathbb{P}_{c|e}, \mathbb{P}_{c|s} \rangle \leqslant r \cdot (rc_e)^{-1} \cdot |\mathcal{C}|^{-2}$. Therefore $\sum_{e \in \Gamma_{out}(v)} \mathbf{Pr}[e|v] \cdot \langle \mathbb{P}_{c|e}, \mathbb{P}_{c|s} \rangle^{\gamma \log_r |\mathcal{X}|}$ is at most $(2r \cdot |\mathcal{C}|^{-2})^{\gamma \log_r |\mathcal{X}|} \leqslant |\mathcal{C}|^{-\gamma \log_r |\mathcal{X}|}$ for $|\mathcal{C}| \geqslant 2r$.

CASE $F \geqslant |\mathcal{C}|^{-1}$: In this case we will show that $\|\mathbb{P}_f\|_2$ is not too large and use this together with the bound on the 2-norm amplification curve of each $M^{(j)}$ to show that $\|M^{(j)} \cdot \mathbb{P}_f\|_2$ is small for each $j = 1, \cdots, r-1$. This will be important because of the following connection:

By the Claim, we have

$$\sum_{e\in\Gamma_{out}(v)} \mathbf{Pr}[e|v]\cdot\langle \mathbb{P}_{c|e},\mathbb{P}_{c|s}\rangle^{\gamma\log_r|\mathcal{X}|} \tag{3.2}$$

$$\leqslant \sum_{e\in\Gamma_{out}(v)} \mathbf{Pr}[e|v]\cdot[(rc_e)^{-1}\cdot(1+\sum_{j=1}^{r-1}|(M^{(j)}\cdot\mathbb{P}_f)(x_e)|)]^{\gamma\log_r|\mathcal{X}|}\cdot\langle \mathbb{P}_{c|v},\mathbb{P}_{c|s}\rangle^{\gamma\log_r|\mathcal{X}|} \tag{3.3}$$

where $a_e$ is the input labelling edge $e$. By definition, for each $x\in\mathcal{X}$ there are precisely $r$ edges $e_0,\cdots,e_{r-1}\in\Gamma_{out}(v)$ with $a_{e_i}=a$ for $i=0,\cdots,r-1$ and $\sum_{i=0}^{r-1}\mathbf{Pr}[e_i|v]\leqslant 1/|\mathcal{X}|$ since the next input is chosen uniformly at random from $\mathcal{X}$. (It would be equality but some inputs $a$ have high bias and in that case $\mathbf{Pr}[e_i|v]=0$ for all $i$.) Previously, we also observed that $(rc_e)^{-1}\leqslant 1+|\mathcal{X}|^{-\sigma}$ where $\sigma=\min(\gamma,\delta)/2$. Therefore,

$$\begin{aligned}
&\sum_{e\in\Gamma_{out}(v)} \mathbf{Pr}[e|v]\cdot\langle \mathbb{P}_{c|e},\mathbb{P}_{c|s}\rangle^{\gamma\log_r|\mathcal{X}|}\\
&\leqslant \sum_{x\in\mathcal{X}}\frac{1}{|\mathcal{X}|}[(1+|\mathcal{X}|^{-\sigma})\cdot(1+\sum_{j=1}^{r-1}|(M^{(j)}\cdot\mathbb{P}_f)(x_e)|)]^{\gamma\log_r|\mathcal{X}|}\cdot\langle \mathbb{P}_{c|v},\mathbb{P}_{c|s}\rangle^{\gamma\log_r|\mathcal{X}|}\\
&= (1+|\mathcal{X}|^{-\sigma})^{\gamma\log_r|\mathcal{X}|}\cdot\mathop{\mathbf{E}}_{x\in_R\mathcal{X}}[(1+\sum_{j=1}^{r-1}|(M^{(j)}\cdot\mathbb{P}_f)(x_e)|)^{\gamma\log_r|\mathcal{X}|}]\cdot\langle \mathbb{P}_{c|v},\mathbb{P}_{c|s}\rangle^{\gamma\log_r|\mathcal{X}|}.
\end{aligned}$$

To prove the lemma we therefore need to bound $\mathbf{E}_{x\in_R\mathcal{X}}[(1+\sum_{j=1}^{r-1}|(M^{(j)}\cdot\mathbb{P}_f)(x_e)|)^{\gamma\log_r|\mathcal{X}|}]$. We will bound this by first analyzing $\|M^{(j)}\cdot\mathbb{P}_f\|_2$ for each $j$.

Fix $j$. By definition,

$$\|f\|_2^2=\mathop{\mathbf{E}}_{c'\in_R\mathcal{C}}\mathbf{1}_{c'\notin\mathrm{High}(v)}\cdot\mathbb{P}_{c|v}^2(c')\cdot\mathbb{P}_{c|s}^2(c')\leqslant|\mathcal{C}|^{-2\alpha}\cdot\mathop{\mathbf{E}}_{c'\in_R\mathcal{C}}\mathbb{P}_{c|s}^2(c')=|\mathcal{C}|^{-2\alpha}\cdot\|\mathbb{P}_{c|s}\|_2^2.$$

Therefore, by Lemma 3.5.11, and the fact that $F\geqslant|\mathcal{C}|^{-1}$,

$$\|\mathbb{P}_f\|_2=\frac{\|f\|_2}{F}\leqslant\frac{|\mathcal{C}|^{-\alpha}\cdot\|\mathbb{P}_{c|s}\|_2}{|\mathcal{C}|^{-1}}\leqslant|\mathcal{C}|^{(1-\alpha)}\cdot 2r\cdot|\mathcal{C}|^{-(1-\delta/2)}=|\mathcal{C}|^{1-\alpha+\delta/2+\log_{|\mathcal{C}|}2r}\cdot|\mathcal{C}|^{-1}.$$

Since, for sufficiently large $|\mathcal{C}|$,

$$1-\alpha+\delta/2+\log_{|\mathcal{C}|}2r=2\delta+\delta/2+\log_{|\mathcal{C}|}2r\leqslant 3\delta=\delta'/2,$$

we have $\|\mathbb{P}_f\|_2 \leqslant |\mathcal{C}|^{-(1-\delta'/2)}$. So, definition of $\tau$ we have $\|M^{(j)} \cdot \mathbb{P}_f\|_2 \leqslant |\mathcal{X}|^{\tau_{M^{(j)}}(\delta')} \leqslant |\mathcal{X}|^{-2\gamma}$. Thus $\mathbf{E}_{x\in_R\mathcal{X}}[|(M^{(j)} \cdot \mathbb{P}_f)(x)|^2] = \|M^{(j)} \cdot \mathbb{P}_f\|_2^2 \leqslant |\mathcal{X}|^{-4\gamma}$. So, by Markov's inequality,

$$\mathbf{Pr}_{x\in_R\mathcal{X}}[|(M^{(j)} \cdot \mathbb{P}_f)(x)| \geqslant |\mathcal{X}|^{-\gamma}] = \mathbf{Pr}_{x\in_R\mathcal{X}}[|(M^{(j)} \cdot \mathbb{P}_f)(x)|^2 \geqslant |\mathcal{X}|^{-2\gamma}] \leqslant |\mathcal{X}|^{-2\gamma}.$$

By a union bound,

$$\mathbf{Pr}_{x\in_R\mathcal{X}}\left[\exists j \in [r-1],\ |(M^{(j)} \cdot \mathbb{P}_f)(x)| \geqslant |\mathcal{X}|^{-\gamma}\right] \leqslant (r-1) \cdot |\mathcal{X}|^{-2\gamma}.$$

Therefore, since we always have $|(M^{(j)} \cdot \mathbb{P}_f)(x)| \leqslant 1$,

$$\begin{aligned}
&\mathop{\mathbf{E}}_{x\in_R\mathcal{X}}[(1 + \sum_{j=1}^{r-1} |(M^{(j)} \cdot \mathbb{P}_f)(x)|^{\gamma \log_r |\mathcal{X}|}] \\
&\quad\leqslant \mathop{\mathbf{E}}_{x\in_R\mathcal{X}}\left[\mathbf{1}_{\forall j\in[r-1],\ |(M^{(j)}\cdot\mathbb{P}_f)(x)|\leqslant|\mathcal{X}|^{-\gamma}} \cdot (1 + (r-1)\cdot|\mathcal{X}|^{-\gamma})^{\gamma\log_r|\mathcal{X}|}\right] \\
&\qquad+ \mathop{\mathbf{E}}_{x\in_R\mathcal{X}}\left[\mathbf{1}_{\exists j\in[r-1],\ |(M^{(j)}\cdot\mathbb{P}_f)(x)|>|\mathcal{X}|^{-\gamma}} \cdot r^{\gamma\log_r|\mathcal{X}|}\right] \\
&\quad\leqslant (1 + (r-1)\cdot|\mathcal{X}|^{-\gamma})^{\gamma\log_r|\mathcal{X}|} + (r-1)\cdot|\mathcal{X}|^{-2\gamma}\cdot|\mathcal{X}|^{\gamma} \\
&\quad= (1 + (r-1)\cdot|\mathcal{X}|^{-\gamma})^{\gamma\log_r|\mathcal{X}|} + (r-1)\cdot|\mathcal{X}|^{-\gamma} \leqslant 1 + |\mathcal{X}|^{-\gamma/2}
\end{aligned}$$

for $\gamma \log_r |\mathcal{X}|$ sufficiently large. Therefore, the total factor increase over $\langle \mathbb{P}_{c|v}, \mathbb{P}_{c|s}\rangle^{\gamma\log_r|\mathcal{X}|}$ is at most $(1 + |\mathcal{X}|^{-\sigma})^{\gamma\log_r|\mathcal{X}|} \cdot (1 + |\mathcal{X}|^{-\gamma/2})$ where $\sigma = \min(\gamma,\delta)/2$. Therefore, for sufficiently large $|\mathcal{X}|$ this is at most $1 + |\mathcal{X}|^{-\min(\gamma,\delta)/4}$. Since $\beta \leqslant \min(\gamma,\delta)/8$ this is at most $1 + |\mathcal{X}|^{-2\beta}$ as required to prove Lemma 3.5.9.

### *3.6 An SDP Relaxation for Norm Amplification on the Positive Orthant*

For a matrix $M \in \mathbb{C}^{\mathcal{X}\times\mathcal{C}}$,

$$\tau_M(\delta) = \sup_{\substack{\mathbb{P}\in\Delta_{\mathcal{C}} \\ \|\mathbb{P}\|_2\leqslant 1/|\mathcal{C}|^{1-\delta/2}}} \log_{|\mathcal{X}|}(\|M\cdot\mathbb{P}\|_2).$$

That is, $\tau_M(\delta) = \frac{1}{2}\log_{|\mathcal{X}|} OPT_{M,\delta}$ where $OPT_{M,\delta}$ is the optimum of the following quadratic program:

$$\begin{aligned}
&\text{Maximize} \quad \|M \cdot \mathbb{P}\|_2^2 = \langle M \cdot \mathbb{P}, M \cdot \mathbb{P} \rangle, \\
&\text{subject to:} \\
&\qquad \sum_{i \in \mathcal{C}} \mathbb{P}_i = 1, \\
&\qquad \sum_{i \in \mathcal{C}} \mathbb{P}_i^2 \leqslant |\mathcal{C}|^{\delta - 1}, \\
&\qquad \mathbb{P}_i \geqslant 0 \qquad\qquad\qquad\qquad \text{for all } i \in \mathcal{C}.
\end{aligned} \tag{3.4}$$

Instead of attempting to solve (3.4), presumably a difficult quadratic program, we consider the following semidefinite program (SDP):

$$\begin{aligned}
&\text{Maximize} \quad \langle M^* M, U \rangle \cdot |\mathcal{C}|^2 / |\mathcal{X}| \\
&\text{subject to:} \\
&\qquad [V] \quad U \succeq 0, \\
&\qquad [w] \quad \sum_{i,j \in \mathcal{C}} U_{ij} = 1, \\
&\qquad [z] \quad \sum_{i \in \mathcal{C}} U_{ii} \leqslant |\mathcal{C}|^{\delta - 1}, \\
&\qquad\qquad U_{ij} \in \mathbb{R}, U_{ij} \geqslant 0 \qquad\qquad \text{for all } i, j \in \mathcal{C}.
\end{aligned} \tag{3.5}$$

Recall that $M^*$ is the conjugate transpose of $M$. Note that for any $\mathbb{P} \in \Delta_{\mathcal{C}}$ achieving the optimum value of (3.4) the positive semidefinite matrix $U = \mathbb{P} \cdot \mathbb{P}^\top$ has the same value in (3.5) (where the $|\mathcal{C}|^2/|\mathcal{X}|$ factor accounts for the difference in scaling factors based on the dimensions for the two expectation inner products), and hence (3.5) is an SDP relaxation of (3.4).

But this is not a standard SDP, since $M$ is over $\mathbb{C}$ and $M^*M$ might contain complex entries. In order to apply techniques on real matrices, we define $N : \mathcal{C} \times \mathcal{C} \to \mathbb{R}$ as $N(c, c') = Re(M^*M(c, c'))$, that is, $N$ is the real part of $M^*M$. Then we have the following

(real) program:

$$\begin{aligned}
&\text{Maximize} && \langle N, U\rangle \cdot |\mathcal{C}|^2/|\mathcal{X}| \\
&\text{subject to:} && \\
&\quad [V] && U \succeq 0, \\
&\quad [w] && \sum_{i,j\in\mathcal{C}} U_{ij} = 1, \\
&\quad [z] && \sum_{i\in\mathcal{C}} U_{ii} \leqslant |\mathcal{C}|^{\delta-1}, \\
&\quad && U_{ij} \geqslant 0 \qquad\qquad \text{for all } i,j \in \mathcal{C}.
\end{aligned} \tag{3.6}$$

The key observation is that (3.5) and (3.6) have the same optimal value. This is because for any $U \in \mathbb{R}^{\mathcal{C}\times\mathcal{C}}$,

$$|\mathcal{C}|^2\langle M^*M, U\rangle = \sum_{i,j}(M^*M)_{ij}\cdot U_{ij} = \sum_{ij} Re((M^*M)_{ij})\cdot U_{ij} + i\cdot\sum_{c,c'} Im((M^*M)_{ij})\cdot U_{ij}$$

Since $M^*M$ is a Hermitian matrix, we have $(M^*M)_{ij} = \overline{(M^*M)_{ji}}$. But $U$ is real symmetric, so we have $\sum_{i,j} Im((M^*M)_{ij})\cdot U_{ij} = 0$, namely

$$|\mathcal{C}|^2\langle M^*M, U\rangle = \sum_{i,j} Re((M^*M)_{ij})\cdot U_{ij} = |\mathcal{C}|^2\langle N, U\rangle$$

and we only need to consider the real parts. In order to upper bound the value of (3.6), we consider its dual program:

$$\begin{aligned}
&\text{Minimize} && w + z\cdot|\mathcal{C}|^{\delta-1} \\
&\text{subject to:} && \\
&\quad [U] && V \succeq 0, \\
&\quad [U_{ii}] && w + z \geqslant V_{ii} + N_{ii}/|\mathcal{X}|, \qquad \text{for all } i \in \mathcal{C} \\
&\quad [U_{ij}] && w \geqslant V_{ij} + N_{ij}/|\mathcal{X}|, \qquad \text{for all } i \neq j \in \mathcal{C} \\
&\quad && z \geqslant 0
\end{aligned} \tag{3.7}$$

or equivalently,

$$\begin{aligned} \text{Minimize} \quad & w + z \cdot |\mathcal{C}|^{\delta} \cdot |\mathcal{C}|^{-1} \\ \text{subject to:} \quad & \\ & V \succeq 0, \\ & zI + wJ \geqslant V + N/|\mathcal{X}|, \\ & z \geqslant 0. \end{aligned} \tag{3.8}$$

where $I$ is the identity matrix and $J$ is the all 1's matrix over $\mathcal{C} \times \mathcal{C}$.

Any dual solution of (3.8) yields an upper bound on the optimum of (3.5) and hence $OPT_{M,\delta}$ and $\tau_M(\delta)$. To simplify the complexity of analysis we restrict ourselves to considering semidefinite matrices $V$ that are suitably chosen Laplacian matrices. For any set $S$ in $\mathcal{C} \times \mathcal{C}$ and any $\alpha : S \to \mathbb{R}_+$ the Laplacian matrix associated with $S$ and $\alpha$ is defined by $L_{(S,\alpha)} := \sum_{(i,j)\in S} \alpha(i,j) L_{ij}$ where $L_{ij} = (e_i - e_j)(e_i - e_j)^\top$ for the standard basis $\{e_i\}_{i\in\mathcal{C}}$ . Intuitively, in the dual SDP (3.8), by adding matrix $V = L_{S,\alpha}$ for suitable $S$ and $\alpha$ depending on $M$ we can shift weight from the off-diagonal entries of $N$ to the diagonal where they can be covered by the $z + w$ entries on the diagonal rather than being covered by the $w$ values in the off-diagonal entries. This will be advantageous for us since the objective function has much smaller coefficient for $z$ which helps cover the diagonal entries than coefficient for $w$, which is all that covers the off-diagonal entries.

**Definition 3.6.1.** *Suppose that $N \in \mathbb{R}^{\mathcal{C}\times\mathcal{C}}$ is a symmetric matrix. For $\kappa \in \mathbb{R}_+$, define*

$$W_\kappa(N) = \max_{i\in\mathcal{C}} \sum_{j\in\mathcal{C}:\ N_{i,j}>\kappa} (N_{i,j} - \kappa).$$

The following lemma is the basis for our bounds on $\tau_M(\delta)$.

**Lemma 3.6.2.** *Let $\kappa \in \mathbb{R}_+$. Then*

$$OPT_{M,\delta} \leqslant (\kappa + W_\kappa(N) \cdot |\mathcal{C}|^{\delta-1})/|\mathcal{X}|.$$

*Proof.* For each off-diagonal entry of $N$ with $N(i,j) > \kappa$, include matrix $L_{ij}$ with coefficient $(N(i,j) - \kappa)/|\mathcal{X}|$ in the sum for the Laplacian $V$. By construction, the matrix $V + N/|\mathcal{X}|$

has off-diagonal entries at most $\kappa/|\mathcal{X}|$ and diagonal entries at most $(\kappa + W_\kappa(N))/|\mathcal{X}|$. The solution to (3.8) with $w = \kappa/|\mathcal{X}|$ and $z = W_\kappa(N)/|\mathcal{X}|$ is therefore feasible, which yields the bound as required. □

It may not be easy to bound $W_\kappa(N)$ directly, since the real part of $M^*M$ may not have good structure. Fortunately, we have the following measure:

**Definition 3.6.3.** *Let $M \in \mathbb{C}^{\mathcal{X}\times\mathcal{C}}$ be a complex matrix. For $\kappa \in \mathbb{R}_+$, define*

$$\tilde{W}_\kappa(M) = \max_{i\in\mathcal{C}} \sum_{j\in\mathcal{C}:\ |(M^*M)_{i,j}|>\kappa} (|(M^*M)_{i,j}| - \kappa)$$

.

**Proposition 3.6.4.** *Let $\kappa \in \mathbb{R}_+$. Then $W_\kappa(N) \leqslant \tilde{W}_\kappa(M)$*

*Proof.* Whenever $N_{i,j} > \kappa$, we have $|(M^*M)_{i,j}| \geqslant N_{i,j} > \kappa$. Moreover, this gives $|(M^*M)_{i,j}| - \kappa \geqslant N_{i,j} - \kappa$. Then the statement follows the two definitions.

□

For specific matrices $M$, we obtain the required bounds on $\tau_M(\delta) < 0$ for some $0 < \delta < 1$ by showing that we can set $\kappa = |\mathcal{X}|^\gamma$ for some $\gamma < 1$ and obtain that $W_\kappa(N)$ or $\tilde{W}_\kappa(M)$ is at most $\kappa \cdot |\mathcal{C}|^{\gamma'}$ for some $\gamma' < 1$.

### *3.7 Applications to Learning Polynomial Functions over Finite Fields*

In this section, we prove all the bounds on the norm amplification curves needed to obtain the lower bounds on learning polynomials discussed in Section 3.4. We use the strategy in Section 3.6 by studying the $W_\kappa$ function for matrices associated with learning polynomials over finite fields. We show that the values of this function are determined by the weight distribution and expected bias of these polynomials.

### 3.7.1 The Bias of $\mathbb{F}_2$ Polynomials and the Weight Distribution of Reed-Muller Codes

Let $d \geqslant 2$ be an integer. For any integer $n \geqslant d$, consider the learning problem for $\mathbb{F}_2$ polynomials in $n$ variables of degree at most $d$, That is $\mathcal{X} = \mathbb{F}_2^n$ and, expressing polynomials by their coefficients, we have $\mathcal{C} = \mathbb{F}_2^m$ where $m = \sum_{i=0}^{d} \binom{n}{d}$ and for $x \in \mathcal{X}$ and $c \in \mathcal{C}$, $L(x,c) = c(x) = \sum_{S:0\leqslant|S|\leqslant d} c_S \prod_{i\in S} x_i$ over $\mathbb{F}_2$.

Recall that since the range of $L$ is $\{0,1\}$, we have a $N = M^\top \cdot M$ where $M(x,c) = (-1)^{L(x,c)} = (-1)^{c(x)}$. Let $M_c$ denote the $c$-th column of $M$ where $c \in \{0,1\}^n$. Then $N_{cc'} = 2^n \cdot \langle M_c, M_{c'} \rangle$.

**Proposition 3.7.1.** *Let* $\mathbf{0} = 0^m$. *Then* $\langle M_c, M_{c'} \rangle = \langle M_{\mathbf{0}}, M_{c+c'} \rangle$.

*Proof.*

$$\begin{aligned}\langle M_c, M_{c'} \rangle &= \mathop{\mathbf{E}}_{x\in\mathbb{F}_2^n} M_c(x) M_{c'}(x) = \mathop{\mathbf{E}}_{x\in\mathbb{F}_2^n} (-1)^{c(x)}(-1)^{c'(x)} = \mathop{\mathbf{E}}_{x\in\mathbb{F}_2^n} (-1)^{c(x)+c'(x)} \\ &= \mathop{\mathbf{E}}_{x\in\mathbb{F}_2^n} (-1)^{(c+c')(x)} = \mathop{\mathbf{E}}_{x\in\mathbb{F}_2^n} M_{\mathbf{0}}(x) M_{c+c'}(x) = \langle M_{\mathbf{0}}, M_{c+c'} \rangle\end{aligned}$$

□

Since the mapping $c' \mapsto c + c'$ for $c \in \mathbb{F}_2^m$ is 1-1 on $\mathbb{F}_2^m$, every row of $N_c$ for $c \in \mathcal{C}$ contains the same multi-set of values. Therefore, in order to analyze the function $W_\kappa(N)$, we only need to examine the fixed row $N_{\mathbf{0}}$ of $N$, where each entry

$$N_{\mathbf{0}c} = \sum_{x\in\mathbb{F}_2^n} M(x,c) = \sum_{x\in\mathbb{F}_2^n} (-1)^{c(x)}.$$

For $c \in \mathcal{C}$, define $\text{weight}(c) = |\{x \in \mathbb{F}_2^n \ : \ c(x) = 1\}|$. By definition, for $c \in \mathbb{F}_2^m$, $N_{\mathbf{0}c} = \sum_{x\in\mathbb{F}_2^n} (-1)^{c(x)} = 2^n - 2 \cdot \text{weight}(c)$. Thus, understanding the function $W_\kappa(N)$ that we use to derive our bounds via Theorem 3.4.3 reduces to understanding the distribution of $\text{weight}(c)$ for $c \in \mathcal{C}$. In particular, our goal of showing that for some $\kappa$ for which $(\kappa + W_\kappa(N))/2^n$ is at most $2^{2\tau n}$ for some $\tau < 0$ follows by showing that the distribution of $\text{weight}(c)$ is tightly concentrated around $2^n/2$.

We can express this question in terms of the Reed-Muller error-correcting code $RM(d,n)$ over $\mathbb{F}_2$ (see, e.g. [16]).

**Definition 3.7.2.** *The Reed-Muller code $RM(d,n)$ over $\mathbb{F}_2$ is the set of vectors $\{G \cdot c \mid c \in \{0,1\}^m\}$ where $G$ is the $2^n \times m$ matrix for $m = \sum_{t=0}^{d} \binom{n}{t}$ over $\mathbb{F}_2$ with rows indexed by vectors $x \in \{0,1\}^n$ and columns indexed by subsets $S \subseteq [n]$ with $|S| \leqslant d$ given by $G(x,S) = \prod_{i \in S} x_i$.*

Evaluating weight$(c)$ for all $c \in \{0,1\}^m$ is that of understanding the distribution of Hamming weights of the vectors in $RM(d,n)$, a question with a long history.

**Quadratic polynomials over $\mathbb{F}_2$** For the special case that $d = 2$, Sloane and Berlekamp [123] derived an exact enumeration of the number of vectors of each weight in $RM(2,n)$.

**Proposition 3.7.3** ([123])**.** *The weight of every codeword of $RM(2,n)$ is of the form $2^{n-1} \pm 2^{n-i}$ for some integer $i$ with $1 \leqslant i \leqslant \lceil n/2 \rceil$ or precisely $2^{n-1}$ and the number of codewords of weight $2^{n-1} + 2^{n-i}$ or $2^{n-1} - 2^{n-i}$ is precisely*

$$2^{i(i+1)} \prod_{j=0}^{i-1} \frac{2^{n-2j}(2^{n-2j-1}-1)}{2^{2(j+1)}-1}.$$

(Though the original proof used other methods, a simpler alternative proof by McEliece [100] follows from a lemma of Dickson [49] giving a normal form theorem for quadratic polynomials over $\mathbb{F}_{2^t}$. We will use a similar approach when we analyze quadratic polynomials over $\mathbb{F}_{p^t}$.)

*Proof of Theorem 3.4.2 (a).* Let the threshold $\kappa = 2^{n-k}$ for some integer $k$ to be determined later. By Lemma 3.6.2 with $\mathcal{C} = \{0,1\}^m$, for (3.4), we have $OPT_{M,\delta} \leqslant (\kappa + W_\kappa(N) 2^{(\delta-1)m})/2^n$ where $N = M^\top \cdot M$. By definition for all $c \in \mathcal{C}$ we have $N_{\mathbf{0}c} = 2^n - 2 \cdot \text{weight}(c)$ and by Proposition 3.7.3, we know that if $2^n - 2 \cdot \text{weight}(c) > 0$ then it is $2^{n-i+1}$ for some $1 \leqslant i \leqslant \lceil n/2 \rceil$. Also by Proposition 3.7.3, the number, $\gamma_i$, of $c \in \mathcal{C}$ such that $N_{\mathbf{0}c} = 2^{n-i+1}$ is at most

$$2^{i(i+1)} \prod_{j=0}^{i-1} \frac{2^{n-2j}(2^{n-2j-1}-1)}{2^{2(j+1)}-1} \leqslant 2^{2(i-1)n}.$$

Therefore, by definition of $W_\kappa$ and Proposition 3.7.1, for any $c \in \mathcal{C}$ we have

$$\begin{aligned} W_\kappa(N) &\leqslant \sum_{c' \in \mathcal{C}: N_{cc'} > 2^{n-k}} N_{cc'} \\ &= \sum_{i=1}^{k} \gamma_i \cdot 2^{n-i+1} \\ &\leqslant \sum_{i=1}^{k} 2^{2(i-1)n} \cdot 2^{n-i+1} \\ &= \sum_{i=1}^{k} 2^{(2n-1)(i-1)+n} \\ &< 2^{(2n-1)k}. \end{aligned}$$

Thus for any $k$,

$$OPT_{M,\delta} \leqslant (2^{n-k} + 2^{(2n-1)k+(\delta-1)m})/2^n \leqslant 2^{-k} + 2^{(2n-1)k-(1-\delta)n(n+1)/2-n}.$$

The first term is larger for $k \leqslant (1-\delta)n/4 + (3-\delta)/4$ so to balance them as much as possible we choose $k = \lfloor (1-\delta)n/4 + (3-\delta)/4 \rfloor \geqslant (1-\delta)n/4 - (1+\delta)/4$. Hence $OPT_{M,\delta} \leqslant 2 \cdot 2^{-k} \leqslant 2^{-\frac{1-\delta}{4}n + \frac{5+\delta}{4}}$ Therefore, $\tau_M(\delta) = \frac{1}{2}\log_{2^n} OPT_{M,\delta} \leqslant -\frac{(1-\delta)}{8} + \frac{(5+\delta)}{8n}$ as required.

□

**Polynomials of degree $d > 2$ over $\mathbb{F}_2$** For the case that $d > 2$, the minimum distance, the smallest weight of a non-zero codeword, in $RM(d,n)$ is known to be $2^{n-d}$ but for $2 < d < n-2$, no exact enumeration of the weight distribution of the code $RM(d,n)$ is known. It was a longstanding problem even to approximate the number of codewords of different weights in $RM(d,n)$. Relatively recently, bounds on these weights (or more precisely the associated biases) that are good enough for our purposes were shown by Ben-Eliezer, Hod, and Lovett [15].

**Proposition 3.7.4.** *For $\varepsilon > 0$ there are constants $c_1, c_2$ with $0 < c_1, c_2 < 1$ such that if $p$ is a*

*uniformly random degree d polynomial over* $\mathbb{F}_2^n$ *and* $d \leqslant (1-\varepsilon)n$ *then*

$$\mathbf{Pr}[|\mathop{\mathbf{E}}_{x\in\{0,1\}^n}(-1)^{p(x)}| > 2^{-c_1 n/d}] \leqslant 2^{-c_2\sum_{i=0}^{d}\binom{n}{i}}.$$

From this form we can obtain the bound on the norm amplification curve of the associated matrix fairly directly.

*Proof of Theorem 3.4.2 (b).* Fix $\varepsilon > 0$ and let $0 < c_1, c_2 < 1$ be the constants depending on $\varepsilon$ from Proposition 3.7.4. Let $\delta = c_2/2$ so $0 < \delta < 1/2$. Let $M$ be the $2^n \times 2^m$ matrix associated with learning polynomials of degree at most $d$ over $\mathbb{F}_2$, let $N = M^\top \cdot M$ and Setting $\kappa = 2^{(1-c_1/d)n}$, by Proposition 3.7.4 at most $2^{(1-c_2)m}$ polynomials $p$ have entries $N_{0p}$ larger than $\kappa$. Each such entry has value at most $2^n$ so $W_\kappa(N) \leqslant 2^n \cdot 2^{(1-c_2)m}$. by Lemma 3.6.2 with $\mathcal{C} = \{0,1\}^m$ we have

$$OPT_{M,\delta} \leqslant (\kappa + W_\kappa(N)\cdot 2^{(\delta-1)m})/2^n \leqslant 2^{-c_1 n/d} + 2^{(\delta-c_2)m+1} \leqslant 2^{-c_1 n/d} + 2^{1-\delta m}$$

which is at most $2^{-c' n/d}$ for some constant $c' > 0$. Hence $\tau_M(\delta) \leqslant -c'/d$. □

### 3.7.2 *The Bias of* $\mathbb{F}_p$ *Polynomials for Odd Prime* $p$

Let $d \geqslant 1$ be an integer and $p$ be an odd prime. For any integer $n \geqslant d$, consider the learning problem for $\mathbb{F}_p$ polynomials in $n$ variables of degree at most $d$. Unlike the case over $\mathbb{F}_2$, the monomials are not necessarily multilinear but can have degree at most $p-1$ in each variable. Let $\mathcal{M}_p(d,n)$ be the set of monomials in $n$ variables of total degree at most $d$ and degree at most $p-1$ in each variable. That is $\mathcal{X} = \mathbb{F}_p^n$ and, expressing polynomials by their coefficients, we have $\mathcal{C} = \mathbb{F}_p^m$ where $m = |\mathcal{M}_p(d,n)|$ is the number of monomials of total degree at most $d$ and degree at most $p-1$ in each variable. As in the case of $\mathbb{F}_2$, $m$ is the dimension of a Reed-Muller code $RM_p(d,n)$ over $\mathbb{F}_p$, and for $x \in \mathcal{X}$ and $c \in \mathcal{C}$, $L(x,c) = c(x) \in \mathbb{F}_p$. For $d \geqslant p$ there is no convenient closed form known for $|\mathcal{M}_p(d,n)|$ but the following is known:

**Proposition 3.7.5.** *For $d < p$, $|\mathcal{M}_p(d,n)| = \binom{n+d}{d}$ and for $2 < p \leqslant d \leqslant n$, $\sum_{i=0}^{d}\binom{n}{d} \leqslant |\mathcal{M}_p(d,n)| \leqslant \binom{n+d}{d}$.*

Since $p > 2$, the learning problem for $\mathbb{F}_p$ polynomials is governed by $p-1$ complex matrices $M^{(1)},\ldots,M^{(p-1)}$ where $M^{(j)}(x,c) = \omega^{j\cdot c(x)}$ and $\omega = e^{2\pi i/p}$. We need to bound the norm amplification curves of all these matrices. We will relate these curves to the values of $\mathrm{bias}_j(c)$ for $j \in \{1,\ldots,p-1\}$ and $c \in \mathcal{C}$, where

$$\mathrm{bias}_j(c) = \mathop{\mathbf{E}}_{x\in_R\mathcal{X}} \omega^{j\cdot c(x)}.$$

Fix an arbitrary $j^* \in \{1,\ldots,p-1\}$, For $N = (M^{(j^*)})^* \cdot M^{(j^*)}$, the $(c,c')$ entry of $N$ is $p^n\langle M_c^{(j^*)}, M_{c'}^{(j^*)}\rangle$ where $\langle\cdot,\cdot\rangle$ is the complex inner product.

**Proposition 3.7.6.** *Let $\mathbf{0} = 0^m$. Then for $c,c' \in \mathcal{C}$, $\langle M_c^{(j^*)}, M_{c'}^{(j^*)}\rangle = \langle M_{\mathbf{0}}^{(j^*)}, M_{c'-c}^{(j^*)}\rangle$.*

*Proof.*

$$\begin{aligned}\langle M_c^{(j^*)}, M_{c'}^{(j^*)}\rangle &= \mathop{\mathbf{E}}_{x\in\mathbb{F}_p^n} \overline{M_c^{(j^*)}(x)} M_{c'}^{(j^*)}(x) = \mathop{\mathbf{E}}_{x\in\mathbb{F}_p^n} \omega^{-j^*\cdot c(x)}\omega^{j^*\cdot c'(x)} = \mathop{\mathbf{E}}_{x\in\mathbb{F}_p^n} \omega^{-j^*\cdot c(x)+j^*\cdot c'(x)} \\ &= \mathop{\mathbf{E}}_{x\in\mathbb{F}_p^n} \omega^{j^*(c'-c)(x)} = \mathop{\mathbf{E}}_{x\in\mathbb{F}_p^n} M_{\mathbf{0}}^{(j^*)}(x) M_{c'-c}^{(j^*)}(x) = \langle M_{\mathbf{0}}^{(j^*)}, M_{c'-c}^{(j^*)}\rangle\end{aligned}$$

□

Since the mapping $c' \mapsto c'-c$ for $c \in \mathbb{F}_p^m$ is 1-1 on $\mathbb{F}_p^m$, every row of $N_c$ for $c \in \mathcal{C}$ contains the same multi-set of values. Therefore, in order to analyze the function $\tilde{W}_\kappa(N)$, we only need to examine the fixed row $N_{\mathbf{0}}$ of $N$. where each entry

$$N_{\mathbf{0}c} = \sum_{x\in\mathbb{F}_p^n} \omega^{j^*\cdot c(x)} = p^n \cdot \mathrm{bias}_{j^*}(c).$$

Therefore we have shown the following:

**Lemma 3.7.7.** *Let $j^* \in \{1,\ldots,p-1\}$. For every $v \in \mathbb{C}$, the number of entries in each row of $N = (M^{(j^*)})^* \cdot M^{(j^*)}$ equal to $v$ is precisely the number of polynomials $c \in \mathcal{C}$ such that $p^n \cdot \mathrm{bias}_{j^*}(c) = v$.*

Therefore, to bound $\tilde{W}_\kappa(N)$ it suffices to bound the numbers of polynomials $c \in \mathcal{C}$ such that $|\mathrm{bias}_{j^*}(c)|$ is large.

**Affine Functions over $\mathbb{F}_p$** For $d = 1$, an $c \in \mathcal{C} = \mathbb{F}_p^{n+1}$ yields the function $c(x) = c_0 + \sum_{i=1}^{n} c_i x_i$. Unless $c_1 = \cdots = c_n = 0$, for every $k \in \mathbb{F}_p$ we have exactly $p^{n-1}$ values $x \in \mathbb{F}_p^n$ for which $c(x) = k$ and hence $\mathrm{bias}_{j^*}(c) = 0$. For each of the remaining $p$ inputs with $c_1 = \cdots = c_n = 0$ and different values for $c_0$, we get $\mathrm{bias}_{j^*}(c) = \omega^{j^* \cdot c_0}$ and hence $|\mathrm{bias}_{j^*}(c)| = 1$. In this case we choose $\kappa = 0$ and observe that $\tilde{W}_0(N) = p^{n+1}$. Therefore for any $\delta$ with $0 \leqslant \delta \leqslant 1$, we have

$$OPT_{M^{(j^*)},\delta} \leqslant p^{n+1}|\mathcal{C}|^{\delta-1}/|\mathcal{X}| = p^{1+(\delta-1)(n+1)} = (p^n)^{-(1-\delta)+\delta/n},$$

so $\tau_{M^{(j^*)}}(\delta) = \frac{1}{2}\log_{|\mathcal{X}|} OPT_{M^{(j^*)},\delta} \leqslant -\frac{1-\delta}{2} + \frac{\delta}{2n}$. This proves Theorem 3.4.4 (a). If we only took linear functions instead of affine functions, all non-zero $x$ would be balanced and the term $\frac{\delta}{2n}$ would not appear. (This is the analog of the parity learning bound for higher moduli.) This proves Theorem 3.4.4 (b).

**Quadratic Polynomials over $\mathbb{F}_p$**

**Lemma 3.7.8.** *Let $p$ be an odd prime and $n \geqslant 2$ be an integer. Let $\mathcal{C}$ be the set of quadratic polynomials over $\mathcal{X} = \mathbb{F}_p^n$. Then for $j^* \in \{1, \ldots, p-1\}$,*

1. *For any $c \in \mathcal{C}$, $\mathrm{bias}_{j^*}(c) = 0$ or $|\mathrm{bias}_{j^*}(c)| \in \{p^{-n/2}, p^{(n-1)/2}, \cdots, p^{-1/2}, 1\}$.*

2. *For $0 \leqslant k \leqslant n$ the number of $c \in \mathcal{C}$ such that $|\mathrm{bias}_{j^*}(c)| = p^{-k/2}$ is less than $p^{kn+2k+1}$.*

To prove Lemma 3.7.8 we start with the following structure lemma for quadratic polynomials over fields of odd characteristic. This lemma is an easier analog of Dickson's Lemma for characteristic 2 [49] and is well known but we include a proof for completeness.

**Lemma 3.7.9.** *Let $p$ be an odd prime and integer $t \geqslant 1$. For every quadratic polynomial $q$ over $\mathbb{F}_{p^t}$ in variables $z = (z_1, \ldots, z_n)$, there is an invertible affine transformation $T$ over $\mathbb{F}_{p^t}$ such that for $z' = T(z)$, there is a unique $k \leqslant n$, and $(\alpha_1, \ldots, \alpha_k) \in \{1, \cdots, p-1\}^k$, and an affine form $\ell$*

*over* $\mathbb{F}_{p^t}$ *in* $n-k$ *variables such that:*

$$q(z) = \sum_{i=1}^{k} \alpha_i z_i'^2 + \ell(z'_{k+1}, \cdots, z'_n)$$

*Proof.* We show this by induction on $n$. The statement is clearly true when $n = 0$. Assume that this is true for any polynomial in $n-1$ variables. We have several cases when $q$ has $n$ variables:

CASE 1: $q$ is affine: Then the statement is true with $k = 0$.

CASE 2: $q$ contains some square term $b_i \cdot z_i^2$: In this case we can write $q$ as $b_i \cdot z_i^2 + \ell_i \cdot z_i + q'$, where $\ell_i$ is affine, $q'$ is a quadratic polynomial, and neither of them involves $z_i$. Then we can define

$$z_i' = z_i + 2^{-1} b_i^{-1} \cdot \ell_i$$

since $b_i^{-1}$ and $2^{-1}$ are defined in field $\mathbb{F}_{p^t}$ because $b_i \neq 0$ and the characteristic $p$ is odd. Also define $q'' = q' - 2^{-2} b_i^{-1} \ell_i^2$. Thus

$$\begin{aligned}
&b_i (z_i')^2 + q'' \\
&= b_i (z_i')^2 + q' - 2^{-2} b_i^{-1} \ell_i^2 \\
&= b_i (z_i + 2^{-1} b_i^{-1} \cdot \ell_i)^2 + q' - 2^{-2} b_i^{-1} \ell_i^2 \\
&= b_i (z_i^2 + b_i^{-1} \ell_i \cdot z_i + 2^{-2} b_i^{-2} \ell_i^2) + q' - 2^{-2} b_i^{-1} \ell_i^2 \\
&= b_i \cdot z_i^2 + \ell_i \cdot z_i + q' = q.
\end{aligned}$$

Define $T_i$ to be the map which sets $z_j' = z_j$ for $j \neq i$ and replaces $z_i$ with $z_i'$ according to the above formula. Clearly by the definition of $z_i'$, $T_i$ is an affine map; moreover, it is invertible, with $T_i^{-1}$ setting $z_i = z_i' - 2^{-1} b_i^{-1} \cdot \ell_i$ and leaving all other $z_j$ for $j \neq i$ unchanged. By definition, $q''$ is a quadratic form defined only on the $m-1$ variables $z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_m$, a property inherited from $q'$ and $\ell_i$. Let $P_{in}$ be the permutation that swaps positions $i$ and $n$ and leaves the rest alone and define $q''' = P_{in}(q'')$.

We now can apply the inductive hypothesis to $q'''$ and derive that there is an invertible affine mapping $T'$ on the $n-1$ variables (excluding $z_i$) and some $k'$ together with

constants $a'_1, \ldots, a'_{k'} \in \mathbb{F}_p^*$, yielding variables $z''_1, \ldots, z''_{n-1}$ as affine functions of the previous values such that

$$q''' = \sum_{j=1}^{k'} \alpha_i z_j''^2 + \ell''(z''_{k'+1}, \ldots, z''_{n-1}).$$

We can extend $T'$ to an affine transformation $T''$ on $n$ variables by keeping the $n$-th variable unchanged.

Finally, define $k = k' + 1$, $\alpha_k = b_i$ and the invertible affine transformation, $T = P_{nk} \circ T'' \circ P_{in} \circ T_i$ where $P_{nk}$ is the permutation that swaps positions $k$ and $n$. Then $T(z) = (z''_1, \ldots, z''_{k-1}, z'_i, z''_{k+1}, \ldots, z''_{n-1}, z''_k)$.

$$T(q) = \sum_{j=1}^{k-1} \alpha_i z_j''^2 + \alpha_k (z'_i)^2 + \ell''(z''_{k+1}, \ldots, z''_{n-1}, z''_k)$$

which is of the required form.

CASE 3: $q$ has no squared terms and is not affine. Then $q$ must contain some cross term $b_{ij} \cdot z_i z_j$ for $i \neq j$. Here we can use the identity

$$z_i z_j = 2^{-2} \cdot ((z_i + z_j)^2 - (z_i - z_j)^2)$$

and let $S_{ij}$ be the affine mapping that leaves all other variables unchanged and assigns $z'_i = 2^{-1}(z_i + z_j)$ and $z'_j = 2^{-1}(z_i - z_j)$ which exists since 2 is invertible over $\mathbb{F}_{p^t}$. $S_{ij}$ is clearly invertible since $z_i = z'_i + z'_j$ and $z_j = z'_i - z'_j$. Hency, for $z' = S_{ij}(z)$, we have $q(z) = q_{ij}(z')$ for some quadratic $q_{ij}$ that has two squared terms $(z'_i)^2$ and $(z'_j)^2$ and hence is covered by Case 2 above. Let $T_2$ be the resulting affine transformation derived for $q_{ij}$. It follows that $T = T_2 \circ S_{ij}$ is the required transformation for $q$. □

Lemma 3.7.9 provides a clean way of studying the bias of quadratic polynomials. For any invertible affine mapping $T$ on $\mathbb{F}_p^n$, for $c' \in \mathcal{C}$ and $c(z) = c'(T(z))$, we have $c \in \mathcal{C}$ and

$$\mathrm{bias}_{j^*}(c) = \mathop{\mathbf{E}}_{x \in \mathbb{F}_p^n} \omega^{j^* \cdot c(x)} = \mathop{\mathbf{E}}_{x \in \mathbb{F}_p^n} \omega^{j^* \cdot c'(T(x))} = \mathop{\mathbf{E}}_{b \in \mathbb{F}_p^n} \omega^{j^* c'(b)} = \mathrm{bias}_{j^*}(c')$$

since $T$ is a bijection on $\mathbb{F}_p^n$.

We therefore first analyze the polynomials of the normal form in Lemma 3.7.9. Let $c'(z) = \sum_{i=1}^{k} \alpha_i z_i^2 + \ell(z_{k+1}, \ldots, z_n)$ where each $\alpha_1, \ldots, \alpha_k \neq 0$. Write $\ell = \alpha_0 + \sum_{i=k+1}^{n} \alpha_i z_i$. If there is any $j$ with $k+1 \leqslant j \leqslant n$ such that $\alpha_j \neq 0$ then then $\mathrm{bias}_{j^*}(c') = 0$ just as in the affine case. Therefore it remains to consider

$$c'(z) = \sum_{i=1}^{k} \alpha_i z_i^2 + \alpha_0 \quad \text{for } \alpha_1, \ldots, \alpha_k \in \mathbb{F}_p^*,\ \alpha_0 \in \mathbb{F}_p. \tag{3.9}$$

Observe that the number of such $c'$ is $(p-1)^k p < p^{k+1}$. Furthermore,

$$p^n \cdot |\mathrm{bias}_{j^*}(c')| = |\sum_{x \in \mathbb{F}_p^n} \omega^{j^* \cdot \sum_{i=1}^{k} \alpha_i x_i^2 + \alpha_0}| = p^{n-k} \cdot \prod_{i=1}^{k} |\sum_{x_i=0}^{p-1} \omega^{j^* \cdot \alpha_i x_i^2}|$$

The term $\sum_{x_i=0}^{p-1} \omega^{j^* \cdot \alpha_i x_i^2}$ in the product is called a *quadratic Gauss sum* and has been studied previously. For our purpose, we need the following result:

**Proposition 3.7.10** (Proposition 6.3.2 in [70])**.** *Let $p$ be an odd prime. For $\gamma \in \{1, \cdots, p-1\}$,*

$$|\sum_{j=0}^{p-1} \omega^{\gamma j^2}| = \sqrt{p}.$$

Therefore setting $\gamma = \alpha_i \cdot j^*$ for the $i$-th term, we have $|\mathrm{bias}_{j^*}(c')| = p^{-k/2}$. We now put things together to prove Lemma 3.7.8.

*Proof of Lemma 3.7.8.* By Lemma 3.7.9, since $\mathrm{bias}_{j^*}$ is preserved under invertible linear transformations $T$ of the inputs, it follows that every polynomial $c$ such that $\mathrm{bias}_{j^*}(c) \neq 0$ must have $|\mathrm{bias}_{j^*}(c)| = p^{-k/2}$ for some non-negative integer $k \leqslant n$. Moreover, the number of polynomials $c$ whose normal form $y$ of the form (3.9) is at most the number of affine transformations that define $z'_1, \ldots, z'_k$ in terms of $z_1, \ldots, z_k$ which is $(p^{n+1})^k$ since there are precisely $p^{n+1}$ affine functions on $\mathbb{F}_p^n$. Therefore the total number of $c$ such that $\mathrm{bias}_{j^*}(c) = p^{-k/2}$ is less than $p^{(n+1))k} \cdot p^{k+1} = p^{nk+2k+1}$. □

Now we can use Proposition 3.7.8 to prove part (c) of Theorem 3.4.4.

*Proof of Theorem 3.4.4 (c).* Proposition 3.7.8 implies that for $N = (M^{(j^*)}) * \cdot M^{(j^*)}$,

$$W_{p^{n-k/2}}(N) \leqslant \tilde{W}_{p^{n-k/2}}(N) \leqslant \sum_{t=0}^{k-1} p^{n-t/2} \cdot p^{tn+2t+1} = p^{n+1} \cdot \frac{p^{kn+3k/2}-1}{p^{n+3/2}-1} \leqslant p^{kn+3k/2}$$

Therefore if we set $k = \lfloor \frac{1-\delta}{2} n \rfloor \geqslant \frac{1-\delta}{2} n - 1$, then by Lemma 3.6.2, since $|\mathcal{C}| = p^{\binom{n+2}{2}}$ we have

$$OPT_{M^{(j^*)},\delta} \leqslant p^{-k} + p^{-n} \cdot p^{kn+3k/2+(\delta-1)\binom{n+2}{2}} \leqslant 2p^{-k} \leqslant p^{-\frac{1-\delta}{2}n+2}$$

Therefore,

$$\tau_{M^{(j^*)}}(\delta) = \frac{1}{2} \log_{p^n} OPT_{M^{(j^*)},\delta} \leqslant \frac{1-\delta}{4} + \frac{1}{n}$$

Since $j^*$ was an arbitrary fixed element of $\{1, \ldots, p-1\}$, the theorem follows. □

**Polynomials of degree $d > 2$ over $\mathbb{F}_p$** Similar to the $\mathbb{F}_2$ case, we need to understand the weight distribution of Reed-Muller codes over $\mathbb{F}_p$. In Chapter 4, we give the following estimate which is the analogue for odd prime fields of the bounds of Ben-Eliezer, Hod, and Lovett [15].

**Proposition 3.7.11** (Theorem 4.1.1)**.** *For $0 < \varepsilon < 1/2$, for all $j \in \mathbb{F}_p^*$, there are constants $c_1, c_2$ depending on $\varepsilon$ with $0 < c_1, c_2 < 1$ such that if $f$ is a uniformly random degree $d$ polynomial over $\mathbb{F}_p^n$ and $d \leqslant \varepsilon n$ then*

$$\mathbf{Pr}[|\mathrm{bias}_{j^*}(f)| > p^{-c_1 n/d}] \leqslant p^{-c_2 m}.$$

From this form we can obtain the bound on the norm amplification curve of the associated matrix fairly directly, and complete our proof of Theorem 3.4.4.

*Proof of Theorem 3.4.4 (d).* Fix $\varepsilon > 0$, $j \in \mathbb{F}_p^*$, and let $0 < c_1, c_2 < 1$ be the constants depending on $\varepsilon$ from Proposition 3.7.4. Let $\delta = c_2/2$ so $0 < \delta < 1/2$. For $N = (M^{(j^*)})^* \cdot M^{(j^*)}$, when we set $\kappa = p^{(1-c_1/d)n}$, Proposition 3.7.11 implies that at most $p^{(1-c_2)m}$ polynomials $f$ satisfy $|N_{0f}| > \kappa$. The norm of each entry is at most $p^n$ so $\tilde{W}_\kappa(N) \leqslant p^n \cdot p^{(1-c_2)m}$. by Lemma 3.6.2 with $\mathcal{C} = \mathbb{F}_p^m$ we have

$$OPT_{M,\delta} \leqslant (\kappa + W_\kappa(N) \cdot p^{(\delta-1)m})/p^n \leqslant p^{-c_1 n/d} + p^{(\delta-c_2)m+1} \leqslant p^{-c_1 n/d} + p^{1-\delta m}$$

which is at most $p^{-c'n/d}$ for some constant $c' > 0$. Hence $\tau_{M^{(j^*)}}(\delta) \leqslant -c'/d$. $\square$

# Chapter 4

# ON THE BIAS OF REED-MULLER CODES OVER ODD PRIME FIELDS

## *4.1 Introduction*

Reed-Muller codes are among the oldest error correcting codes, first introduced by Muller [104] and Reed [116] in the 1950s. These codes were initially defined in terms of bounded-degree multivariate polynomials over $\mathbb{F}_2$ but the same definition can be applied over any finite field. To be more precise, the $(d, n)$ Reed-Muller code over finite field $\mathbb{F}$, denoted $RM_{\mathbb{F}}(d, n)$, takes the message as the coefficients of some $n$-variate polynomial of degree at most $d$ over $\mathbb{F}$, and the encoding is simply the evaluation of that polynomial over all possible inputs chosen from $\mathbb{F}^n$.

A function $f : \mathbb{F}^n \to \mathbb{F}$ is *balanced* if elements of $\mathbb{F}$ occur an equal number of times as an output of $f$. The bias of a function $f$ with co-domain $\mathbb{F}$ is a measure of the fractional deviation of $f$ from being balanced. Since each codeword in a Reed-Muller code is the evaluation of a (polynomial) function over all elements of its domain, the definition of bias directly applies to the codewords of a Reed-Muller code.

Some elements of a Reed-Muller code are very far from balanced (for example the 0 polynomial yields the all-0 codeword, and the codeword for the polynomial $1 + x_1x_2$ has value 1 much more frequently than average) but since, as we might expect, randomly-chosen polynomials behave somewhat like randomly-chosen functions, most codewords are close to being balanced. We quantify that statement and show that for all prime fields, only an exponentially small fraction of Reed-Muller codewords (equivalently, an exponentially small fraction of polynomials of bounded degree) have as much an exponentially small deviation from perfect balance. That is, at most an exponentially small frac-

tion of polynomials have more than an exponentially small bias. Such a result is already known for the case of $\mathbb{F}_2$ [15] so we will only need to prove the statement for odd prime fields.

We now define bias formally and discuss its applications. In the case that $f : \mathbb{F}_2^n \to \mathbb{F}_2$, the *bias* of $f$,

$$\text{bias}(f) := \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)} = \mathbf{Pr}_{x \in_R \mathbb{F}_2^n}[f(x) = 0] - \mathbf{Pr}_{x \in_R \mathbb{F}_2^n}[f(x) = 1].$$

More generally, for $p$ a prime, $\omega = e^{2\pi i/p}$, and $j \in \mathbb{F}_p^*$, we define the *j-th order bias* of $f : \mathbb{F}_p^n \to \mathbb{F}_p$ as

$$\text{bias}_j(f) := \frac{1}{p^n} \sum_{x \in \mathbb{F}_p^n} \omega^{j \cdot f(x)}.$$

Prior uses of bias over these larger co-domains often focus only on the case of a single $j$ (e.g., [18, 67]) since they consider structural implications of bias. However, the use of different values of $j$ is essential for the applications of bias to bounding the imbalance of functions and codewords since, for $p > 3$, one can have functions with 1st-order bias 0 that are very far from balanced. It turns out that it is necessary and sufficient to bound $|\text{bias}_j(f)|$ for all $j \in \mathbb{F}_p^*$ (or, equivalently, all integers $j$ with $1 \leqslant j \leqslant (p-1)/2$ since $|\text{bias}_j(f)| = |\text{bias}_{-j}(f)|$) in order to bound the imbalance: A standard exponential summation argument (e.g., Proposition 3.2.1)shows that for every $b \in \mathbb{F}_p$,

$$\left| \mathbf{Pr}_{x \in_R \mathbb{F}_p^n}[f(x) = b] - \frac{1}{p} \right| \leqslant \max_{j \in \mathbb{F}_p^*} |\text{bias}_j(f)|.$$

For Reed-Muller codes, the bias of a codeword exactly determines its fraction (number) of non-zero entries, which is called the *weight* of the codeword. (In the case of $\mathbb{F}_2$ the bias is determined by the weight but that is not true for $\mathbb{F}_p$ for odd prime $p$, where the *generalized* weight [69] is required.) The distribution of weights of codewords in Reed-Muller codes over $\mathbb{F}_2$ plays a critical role in many applications in coding theory and in many other applications in theoretical computer science. As a consequence, the weight distribution of Reed-Muller codes over $\mathbb{F}_2$ has been the subject of considerable study. For

degrees $d = 1$ and $d = 2$, the exact weight distribution (and hence the distribution of the bias) for $RM_{\mathbb{F}_2}(2, n)$ has been known for roughly 50 years [123, 100]. For other degrees, precise bounds are only known for weights up to 2.5 times the minimum distance of such codes [74, 75] but this is very far from the balanced regime.

For general constant degrees, Kaufman, Lovett and Porat [76] give a bound on the weight distribution for Reed-Muller codes over $\mathbb{F}_2$, and Abbe, Shpilka, and Wigderson [1] generalize the result to linear degrees. These results yield tail bounds for the number of codewords with bias approaching 0 and, using the cases for arbitrarily small constant bias, imply good bounds for list-decoding algorithms [61, 76].

Ben-Eliezer, Hod, and Lovett [15] proved sharper bounds showing that the fraction of codewords with more than exponentially small bias (of the form $2^{-c_1 n/d)}$ for constant $c_1 > 0$) is at most $2^{-c_2 m} = |RM_{\mathbb{F}_2}(d, n)|^{-c_2}$ for constant $c_2 > 0$ where $m = \log_2 |RM_{\mathbb{F}_2}(d, n)|$ is the dimension of the code. (For $d < n/2$ they also showed that this fraction of codewords is tight by exhibiting a set of codewords in $RM_{\mathbb{F}_2}(d, n)$ of size $|RM_{\mathbb{F}_2}(d, n)|^{c_3}$ for $c_3 > 0$ that has such a bias.) This bound was used by [12, 11, 56] to show that learning bounded degree polynomials over $\mathbb{F}_2$ from their evaluations with success probability $2^{-o(n)}$ requires space $\Omega(nm/d)$ or time $2^{\Omega(n/d)}$.

**Our Results** We generalize the results of Ben-Eliezer, Hod, and Lovett [15] to show that only an exponentially small fraction of polynomials over prime fields can have non-negligible bias. Formally speaking, let $\mathcal{P}_p(d, n)$ denote the set of polynomials of degree at most $d$ in $n$ variables over $\mathbb{F}_p$, and let $\mathcal{M}_p(d, n)$ denote the set of monic monomials of degree at most $d$ in $n$ variables. (The Reed-Muller code $RM_{\mathbb{F}_p}(d, n)$ has dimension $|\mathcal{M}_p(d, n)|$ and satisfies $|RM_{\mathbb{F}_p}(d, n)| = |\mathcal{P}_p(d, n)|$.)

Our main result is the following theorem:

**Theorem 4.1.1.** *For any $0 < \delta < 1/2$ there are constants $c_1, c_2 > 0$ depending on $\delta$ such that*

*for any odd prime $p$, for all integers $d \leqslant \delta n$ and all $j \in \mathbb{F}_p^*$, we have*

$$\mathbf{Pr}_{f \in_R \mathcal{P}_p(d,n)}[|\mathrm{bias}_j(f)| > p^{-c_1 n/d}] \leqslant p^{-c_2 |\mathcal{M}_p(d,n)|}.$$

In Chapter 3, we show that this theorem can be plugged into a general theorem on learning finite functions to obtain that any algorithm that learns polynomials over $\mathbb{F}_p$ of degree at most $d$ with probability at least $p^{-O(n)}$ from their evaluations on random inputs either requires time $p^{\Omega(n/d)}$ or space $\Omega(n \cdot |\mathcal{M}_p(d,n)|/d \cdot \log p)$. Similar results can be shown to follow by using the theorem above with the methods of [56]. For details, see Chapter 3.

The following corollary of Theorem 4.1.1 is also immediate:

**Corollary 4.1.2.** *For any $0 < \delta < 1/2$ there are constants $c_1, c_2 > 0$ such that for any odd prime $p$ and integers $d, n$ with $d \leqslant \delta n$, the number of codewords of $RM_{\mathbb{F}_p}(d,n)$ of weight at most $1 - 1/p - p^{-c_1 n/d}$ is at most $|RM_{\mathbb{F}_p}(d,n)|^{1-c_2}$.*

There is a limit to the amount that Theorem 4.1.1 can be improved, as shown by the following proposition:

**Proposition 4.1.3.** *For any $0 < \delta < 1/2$ there are constants $c' < 1$ and $c'' > 0$ depending on $\delta$ such that for all integers $d \leqslant \delta n$ and all $j \in \mathbb{F}_p^*$, we have*

$$\mathbf{Pr}_{f \in_R \mathcal{P}_p(d,n)}[|\mathrm{bias}_j(f)| > p^{-c'' n/d}] \geqslant p^{-c' |\mathcal{M}_p(d,n)|}.$$

As part of our proof of Theorem 4.1.1, we must prove the following tight bound on the rank of the evaluations of monomials of degree at most $d$ on sets of points. Alternatively this can be seen as the extremal dimension of the span of truncated Reed-Muller codes at sizes that are powers of the field size.

**Lemma 4.1.4.** *Let $S$ be a subset of $\mathbb{F}_p^n$ such that $|S| = p^r$. Then the dimension of the subspace spanned by $\{(q(x))_{q \in \mathcal{M}_p(d,n)} : x \in S\}$ is at least $|\mathcal{M}_p(d,r)|$.*

Though this is all that we require to prove Theorem 4.1.1, and can be shown to follow from [69, 13], it is a special case of a more general theorem, which we derive using [13], giving an exact extremal characterization of the dimension of the span of truncated Reed-Muller codes of all sizes[1] and generalizing a characterization for the case of $\mathbb{F}_2$ proved by Keevash and Sudakov [78].

**Theorem 4.1.5.** *Let $1 \leqslant m \leqslant p^r$ and let $n \geqslant r$. For $S \subseteq \mathbb{F}_p^n$ with $|S| = m$, the value of*

$$\dim\langle\{(q(x))_{q\in\mathcal{M}_p(d,n)} \ : \ x \in S\}\rangle$$

*is minimized when $S = S_m$, the set of $m$ lexicographically minimal vectors in $\mathbb{F}_p^r$.*

**Proof Overview** Our basic approach is a generalization of the high level outline of [15] to odd prime fields, though parts of the argument are substantially more complex:

We begin by using a moment method, showing that that $\mathbf{E}_{f\in_R\mathcal{P}_p(d,n)}[|\mathrm{bias}_j(f)|^t]$ is bounded for suitable $t$. Because we are dealing with odd prime fields rather than $\mathbb{F}_2$ we restrict ourselves to the case that $t$ is even. For bounding these high moments, we reduce the problem to lower bounding the rank of certain random matrices (Lemma 4.2.4). This is the place where we can apply Lemma 4.1.4 to prove the bound.

For the case of $\mathbb{F}_2$ handled in [15], a similar property to Lemma 4.2.4 (Lemma 4 in [15]), which follows from an extremal characterization of $\mathbb{F}_2$ polynomial evaluations by Keevash and Sudakov [78], was independently shown to follow more simply via an algorithmic construction that avoids consideration of any subset size that is not a power of 2. Unfortunately, this simpler algorithmic construction seems to break down completely for the case of odd prime fields.

We instead consider the duality between the rank of sub-matrix of the generating matrix of the truncated Reed-Muller codes, and the maximal number of common zeros for a given number of polynomials over the finite field. The latter problem has been extensively

[1] In a preliminary version of this chapter [10] we had a more complicated direct proof of this characterization.

studied in the context of generalized Hamming weights of Reed-Muller codes [69, 13]. Using the results of [13] on the generalized Hamming weights of Reed-Muller codes, we are able to characterize the desired rank property. We also derive an explicit recursive formula for the extremal rank, which may be of independent interest. This recursive formula is the analogue of the formula used by Keevash and Sudakov [78] to obtain their characterization over $\mathbb{F}_2$.

**Discussion and Related Work** Prior to our work, the main approach to analyzing the bias of polynomials over arbitrary prime fields has been to take a structural point of view. The general idea is to show that polynomials of large bias must have this bias because of some structural property. For polynomials of degree $d = 2$, a complete structural characterization has been known for more than a century ([49]). Green and Tao [63] initiated the modern study of the relationship between the bias and the structure of polynomials over finite fields. Kaufman, Lovett, and Porat [76] used this approach to obtain their bounds on bias over $\mathbb{F}_2$. Over general prime fields, Haramaty and Shpilka [67] gave sharper structural properties for polynomials of degrees $d = 3, 4$. In papers [18] for constant degree and [17] for large degree, Bhowmick and Lovett generalized the result of [76] to show that if a degree $d$ polynomial $f$ has large bias, then $f$ can be expressed as a function of a constant number of polynomials of degree at most $d - 1$. These bounds are sufficient to analyze the list-decoding properties of Reed-Muller codes. However, all of these structural results, except for the characterization of degree 2 polynomials, are too weak to obtain the bounds on sub-constant bias that we derive. Indeed, none is sufficient even to derive Corollary 4.1.2.

An open problem that remains from our work, as well as that of Ben-Eliezer, Hod, and Lovett [15] is whether the amount of the bias can be improved still further by removing the $1/d$ factor from the exponent in the bias in the statement of Theorem 4.1.1 for some range of values of $d$ growing with $n$. Though Proposition 4.1.3 (and its analogue in [15]) show that a large number of polynomials have bias $p^{-O(n/d)}$, we would need to extend

them to say that for *all* $c' > 0$ there is a $c'' > 0$ such that the conclusion of the proposition holds in order to rule out improving the bias in Theorem 4.1.1.

**Organization** The proof of Theorem 4.1.1, except for the proof of Lemma 4.1.4, is in Section 4.2. Section 4.2 also contains the proof of Proposition 4.1.3. In Section 4.3 we prove Theorem 4.1.5, which is a generalization of Lemma 4.1.4.

### *4.2 The bias of random polynomials over odd prime fields*

In this section we prove Theorem 4.1.1. To provide tail bounds on the bias, we first characterize its high moments, focusing on even moments to ensure that they are real-valued.

**Lemma 4.2.1.** *Let $p$ be an odd prime and $d \leqslant n$. For $t \in \mathbb{N}$, let $x^{(1)}, \cdots, x^{(t)}$ and $y^{(1)}, \cdots, y^{(t)}$ be chosen uniformly at random from $\mathbb{F}_p^n$. Then*

$$\mathop{\mathbf{E}}_{f \in_R \mathcal{P}_p(d,n)}[\ |\mathrm{bias}_j(f)|^{2t}\ ] = \mathbf{Pr}_{x^{(1)},\cdots,x^{(t)},y^{(1)},\cdots,y^{(t)}}[\ \forall q \in \mathcal{M}_p(d,n),\ \sum_{k=1}^{t} q(x^{(k)}) = \sum_{k=1}^{t} q(y^{(k)})\ ].$$

*Proof.* Note that $\overline{\mathrm{bias}_j(f)} = \overline{\mathbf{E}_x[\omega^{j\cdot f(x)}]} = \mathbf{E}_x[\overline{\omega^{j\cdot f(x)}}] = \mathbf{E}_x[\omega^{-j\cdot f(x)}] = \mathrm{bias}_{-j}(f)$, therefore $|\mathrm{bias}_j(f)|^2 = \mathrm{bias}_j(f) \cdot \mathrm{bias}_{-j}(f)$. So we have

$$\begin{aligned}
\mathop{\mathbf{E}}_{f \in_R \mathcal{P}_p(d,n)}[|\mathrm{bias}_j(f)|^{2t}] &= \mathop{\mathbf{E}}_{f \in_R \mathcal{P}_p(d,n)}[\mathrm{bias}_j(f)^t \cdot \mathrm{bias}_{-j}(f)^t] \\
&= \mathop{\mathbf{E}}_{f \in_R \mathcal{P}_p(d,n)}[\prod_{k=1}^{t} \mathop{\mathbf{E}}_{x^{(k)}}[\omega^{j\cdot f(x^{(k)})}] \cdot \prod_{k=1}^{t} \mathop{\mathbf{E}}_{y^{(k)}}[\omega^{-j\cdot f(y^{(k)})}]] \\
&= \mathop{\mathbf{E}}_{f \in_R \mathcal{P}_p(d,n)}[\mathop{\mathbf{E}}_{x^{(1)},\cdots,x^{(t)},y^{(1)},\cdots,y^{(t)}}[\omega^{j\cdot(\sum_{k=1}^{t} f(x^{(k)}) - \sum_{k=1}^{t} f(y^{(k)}))}]] \\
&= \mathop{\mathbf{E}}_{x^{(1)},\cdots,x^{(t)},y^{(1)},\cdots,y^{(t)}}[\mathop{\mathbf{E}}_{f \in_R \mathcal{P}_p(d,n)}[\omega^{j\cdot(\sum_{k=1}^{t} f(x^{(k)}) - \sum_{k=1}^{t} f(y^{(k)}))}]]
\end{aligned}$$

For each $q \in \mathcal{M}_p(d,n)$ let $f_q \in \mathbb{F}_p$ denote the coefficient of $q$ in $f$. We identify $f$ with its vector of coefficients $(f_q)_{q \in \mathcal{M}_p(d,n)}$ and choose $f$ uniformly by choosing the $f_q$ uniformly.

Therefore

$$\begin{aligned}
&\mathop{\mathbf{E}}_{f\in_R\mathcal{P}_p(d,n)}[|\mathrm{bias}_j(f)|^{2t}] \\
&= \mathop{\mathbf{E}}_{x^{(1)},\cdots,x^{(t)},y^{(1)},\cdots,y^{(t)}}\Big[\mathop{\mathbf{E}}_{f\in_R\mathcal{P}_p(d,n)}\big[\omega^{j\cdot(\sum_{q\in\mathcal{M}_p(d,n)} f_q\cdot(\sum_{k=1}^t q(x^{(k)})-\sum_{k=1}^t q(y^{(k)})))}\big]\Big] \\
&= \mathop{\mathbf{E}}_{x^{(1)},\cdots,x^{(t)},y^{(1)},\cdots,y^{(t)}}\Big[\prod_{q\in\mathcal{M}_p(d,n)}\mathop{\mathbf{E}}_{f_q\in_R\mathbb{F}_p}\big[\omega^{j\cdot f_q\cdot(\sum_{k=1}^t q(x^{(k)})-\sum_{k=1}^t q(y^{(k)}))}\big]\Big] \\
&= \mathop{\mathbf{E}}_{x^{(1)},\cdots,x^{(t)},y^{(1)},\cdots,y^{(t)}}\big[\mathbf{1}_{(\forall q\in\mathcal{M}_p(d,n),\ \sum_{k=1}^t q(x^{(k)})-\sum_{k=1}^t q(y^{(k)})=0)}\big] \\
&= \mathbf{Pr}_{x^{(1)},\cdots,x^{(t)},y^{(1)},\cdots,y^{(t)}}[\forall q\in\mathcal{M}_p(d,n),\ \sum_{k=1}^t q(x^{(k)}) = \sum_{k=1}^t q(y^{(k)})]
\end{aligned}$$

where the second equality follows since $\mathbf{E}_{a\in_R\mathbb{F}_p}[\omega^{j\cdot a\cdot b}] = 0$ for all $b\in\mathbb{F}_p^*$. □

Now let us look at the probability

$$\mathbf{Pr}_{x^{(1)},\cdots,x^{(t)},y^{(1)},\cdots,y^{(t)}}[\forall q\in\mathcal{M}_p(d,n),\ \sum_{k=1}^t q(x^{(k)}) = \sum_{k=1}^t q(y^{(k)})].$$

We view $y^{(1)},\cdots,y^{(t)}$ as arbitrary fixed values and we will upper bound this probability following the analysis of a similar probability in [15]. That is, we will upper bound the probability that this holds by considering a special subset $\mathcal{M}'\subseteq\mathcal{M}_p(d,n)$ that allows us to derive a linear system whose rank will bound the probability that the constraints indexed by $\mathcal{M}'$ all hold.

We divide $[n]$ arbitrarily into two disjoint parts $L$ and $R$ with $|L| = \lfloor\frac{n}{d}\rfloor$. $\mathcal{M}'\subseteq\mathcal{M}_p(d,n)$ consists of all monomials of degree are most $d$ that have degree 1 on $L$ and degree at most $d-1$ on $R$.

We use the following properties of the $|\mathcal{M}_p(d,n)|$, whose proof we defer to later, to show that $\mathcal{M}'$ contains a significant fraction of all monomials in $\mathcal{M}_p(d,n)$.

**Proposition 4.2.2.** *If $2\leqslant d\leqslant\delta n$ for some $0<\delta<1$ then*

*(a) there exists a constant $\gamma'=\gamma'(\delta)>0$ such that for sufficiently large $n$, if $n'\geqslant\max(d,(1-\frac{1}{d})n)$ then*

$$|\mathcal{M}_p(d,n')|\geqslant\gamma'|\mathcal{M}_p(d,n)|.$$

*(b) If $p \geqslant 3$ there exist constants $\rho_1, \rho_2 > 0$ that are independent of $\delta$ such that for sufficiently large $n$,*

$$\rho_1 |\mathcal{M}_p(d,n)| \leqslant \frac{n}{d} \cdot |\mathcal{M}_p(d-1,n)| \leqslant \rho_2 |\mathcal{M}_p(d,n)|.$$

**Corollary 4.2.3.** *Let $p \geqslant 3$. If $d \leqslant \delta n$ for some $0 < \delta < 1$, then there exists a constant $\gamma = \gamma(\delta) > 0$ such that for sufficiently large $n$,*

$$|\mathcal{M}'| = \lfloor \frac{n}{d} \rfloor \cdot |\mathcal{M}_p(d-1, n - \lfloor \frac{n}{d} \rfloor)| \geqslant \gamma \cdot |\mathcal{M}_p(d,n)|.$$

*Proof.* The equality follows immediately from the definition of $\mathcal{M}'$. Let $n' = n - \lfloor \frac{n}{d} \rfloor$. Then

$$\begin{aligned}
|\mathcal{M}'| &= \lfloor \frac{n}{d} \rfloor \cdot |\mathcal{M}_p(d-1,n')| \\
&\geqslant \frac{n'}{2d} |\mathcal{M}_p(d-1,n')| \qquad \text{since } d \leqslant n \\
&\geqslant \frac{\rho_1}{2} |\mathcal{M}_p(d,n')| \qquad \text{by Proposition 4.2.2(b)} \\
&\geqslant \frac{\rho_1 \gamma'}{2} |\mathcal{M}_p(d,n)| \qquad \text{by Proposition 4.2.2(a)}
\end{aligned}$$

and setting $\gamma = \rho_1 \gamma'/2$ yields the claim. □

Let $\mathcal{E}$ denote the event that $\sum_{k=1}^t q(x^{(k)}) = \sum_{k=1}^t q(y^{(k)})$ for all $q \in \mathcal{M}'$. To simply notation, since we think of $y^{(1)}, \ldots, y^{(k)}$ as fixed, for each $q \in \mathcal{M}'$ define $b_q \in \mathbb{F}_p$ by $b_q = \sum_{k=1}^t q(y^{(k)})$. Since any $q \in \mathcal{M}'$ is of the form $q = x_i \cdot q'$ for some $i \in L$ and $q'$ a monomial of degree at most $d-1$ on $R$, $\mathcal{E}$ requires that

$$b_q = \sum_{k=1}^t q(x^{(k)}) = \sum_{k=1}^t q'(x_R^{(k)}) \cdot x_i^{(k)}.$$

where for $x \in \mathbb{F}_p^n$, we write $x_R$ for $x$ restricted to the coordinates in $R$. We view these constraints as a system of linear equations over the set of variables $x_i^{(k)}$ for $k \in [t]$ and $i \in L$ whose coefficients are given by the values of $q'(x_R^{(k)})$ for $x_R^{(k)} \in \mathbb{F}_p^R$ for all $k \in [t]$. Observe that for different values of $i \in L$ we get separate and independent subsystems of equations with precisely the same coefficients but potentially different constant terms $b_q$

since $q$ depends on both $i$ and $q'$. Therefore the probability that $(x_i^{(k)})_{i\in L,k\in[t]}$ is a solution is the product of the probabilities for the individual choices of $i \in L$.

For each $\mathbf{x}_R = x_R^{(1)}, \ldots, x_R^{(t)}$, there is a $|\mathcal{M}_p(d-1,R)| \times t$ matrix $Q_{\mathbf{x}_R}$ for a system of linear equations on $(x_i^{(1)}, \ldots, x_i^{(t)})$ for each $i \in L$, having one constraint for each polynomial $q'$ of degree at most $d-1$ on $R$. Observe that $Q_{\mathbf{x}_R}(q',k) = q'(x_R^{(k)})$.

In particular, it follows that

$$\mathbf{Pr}_{x^{(1)},\cdots,x^{(t)},y^{(1)},\cdot,y^{(t)}}[\ \mathcal{E} \mid (x_R^{(1)},\cdots,x_R^{(t)}) = \mathbf{x}_R] \leqslant p^{-rank(Q_{\mathbf{x}_R})\cdot|L|}. \tag{4.1}$$

We now see that for almost all choices of $\mathbf{x}_R$, if $t$ is at least a constant factor larger than $|\mathcal{M}_p(d-1,|R|)|$ then the rank of $Q_{\mathbf{x}_R}$ is large. This follows by replacing $n$ by $|R|$, $d$ by $d-1$, $q'$ by $q$ and $\mathbf{x}$ by $\mathbf{x}_R$ in the following lemma.

**Lemma 4.2.4.** *For any $0 < \delta \leqslant 1/2$ there is a constant $\gamma = \gamma(\delta) > 0$ such that there exist constants $c > 0$ and $\eta > 1$ such that for $d = \lfloor \delta n \rfloor$ and $t \geqslant \eta|\mathcal{M}_p(d,n)|$, if $\mathbf{x} = x^{(1)},\cdots,x^{(t)}$ is chosen uniformly at random from $(\mathbb{F}_p^n)^t$, then the matrix $Q_{\mathbf{x}} \in \mathbb{F}_p^{\mathcal{M}_p(d,n)\times[t]}$ given by $Q_{\mathbf{x}}(q,k) = q(x^{(k)})$. then*

$$\mathbf{Pr}_{\mathbf{x}}[\ rank(Q_{\mathbf{x}}) \leqslant \gamma|\mathcal{M}_p(d,n)|\ ] \leqslant p^{-c|\mathcal{M}_p(d+1,n)|}.$$

We first show how to use Lemma 4.2.4 to prove Theorem 4.1.1.

*Proof of Theorem 4.1.1.* Let $0 < \delta \leqslant 1/2$, and set $\gamma > 0$ and $\eta > 1$ and $c > 0$ as in Lemma 4.2.4.

Let $t = \lceil \eta\mathcal{M}_p(d-1,n)\rceil$. We first bound the expected value of $|\mathrm{bias}_j(f)|^{2t}$. By Lemma 4.2.1 and the definition of event $\mathcal{E}$ we have

$$\mathop{\mathbf{E}}_{f\in_R\mathcal{P}_p(d,n)}[\ |\mathrm{bias}_j(f)|^{2t}\ ] = \mathbf{Pr}_{x^{(1)},\cdots,x^{(t)},y^{(1)},\cdot,y^{(t)}}[\ \mathcal{E}\ ].$$

Let $n' = \lceil n(1-1/d)\rceil$ and $d' = d-1$. Let $\mathcal{A}$ be the event that given $x^{(1)},\cdots,x^{(t)},y^{(1)},\cdot,y^{(t)}$, $rank(Q_{\mathbf{x}_R}) \leqslant \gamma|\mathcal{M}_p(d',n')|$, and $\overline{\mathcal{A}}$ be the complement event of $\mathcal{A}$. Now we have,

$$\begin{aligned}
&\mathbf{Pr}_{x^{(1)},\cdots,x^{(t)},y^{(1)},\cdot,y^{(t)}}[\ \mathcal{E}\ ] \\
&\leqslant \mathbf{Pr}_{x^{(1)},\cdots,x^{(t)},y^{(1)},\cdot,y^{(t)}}[\mathcal{A}] + \mathbf{Pr}_{x^{(1)},\cdots,x^{(t)},y^{(1)},\cdot,y^{(t)}}[\ \mathcal{E} \mid \overline{\mathcal{A}}\ ] \\
&\leqslant \mathbf{Pr}_{\mathbf{x_R}}[rank(Q_{\mathbf{x}_R}) \leqslant \gamma|\mathcal{M}_p(d',n')|] + p^{-\gamma|\mathcal{M}_p(d',n')|\cdot|L|}
\end{aligned}$$

where the last step follows from Eq. (4.1) and the condition $rank(Q_{\mathbf{x}_R}) > \gamma|\mathcal{M}_p(d',n')|$. Observe that $t \geqslant \eta|\mathcal{M}_p(d',n)| \geqslant \eta|\mathcal{M}_p(d',n')|$ so we can apply Lemma 4.2.4 with $(n', d' = d-1)$ in place of $(n,d)$, and $\mathbf{x} = \mathbf{x}_R$ to derive that

$$\mathbf{Pr}_{\mathbf{x_R}}[rank(Q_{\mathbf{x}_R}) \leqslant \gamma|\mathcal{M}_p(d-1,n')|] \leqslant p^{-c|\mathcal{M}_p(d,n')|}.$$

Therefore,

$$\mathop{\mathbf{E}}_{f\in_R\mathcal{P}_p(d,n)}[\ |\mathrm{bias}_j(f)|^{2t}\ ] \leqslant p^{-c|\mathcal{M}_p(d,n')|} + p^{-\gamma|\mathcal{M}_p(d-1,n')|\cdot|L|}. \tag{4.2}$$

Now, for sufficiently large $n$, by Proposition 4.2.2(a), $|\mathcal{M}_p(d,n')| \geqslant \gamma'|\mathcal{M}_p(d,n)|$ and by Corollary 4.2.3 $|\mathcal{M}_p(d-1,n')|\cdot|L| \geqslant \gamma|\mathcal{M}_p(d,n)|$. Therefore,

$$\mathop{\mathbf{E}}_{f\in_R\mathcal{P}_p(d,n)}[\ |\mathrm{bias}_j(f)|^{2t}\ ] \leqslant p^{-c\gamma'|\mathcal{M}_p(d,n)|} + p^{-\gamma^2|\mathcal{M}_p(d,n)|} \geqslant p^{-c'|\mathcal{M}_p(d,n)|}$$

for some constant $c' > 0$. Now we can apply Markov's inequality to obtain that for any $c_1 > 0$.

$$\begin{aligned}
\mathbf{Pr}_{f\in_R\mathcal{P}_p(d,n)}[|\mathrm{bias}_j(f)| > p^{-c_1 n/d}] &= \mathbf{Pr}_{f\in_R\mathcal{P}_p(d,n)}[|\mathrm{bias}_j(f)|^{2t} > p^{-2t\cdot c_1 n/d}] \\
&\leqslant \frac{p^{-c'|\mathcal{M}_p(d,n)|}}{p^{-2t\cdot c_1 n/d}} \\
&= p^{2t\cdot c_1 n/d - c'|\mathcal{M}_p(d,n)|}
\end{aligned}$$

By definition, $t = \lceil \eta|\mathcal{M}_p(d-1,n)|\rceil \leqslant \eta'|\mathcal{M}_p(d-1,n)|$ for a fixed $\eta' > \eta$. Therefore, by Proposition 4.2.2(b), $2tn/d \leqslant 2\eta'\rho_2|\mathcal{M}_p(d,n)|$. By choosing $c_1 = c'/(4\eta'\rho_2)$, we obtain that $2t\cdot c_1 n/d - c'|\mathcal{M}_p(d,n)| \leqslant -c'|\mathcal{M}_p(d,n)|/2$ and setting $c_2 = c'/2$ we derive that

$$\mathbf{Pr}_{f\in_R\mathcal{P}_p(d,n)}[|\mathrm{bias}_j(f)| > p^{-c_1 n/d}] \leqslant p^{-c_2|\mathcal{M}_p(d,n)|}$$

as required. □

It remains to prove Lemma 4.2.4 and Proposition 4.2.2. We first prove Lemma 4.2.4 using Lemma 4.1.4, which follows from Theorem 4.1.5 whose proof is in the next section. Lemma 2.4 is a generalization of Claim 2.3 in [15], and its proof follows the same lines as that in [15].

*Proof of Lemma 4.2.4 using Lemma 4.1.4.* Let $d = \lfloor \delta n \rfloor$ for $0 < \delta \leqslant 1/2$ and let $\gamma > 0$ be the minimum of $\gamma'(\delta)$ from Proposition 4.2.2 and $\gamma(\delta)$ from Corollary 4.2.3. Fix $b = \lfloor \gamma \cdot |\mathcal{M}_p(d,n)| \rfloor$. We will first check the probability that an arbitrary fixed set of $b$ columns spans the whole matrix, and then apply a union bound to obtain the final result.

Let $V$ denote the linear space spanned by those $b$ columns. Recall that each column of $Q_{x_R}$ is the evaluation of all monomials of degree at most $d$ at some point $\mathbb{F}_p^n$.

Let integer $r$ be maximal such that there are at least $p^r$ distinct elements of $\mathbb{F}_p^n$ with evaluations that are in $V$. Then by Lemma 4.1.4, we have $\mathsf{dim}(V) \geqslant |\mathcal{M}_p(d,r)|$. But since $V$ can be spanned by $b$ vectors, we have

$$\gamma|\mathcal{M}_p(d,n)| \geqslant b \geqslant \mathsf{dim}(V) \geqslant |\mathcal{M}_p(d,r)|$$

By Proposition 4.2.2(a), we have

$$|\mathcal{M}_p(d, \lceil n(1-1/d) \rceil)| \geqslant \gamma|\mathcal{M}_p(d,n)| \geqslant |\mathcal{M}_p(d,r)|$$

So $r \leqslant \lceil n(1-1/d) \rceil$. There are $p^n$ distinct evaluations and fewer than $p^{r+1}$ of them fall into $V$. So a uniform random evaluation is in $V$ with probability $< \frac{p^{r+1}}{p^n} \leqslant p^{1-\lfloor n/d \rfloor}$. Since the $t-b$ other columns of $Q_{x_R}$ are chosen uniformly and independently, the probability that these $b$ columns span the whole matrix is at most

$$(p^{1-\lfloor n/d \rfloor})^{t-b} \leqslant (p^{1-\lfloor n/d \rfloor})^{(\eta-\gamma)|\mathcal{M}_p(d,n)|}$$

since $t = \eta|\mathcal{M}_p(d,n)|$ for some $\eta > 1$ to be chosen later. Since $d \leqslant \delta n \leqslant n/2$, we have $1 - \lfloor n/d \rfloor \leqslant -n/(2d)$ and we can apply Proposition 4.2.2 to get that

$$(p^{1-\lfloor n/d \rfloor})^{t-b} \leqslant p^{-(\eta-\gamma)\frac{n}{d} \cdot |\mathcal{M}_p(d,n)|/2} \leqslant p^{-(\eta-\gamma)\rho_1|\mathcal{M}_p(d+1,n)|/2}$$

for some $\rho_1 > 0$. Therefore, by a union bound over all choices of $b$ columns we have

$$\mathbf{Pr}_{x^{(1)},\cdots,x^{(t)}}[rank(Q_{x_R}) \leqslant \gamma|\mathcal{M}_p(d,n)|] \leqslant \binom{t}{b} \cdot p^{-(\eta-\gamma)\rho_1|\mathcal{M}_p(d+1,n)|/2}.$$

Note that $\binom{t}{b} \leqslant (\frac{te}{b})^b \leqslant (\frac{2e\eta}{\gamma})^{\gamma|\mathcal{M}_p(d,n)|} \leqslant (\frac{2e\eta}{\gamma})^{\gamma|\mathcal{M}_p(d+1,n)|}$, so we have

$$\mathbf{Pr}_{x^{(1)},\cdots,x^{(t)}}[rank(Q_{x_R}) \leqslant \gamma|\mathcal{M}_p(d,n)|] \leqslant p^{|\mathcal{M}_p(d+1,n)|(\gamma\log_p(\frac{2e\eta}{\gamma})-(\eta-\gamma)\rho_1/2)}$$

Note that for any constant $c' > 0$, $\gamma\log_p(c'\eta)$ is $o(\eta)$. Therefore, for fixed constant $\gamma > 0$, we can choose a sufficiently large $\eta > 1$ such that

$$\mathbf{Pr}_{x^{(1)},\cdots,x^{(t)}}[rank(Q_{x_R}) \leqslant \gamma|\mathcal{M}_p(d,n)|] \leqslant p^{-c|\mathcal{M}_p(d+1,n)|}$$

for some constant $c > 0$. □

### *4.2.1 Proof of Proposition 4.2.2*

We first give basic inequalities regarding $|\mathcal{M}_p(d,n)|$ that are independent of the choice of $p$.

**Proposition 4.2.5.** *For $d \leqslant n$, $\sum_{i=0}^{d}\binom{n}{i} \leqslant |\mathcal{M}_p(d,n)| \leqslant \binom{n+d}{d}$.*

*Proof.* It is well known that there are $\binom{n+d}{d}$ non-negative integer solutions to the equation $\sum_{i=1}^{n} e_i \leqslant d$. Thus we have $|\mathcal{M}_p(d,n)| \leqslant \binom{n+d}{d}$. On the other hand, if we only consider multi-linear terms, we obtain $\sum_{i=0}^{d}\binom{n}{i} \leqslant |\mathcal{M}_p(d,n)|$. □

We now prove part (a): For $\mathbf{e} = (e_1,\cdots,e_k)$ where $1 \leqslant e_i \leqslant p-1$, let $\mathcal{M}_{\mathbf{e},n}$ denote the set of monomials of the form $\prod_{i=1}^{k} x_{h(i)}^{e_i}$, $1 \leqslant h(1) < h(2) < \cdots < h(k) \leqslant n$. Then we have $|\mathcal{M}_p(d,n)| = \sum_{\mathbf{e}:\sum_i e_i \leqslant d}|\mathcal{M}_{\mathbf{e},n}|$. Therefore

$$\frac{|\mathcal{M}_p(d,n')|}{|\mathcal{M}_p(d,n)|} \geqslant \min_{\mathbf{e}:\sum_i e_i \leqslant d}\frac{|\mathcal{M}_{\mathbf{e},n'}|}{|\mathcal{M}_{\mathbf{e},n}|}$$

For fixed $\mathbf{e} = (e_1,\cdots,e_k)$, We then argue that for all integer $n$, $|\mathcal{M}_{\mathbf{e},n}| = f_{\mathbf{e}} \cdot \binom{n}{k}$, for some $f_{\mathbf{e}} > 0$ that only depends on $\mathbf{e}$. This is because there are $\binom{n}{k}$ different ways to choose

the $k$ variables appearing in the monomial, and each of them are indistinguishable, hence $|\mathcal{M}_{\mathbf{e},n}|$ is proportional to $\binom{n}{k}$.

Therefore, we have

$$\frac{|\mathcal{M}_{\mathbf{e},n'}|}{|\mathcal{M}_{\mathbf{e},n}|} = \frac{\binom{n'}{k}}{\binom{n}{k}}$$

This quantity is a decreasing function of $k$. Hence we have

$$\frac{|\mathcal{M}_p(d,n')|}{|\mathcal{M}_p(d,n)|} \geqslant \frac{\binom{n'}{d}}{\binom{n}{d}} = \prod_{i=0}^{d-1} \frac{n'-i}{n-i}$$

From the well known inequality $\ln(1+x) \geqslant \frac{x}{1+x}$, we have

$$\begin{aligned}\ln \prod_{i=0}^{d-1} \frac{n'-i}{n-i} = \sum_{i=0}^{d-1} \ln \frac{n'-i}{n-i} &\geqslant \sum_{i=0}^{d-1} \frac{n'-n}{n'-i} = -(n-n') \sum_{i=0}^{d-1} \frac{1}{n'-i} \\ &\geqslant -(n-n') \int_{n'-d}^{n'} x^{-1} \mathbf{d}x = -(n-n') \ln \frac{n'}{n'-d} \geqslant -\frac{d(n-n')}{n'-d}\end{aligned}$$

By the condition $n' \geqslant (1-1/d)n$ and $d \leqslant \delta n$, we have

$$\begin{aligned}\prod_{i=0}^{d-1} \frac{n'-i}{n-i} &\geqslant \exp(-\frac{d(n-(1-1/d)n)}{n'-\delta n}) = \exp(-\frac{n}{n'-\delta n}) \\ &\geqslant \exp(-\frac{1}{1-1/d-\delta}) \geqslant \exp(-\frac{1}{1/2-\delta}) := \gamma'(\delta),\end{aligned}$$

which completes the proof of part (a).

We now prove part (b): Define $H := \{(q_1, q_2) : q_1 \in \mathcal{M}_p(d-1,n), q_2 \in \mathcal{M}_p(d,n), \exists i \in [n] \text{ s.t. } q_2 = x_i q_1\}$. We will obtain the inequalities by bounding $|H|$.

We first bound $|H|$ in terms of $|\mathcal{M}_p(d-1,n)|$. Clearly for each $q_1 \in \mathcal{M}_p(d-1,n)$, there are at most $n$ choices of $x_i$ to yield $q_2 = x_i q_1$, so $|H| \leqslant n|\mathcal{M}_p(d-1,n)|$. On the other hand, any $x_i$ that does not have degree $p-1$ in $q_1$ can be chosen. There are at most $\frac{d-1}{p-1}$ variables in $q_1$ having degree $p-1$ so we can choose at least $n - \frac{d-1}{p-1} > n - \frac{n}{p-1} \geqslant \frac{n}{2}$ variables $x_i$ since $p \geqslant 3$. This gives us $|H| > \frac{n}{2}|\mathcal{M}_p(d-1,n)|$.

We now bound $|H|$ in terms of $|\mathcal{M}_p(d,n)|$. Each $q \in \mathcal{M}_p(d,n)$ contains at most $d$ distinct variables, hence $|H| \leqslant d|\mathcal{M}_p(d,n)|$.

It immediately follows that $\mathcal{M}_p(d,n) \geqslant |H|/d \geqslant \frac{n}{2d}|\mathcal{M}_p(d-1,|)$ and hence we can choose $\rho_2 = 2$.

To lower bound $|H|$ in terms of $|\mathcal{M}_p(d,n)|$, we show that a large portion of monomials contain many distinct variables and hence each $q_2 \in \mathcal{M}_p(d,n)$ can be associated with many different $q_1$. We first bound the number of monomials that have degree at most $d$ and are composed of at most $k \leqslant d$ distinct variables. We can generate such monomials by first choosing $k$ variables, then using these variables to form a monomial of degree $\leqslant d$ and so we can upper bound the number of such monomials by $\binom{n}{k}\binom{k+d}{d}$. For sufficiently small $k$ we can argue that this is a small fraction of $\mathcal{M}_p(d,n)$: Suppose that $k \leqslant d/6$. Since by hypothesis, $d \leqslant \delta n \leqslant n/2$, we have $k+d \leqslant n-k$ and

$$\begin{aligned}
\frac{\binom{n}{k}\binom{k+d}{d}}{|\mathcal{M}_p(d,n)|} &\leqslant \frac{\binom{n}{k}\binom{k+d}{d}}{\binom{n}{d}} \qquad \text{by Proposition 4.2.5} \\
&= \frac{(k+d)!}{(k!)^2(n-k)\cdots(n-d+1)} = \frac{(k+d)\cdots(2k+1)}{(n-k)\cdots(n-d+1)} \cdot \binom{2k}{k} \\
&\leqslant \left(\frac{k+d}{n-k}\right)^{d-k} \cdot 2^{2k} \leqslant (7/11)^{5k} \cdot 2^{2k} \leqslant (3/7)^k.
\end{aligned}$$

Summing over all values of $k \leqslant d/6$ we obtain that a total fraction at most $3/4$ of all monomials in $\mathcal{M}_p(d,n)$ have at most $d/6$ distinct variables. Therefore, since at least $1/4$ of $\mathcal{M}_p(d,n)$ contain at least $d/6$ distinct variables, it must be the case that $|H| \geqslant \frac{d}{24} \cdot |\mathcal{M}_p(d,n)|$. Since $|H| \leqslant n \cdot |\mathcal{M}_p(d-1,n)|$, we obtain that $|\mathcal{M}_p(d,n)|/24 \leqslant \frac{n}{d}|\mathcal{M}_p(d-1,n)|$. Hence we derive (b) with $\rho_1 = 1/24$. This completes the proof of Proposition 4.2.2(b).

### *4.2.2 Lower bound on the likelihood of bias*

We now prove Proposition 4.1.3, on the limits on the extent to which Theorem 4.1.1 can be improved. The argument is analogous to that of [15] for the case of $\mathbb{F}_2$.

*Proof of Proposition 4.1.3.* We follow the same division of variables $[n]$ into parts $L$ and $R$ with $|L| = \lfloor \frac{n}{d} \rfloor$ and $|R| = n' = \lceil n(1-1/d) \rceil$ and $d' = d-1$ that we used for the

upper bound on the bias. Define $\mathcal{L}$ to be the set of all polynomials in $\mathcal{P}_p(d,n)$ whose monomials are from the set $\mathcal{M}' \subseteq \mathcal{M}_p(d,n)$ (defined earlier) that have degree 1 on $L$ and degree at most $d-1$ on $R$. By Corollary 4.2.3, there is some constant $\gamma > 0$ such that for sufficiently large $n$, $|\mathcal{M}'| \geqslant \gamma \cdot |\mathcal{M}_p(d,n)|$ and hence $|\mathcal{L}| \geqslant p^{\gamma|\mathcal{M}_p(d,n)|}$. Therefore, we have $|\mathcal{L}|/|\mathcal{P}_p(d,n)| \geqslant p^{-(1-\gamma)|\mathcal{M}_p(d,n)|}$.

Now consider the expected bias of polynomials in $\mathcal{L}$: We can write $f$ chosen uniformly from $\mathcal{L}$ uniquely as

$$f(x) = \sum_{i \in L} x_i \cdot g_i(x_R)$$

where the $g_i$ are independently chosen polynomials over monomials $\mathcal{M}_p(d-1,n')$ on $R$.

For $j \in \mathbb{F}_p^*$,

$$\begin{aligned}
\mathop{\mathbf{E}}_{f \in_R \mathcal{L}} \mathrm{bias}_j(f) &= \mathop{\mathbf{E}}_{f \in_R \mathcal{L}} \mathop{\mathbf{E}}_{x \in_R \mathbb{F}_p^n} \omega^{j \cdot f(x)} \\
&= \mathop{\mathbf{E}}_{f \in_R \mathcal{L}} \mathop{\mathbf{E}}_{x_L \in_R \mathbb{F}_p^L} \mathop{\mathbf{E}}_{x_R \in_R \mathbb{F}_p^R} \omega^{j \cdot f(x)} \\
&= \mathop{\mathbf{E}}_{x_L \in_R \mathbb{F}_p^L} \mathop{\mathbf{E}}_{x_R \in_R \mathbb{F}_p^R} \mathop{\mathbf{E}}_{f \in_R \mathcal{L}} \omega^{j \cdot f(x)}.
\end{aligned}$$

Now with probability $p^{-|L|}$, all the $x_i$ for $i \in L$ are 0 and every $f \in \mathcal{L}$ evaluates to 0, so $\mathbf{E}_{f \in_R \mathcal{L}}\ \omega^{j \cdot f(0^L, x_R)} = 1$. With the remaining probability, $x_L \neq 0^L$ and hence there is some $i \in L$ and $b_i \neq 0$ such that $x_i = b_i$. For $f$ chosen at random from $\mathcal{L}$, for each fixed value of $x_L = b_L$ with $b_i \neq 0$, we have

$$f(b_L, x_R) = b_i g_{i0} + f'(x_R)$$

where $g_{i0}$ is the constant term of the polynomial $g_i$ and is chosen independently of $f'$. Since $g_{i0}$ is uniformly chosen from $\mathbb{F}_p$ for random $f$ in $\mathcal{L}$ and since $b_i \neq 0$, $b_i g_{i0}$ is also uniformly chosen from $\mathbb{F}_p$. Further, since $g_{i0}$ is independent of $f'$, for every fixed $x_R$, the value $\mathbf{E}_{f \in_R \mathcal{L}}\, \omega^{j \cdot f(b_L, x_R)} = 0$. Therefore, $\mathbf{E}_{f \in_R \mathcal{L}}\, \mathrm{bias}_j(f) = p^{-|L|}$. Now since $|\mathrm{bias}_j(f)| \leqslant 1$, we obtain

$$\mathbf{Pr}_{f \in_R \mathcal{L}}[\ |\mathrm{bias}_j(f)| \geqslant p^{-|L|}/2\ ] \geqslant p^{-|L|}/2.$$

Therefore

$$\mathbf{Pr}_{f\in_R\mathcal{P}_p(d,n)}[|\mathrm{bias}_j(f)| \geqslant p^{-|L|}/2] \geqslant \frac{|\mathcal{L}|}{|\mathcal{P}_p(d,n)|}\cdot p^{-|L|}/2 \geqslant p^{-c'|\mathcal{M}_p(d,n)|}$$

for some $c' < 1$ since $|L| \ll |\mathcal{M}_p(d,n)|$. Since $|L| = \lfloor n/d \rfloor \geqslant 2$, we obtain $p^{-|L|}/2 > p^{-c''n/d}$ for some constant $c'' > 0$ as required. □

### *4.3 An extremal property of truncated Reed-Muller codes*

In this section we prove Theorem 4.1.5. It is closely related to the problem of maximizing the number of common zeros of a set of linearly independent polynomials, which has been studied by Heijnen and Pelikaan [69]. Here, we make use of more general recent results of Beelen and Datta [13] who extend the methods of [69].

We start with some useful notations.

**Definition 4.3.1.** *Fix integers $p, n$. Let $F = \{0, \cdots, p-1\}^n$. For $d \leqslant n(p-1)$, define*

$$F_{\leqslant d} = \{a \in F : \sum_{i=1}^{n} a_i \leqslant d\}.$$

*Define the* ascending lexicographic order $\leqslant$ *over $F$ as for $a, b \in F$, $b \leqslant a$ if and only if $\exists i \in [n]$ so that $b_j = a_j$ for all $j < i$ and $b_i < a_i$. Define the* descending lexicographic order $\geqslant$ *as $a \geqslant b$ if and only if $b \leqslant a$.*

**Proposition 4.3.2** (Proposition 4.3 in [13])**.** *Recall that $\mathcal{P}_p(d,n)$ is the set of polynomials of degree at most $d$ in $n$ variables over $\mathbb{F}_p$. Fix integer $r$. Let $f_1, \cdots, f_r \in \mathcal{P}_p(d,n)$ be linear independent over $\mathbb{F}_p$. Let $Z(f_1, \cdots, f_r)$ denote the number of common zeros between $f_1, \cdots, f_r$. Then*

$$Z(f_1, \cdots, f_r) \leqslant \sum_{i=1}^{n} a_{r,i} p^{n-i},$$

*where $a_r \in F_{\leqslant d}$ is the $r$-th element in $F_{\leqslant d}$ in descending lexicographic order.*

**Proposition 4.3.3** (Lemma 4.2, Lemma 4.3 and Proposition 4.5 in [13]). *For $b = (b_1, \cdots, b_n) \in F_{\leqslant d}$, define $f_b \in \mathcal{P}_p(d,n)$ as*

$$f_b(x) := \prod_{i=1}^{n} \prod_{j=1}^{b_i} (x_i - (j-1)).$$

*Let $a_1, \cdots, a_r$ be the first $r$ elements in $F_{\leqslant d}$ in descending lexicographic order. Then*

$$Z(f_{a_1}, \cdots, f_{a_r}) = \sum_{i=1}^{n} a_{r,i} p^{n-i} := h_d(n,r).$$

*Moreover, the common zeros of $f_{a_1}, \cdots, f_{a_r}$ are the first $h_d(n,r)$ elements in $F^n$ in ascending lexicographic order.*

**Proposition 4.3.4.** *$\{f_b\}_{b \in F_{\leqslant d}}$ forms a basis of $\mathcal{P}_p(d,n)$.*

*Proof.* The elements $b \in F_{\leqslant d}$ are the exponent vectors of the monomials $x^b := \prod_{i=1}^{n} x^{b_i} \in \mathcal{M}_p(d,n)$ which form a basis of $\mathcal{P}_p(d,n)$. The proposition follows by observing that the leading term of each $f_b$ is the monomial $x^b$. □

Now we come to the quantities we focus on in this work.

**Definition 4.3.5.** *For integer $m$, let $S_m \subset \mathbb{F}_p^n$ be the subset that contains the smallest $m$ elements.*

**Theorem 4.3.6** (Duality). *Fix integers $n, d, p$. For any subset $S \subset \mathbb{F}_p^n$ with $|S| = m$, we have*

$$rank(M_S^{(d)}) \geqslant rank(M_{S_m}^{(d)}).$$

*Proof.* Let $V_S \subset \mathbb{F}_p^{|\mathcal{M}_p(d,n)|}$ be the linear subspace spanned by the rows in $M^{(d)}$ indexed by $S$. Then $rank(M_S^{(d)}) = k$ if and only if $\dim(V_S) = k$, which is equivalent to $\dim(V_S^\top) = |\mathcal{M}_p(d,n)| - k$.

On the other hand, we can interpret elements in $\mathbb{F}_p^{|\mathcal{M}_p(d,n)|}$ as polynomials by considering each entry as the coefficient in the corresponding monomial. Formally, this is the bijection $\phi : \mathbb{F}_p^{|\mathcal{M}_p(d,n)|} \to \mathcal{P}_p(d,n)$ defined as

$$\phi((v_\alpha)_{\alpha \in \mathcal{M}_p(d,n)}) = \sum_{\alpha \in \mathcal{M}_p(d,n)} v_\alpha \cdot \alpha.$$

From this view, $f \in \mathcal{P}_p(d,n)$ is contained in $\mathsf{dim}(V_S^\top)$ if and only if $\langle \phi^{-1}(f), M_{\{x\}}^{(d)} \rangle = 0$ for all $x \in S$. But $\langle \phi^{-1}(f), M_{\{x\}}^{(d)} \rangle$ is just $f(x)$, which means that $S$ is the set of common zeros for all $f$ so that $\phi(f) \in \mathsf{dim}(V_S^\top)$.

Let $r$ be the largest integer so that $h_d(n,r) \geqslant m$, where $h_d(n,r)$ is defined in Proposition 4.3.3.

We first claim that $rank(M_S^{(d)}) \geqslant |\mathcal{M}_p(d,n)| - r$. If this is not the case, then $\mathsf{dim}(V_S^\top) = |\mathcal{M}_p(d,n)| - rank(M_S^{(d)}) > r$; namely, we can find $r+1$ linearly independent polynomials $f_1, \cdots, f_{r+1}$ so that they evaluate to 0 on $S$. Therefore, $r+1$ linearly independent polynomials can have $m$ common zeros. By Proposition 4.3.2 and Proposition 4.3.3, this means $h_d(n, r+1) \geqslant m$, which violates our definition of $r$.

We then argue that $rank(M_{S_m}^{(d)}) \leqslant |\mathcal{M}_p(d,n)| - r$. This follows because we can choose $a_1, \cdots, a_r$ as in Proposition 4.3.3, and construct $V_r \subset \mathbb{F}_p^{|\mathcal{M}_p(d,n)|}$ as the linear span of $\{\phi^{-1}(f_{a_i})\}_{i=1}^r$. By Proposition 4.3.4, $\{f_{a_i}\}_{i=1}^r$ are linearly independent. Since $\phi$ is linear, $\{\phi^{-1}(f_{a_i})\}_{i=1}^r$ are also linearly independent, so $\mathsf{dim}(V_r) = r$. By Proposition 4.3.3, $S_m$ are the common zeros of $f_{a_1}, \cdots, f_{a_r}$, hence $V_{S_m}$ is contained in $V_r^\top$. Therefore

$$rank(M_{S_m}^{(d)}) \leqslant \mathsf{dim}(V_{S_m}) \leqslant \mathsf{dim}(V_r^\top) = |\mathcal{M}_p(d,n)| - r,$$

which completes the proof. □

Theorem 4.3.6 is sufficient to prove Theorem 4.1.5. However, Proposition 4.3.2 and Proposition 4.3.3 does not give the explicit expression of $rank(M_{S_m}^{(d)})$. Here we give an explicit recursive expression.

**Definition 4.3.7.** *Define $g_d(m)$ as the rank of $M_{S_m}^{(d)}$. For the completeness of definition, we set $g_d(m) = 0$ when $d < 0$ or $m = 0$.*

The $g_d$ function is easy to compute when the input $m$ is a power of $p$.

**Lemma 4.3.8.** *For integer $r \geqslant 0$, $g_d(p^r) = |\mathcal{M}_p(d,r)|$.*

*Proof.* The rows in $S_{p^r}$ correspond to assignments that fix the first $n-r$ variables to 0 and have all possible values for the remaining $r$ variables. Therefore, $M^{(d)}_{S_{p^k}}$ is 0 except on the columns indexed by monomials on these last $r$ variables and has full column rank on the remaining $|\mathcal{M}_p(d,r)|$ columns since no non-zero polynomial using these monomials can be identically 0 over $\mathbb{F}_p^r$. □

More generally, we have the following recursive formulation of the $g_d$ function.

**Theorem 4.3.9.** *Let $r$ be the unique integer so that $p^r \leqslant m < p^{r+1}$. Let $m = k \cdot p^r + c$. Then*

$$g_d(m) = \sum_{i=0}^{k-1} g_{d-i}(p^r) + g_{d-k}(c).$$

*As a special case, when $c = 0$, we have $g_d(k \cdot p^r) = \sum_{i=0}^{k-1} g_{d-i}(p^r)$.*

The $g_d$ function can be analyzed using the $LU$ decomposition of the $(n+1)$-dimension Vandermonde matrix $V^{(n)}$ on variables $x_0, x_1, \ldots x_n$ (i. e., $V^{(n)}_{ij} = x_i^j$) given by Oruç and Phillips [109].

**Proposition 4.3.10** (Theorem 2.1 in [109])**.** *$V^{(n)}$ has LU-decomposition $V^{(n)} = L^{(n)}U^{(n)}$ where lower triangular matrix $L^{(n)} \in \mathbb{R}^{(n+1)\times(n+1)}$ has all diagonal entries 1, and upper triangular matrix $U^{(n)} \in \mathbb{R}^{(n+1)\times(n+1)}$ has $U^{(n)}_{i,i} = \prod_{j<i}(x_i - x_j)$ for all $i \in \{0, 1 \ldots, n\}$.*

*Proof of Theorem 4.3.9.* For the sake of convenience, let $\mathcal{M}_p^{=d}(r)$ be the set of monomials over the last $r$ variables whose degree equals $d$, and $\mathcal{M}_p^{\leqslant d}(r)$ be the set of monomials over the last $r$ variables whose degree is at most $d$.

Consider the block structure of the matrix. For $i = 0, 1, \cdots, p-1$, let $A_i$ be the submatrix with $S_{p^r}$ as rows and $\mathcal{M}_p^{=d-p+i+1}(r)$ as columns. Let $A_{\leqslant i}$ be the submatrix $(A_0, \cdots, A_i)$. Then its columns are given by $\mathcal{M}_p^{\leqslant d-p+i+1}(r)$.

Now, let us consider the rows for $R_t := \{a \in \mathbb{F}_p^n \mid a_1 = \cdots = a_{n-r-1} = 0,\ a_{n-r} = t\}$. The non-zero parts correspond to monomials that only depend on $x_{n-r}, x_{n-r+1}, \cdots, x_n$. If we group all the monomials by their degree on $x_{n-r}$ then, for $t \leqslant k-1$, the row will be of

the form

$$A_{\leqslant p-1},\ t\cdot A_{\leqslant p-2},\ t^2\cdot A_{\leqslant p-3},\ \cdots,\ t^{p-1}\cdot A_{\leqslant 0}.$$

Things are a little different for $t=k$, since $|R_k|<p^r$. In this case, we define $A'_{\leqslant i}$ to be the first $c$ rows of $A_{\leqslant i}$ and it is easy to check that the row is of the form

$$A'_{\leqslant p-1},\ k\cdot A'_{\leqslant p-2},\ k^2\cdot A'_{\leqslant p-3},\ \cdots,\ k^{p-1}\cdot A'_{\leqslant 0}.$$

Therefore the matrix $M^{(d)}_{S_m}$ is of the form:

| | $\mathcal{M}_p^{\leqslant d}(r)$ | $x_{n-r}\cdot\mathcal{M}_p^{\leqslant d-1}(r)$ | $x_{n-r}^2\cdot\mathcal{M}_p^{\leqslant d-2}(r)$ | $\cdots$ | $x_{n-r}^{p-1}\cdot\mathcal{M}_p^{\leqslant d-p+1}(r)$ |
|---|---|---|---|---|---|
| $\{a_{n-r}=0\}$ | $A_{\leqslant p-1}$ | $0$ | $0$ | $\cdots$ | $0$ |
| $\{a_{n-r}=1\}$ | $A_{\leqslant p-1}$ | $A_{\leqslant p-2}$ | $A_{\leqslant p-3}$ | $\cdots$ | $A_{\leqslant 0}$ |
| $\{a_{n-r}=2\}$ | $A_{\leqslant p-1}$ | $2\cdot A_{\leqslant p-2}$ | $4\cdot A_{\leqslant p-3}$ | $\cdots$ | $2^{p-1}\cdot A_{\leqslant 0}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $\{a_{n-r}=k-1\}$ | $A_{\leqslant p-1}$ | $(k-1)\cdot A_{\leqslant p-2}$ | $(k-1)^2\cdot A_{\leqslant p-3}$ | $\cdots$ | $(k-1)^{p-1}\cdot A_{\leqslant 0}$ |
| $\{a_{n-r}=k\}$ | $A'_{\leqslant p-1}$ | $k\cdot A'_{\leqslant p-2}$ | $k^2\cdot A'_{\leqslant p-3}$ | $\cdots$ | $k^{p-1}\cdot A'_{\leqslant 0}$ |

We observe the following:

- $A_{\leqslant p-i-1}$ is the first $|\mathcal{M}_p^{\leqslant d-i-1}(r)|$ columns of $A_{\leqslant p-i}$, and
- $A'_{\leqslant i}$ is the first $c$ rows of $A_{\leqslant i}$.

$M^{(d)}_{S_m}$ is closely related to the the Vandermonde matrix $V^{(p-1)}[0,1,\cdots,p-1]$. To see this, notice that $M^{(d)}_{S_m}$ is a submatrix of $V = V^{(p-1)}[0,1,\cdots,p-1]\otimes A_{\leqslant p-1}\in\mathbb{R}^{p\cdot p^r\times p\cdot|\mathcal{M}_p^{\leqslant d}(r)|}$. By Proposition 4.3.10, $V^{(p-1)}[0,1,\cdots,p-1]=LU$ where lower triangular matrix $L\in\mathbb{R}^{p\times p}$ and upper triangular matrix $U\in\mathbb{R}^{p\times p}$ satisfy $L_{i,i}=1$ and $U_{i,i}\neq 0$ for $i=0,\cdots,p-1$. Therefore, if we set $L'=L\otimes I_{p^r}\in\mathbb{R}^{p\cdot p^r\times p\cdot p^r}$, $C=I_p\otimes A_{\leqslant p-1}\in\mathbb{R}^{p\cdot p^r\times p\cdot|\mathcal{M}_p^{\leqslant d}(r)|}$ and $U'=U\otimes I_{|\mathcal{M}_p^{\leqslant d}(r)|}\in\mathbb{R}^{p\cdot|\mathcal{M}_p^{\leqslant d}(r)|\times p\cdot|\mathcal{M}_p^{\leqslant d}(r)|}$, we have $V=L'CU'$. Notice that $L'$ is lower-triangular, because $L$ is lower-triangular and $I_{p^r}$ is diagonal. Similarly $U'$ is upper-triangular.

Let $S_L\subset[p\cdot p^r]$ be the set of indices of the rows of $M^{(d)}_{S_m}$ and $S_U\subset[p\cdot|\mathcal{M}_p^{\leqslant d}(r)|]$ be the set of indices of the columns of $M^{(d)}_{S_m}$. Then $S_L$ is simply $S_m$; $S_U$ is more complicated. We

can group the columns of $V$ into $p$ groups, each of size $|\mathcal{M}_p^{\leqslant d}(r)|$. By the first observation, for $i = 0, \cdots, p-1$, we simply take the first $|\mathcal{M}_p^{\leqslant d-i}(r)|$ columns in each block.

Let $P$ be the $m$ by $p \cdot p^r$ matrix defined as $P_{ij} = 1$ if $i = j$ and $P_{ij} = 0$ otherwise. Let $Q$ be the $p \cdot |\mathcal{M}_p^{\leqslant d}(r)|$ by $\sum_{i=0}^{p-1} |\mathcal{M}_p^{\leqslant d-i}(r)|$ matrix defined as $Q_{ij} = 1$ if $i = j \in S_U$, and $Q_{ij} = 0$ otherwise. Then clearly we have $M_{S_m}^{(d)} = PL'CU'Q$.

Write $L'$ as $\begin{pmatrix} L_1 & 0 \\ L_2 & L_3 \end{pmatrix}$ where $L_1 \in \mathbb{R}^{m \times m}$. Then we have $PL' = \begin{pmatrix} L_1 & 0 \end{pmatrix} = L_1 P$. Similarly, let $U_1$ be the principal submatrix of $U$ indexed by $S_U$. Then we have $UQ = QU_1$. Since $L', U'$ are triangular matrices with non-zero diagonal entries, $L_1, U_1$ are also triangular matrices with non-zero diagonal entries, and hence invertible.

Now we have $M_{S_m}^{(d)} = PL'CU'Q = L_1 PCQU_1$. Let $C_1 = PCQ$. Then $C_1$ is of the form

| | $\mathcal{M}_p^{\leqslant d}(r)$ | $x_{n-r} \cdot \mathcal{M}_p^{\leqslant d-1}(r)$ | $x_{n-r}^2 \cdot \mathcal{M}_p^{\leqslant d-2}(r)$ | $\cdots$ | $x_{n-r}^{k-1} \cdot \mathcal{M}_p^{\leqslant d-k+1}(r)$ | $x_{n-r}^{k} \cdot \mathcal{M}_p^{\leqslant d-k}(r)$ |
|---|---|---|---|---|---|---|
| $\{a_{n-r} = 0\}$ | $A_{\leqslant p-1}$ | 0 | 0 | $\cdots$ | 0 | 0 |
| $\{a_{n-r} = 1\}$ | 0 | $A_{\leqslant p-2}$ | 0 | $\cdots$ | 0 | 0 |
| $\{a_{n-r} = 2\}$ | 0 | 0 | $A_{\leqslant p-3}$ | $\cdots$ | 0 | 0 |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $\{a_{n-r} = k-1\}$ | 0 | 0 | 0 | $\cdots$ | $A_{\leqslant p-k}$ | 0 |
| $\{a_{n-r} = k\}$ | 0 | 0 | 0 | $\cdots$ | 0 | $A'_{\leqslant p-k-1}$ |

Since $L_1, U_1$ are invertible, we have $rank(M_{S_m}^{(d)}) = rank(C_1)$. By the definition of the $g_d$ function, the rank of $A_{\leqslant p-i}$ is $g_{d-i}(p^r)$ and that of $A'_{\leqslant p-k-1}$ is $g_{d-k}(c)$. Hence

$$g_d(m) = g_d(k \cdot p^r + c) = \sum_{i=0}^{k-1} g_{d-i}(p^r) + g_{d-k}(c).$$

□

# Chapter 5

# BOUNDED SPACE LEARNING ALGORITHMS AND THE STATISTICAL QUERY MODEL

Let $\mathcal{C}$ be a collection of concepts. Let $\mathcal{X}$ be the set of all samples. Recall the streaming model in Chapter 3: There is some unknown ground truth $c \in \mathcal{C}$ which is chosen uniformly at random; we want to identify $c$ with a stream of labelled samples $(x_1, y_1), (x_2, y_2), \cdots$, where $x_i$ are chosen i.i.d from $\mathcal{X}$ and $y_i = c(x_i)$. After seeing each sample, we can make arbitrary edits in the storage. The computational resources we care about are the number of samples and the space we need in the computation.

Without any further structural information on $\mathcal{C}$ or $\mathcal{X}$, we have the following learning algorithms:

1. We can remember all the labelled samples in the memory, and after seeing sufficient samples, we can return the correct answer for sure. This method requires $O(\log |\mathcal{C}|)$ samples and $O(\log |\mathcal{C}| \cdot \log |\mathcal{X}|)$ space.

2. We can go over all concepts and verify each one with $O(\log |\mathcal{C}|)$ samples. This method requires $O(|\mathcal{C}| \log |\mathcal{C}|)$ samples and $O(\log |\mathcal{C}|)$ space.

We now give another non-trivial learning algorithm based on a deterministic query model.

**Lemma 5.0.1.** *There is a generic algorithm that runs with $O(|\mathcal{X}|^2)$ samples and $O(\log |\mathcal{C}|)$ space.*

*Proof.* Let $A$ be some deterministic algorithm for learning $\mathcal{C}$ under the query model, that is, in each round, $A$ chooses and queries some sample $x \in \mathcal{X}$ and receives $c(x)$. $A$ can be represented as a decision tree whose depth is at most $|\mathcal{X}|$, and the width of the tree is

at most $|\mathcal{C}|$. We can simulate $A$ in the stream model, by keeping track of where we are in the tree, and waiting until the sample on the current node in the decision tree appears in the stream. On average we need to wait for $O(|\mathcal{X}|)$ samples to see a specific sample. This gives us a learning algorithm with $O(|\mathcal{X}|^2)$ sample complexity. For the space usage, the algorithm needs to record its position in the decision tree at each step, and it just takes $O(\log |\mathcal{C}|)$ space to specify a node in the tree, so this algorithm uses $O(\log |\mathcal{C}|)$ space. □

Raz et al. [113, 56, 11] showed that, for some problems, the above algorithms are essentially all we can do: either we use $\Omega(\log(|\mathcal{C}|)\log(|\mathcal{X}|))$ space, or we have to see $\mathsf{poly}(\min\{|\mathcal{C}|, |\mathcal{X}|\})$ many samples. There is nothing in between; in other words, no efficient algorithm can learn $\mathcal{C}$ in bounded space. Let $\mathcal{D}_{\mathcal{X}}$ be the distribution over $\mathcal{X}$ that the samples come from. We say that concept class $\mathcal{C}$ *can be learned in bounded space* with accuracy $1-\varepsilon$, if there exists an algorithm that takes $(\min\{|\mathcal{C}|, |\mathcal{X}|\})^{o(1)}$ samples and $o(\log(|\mathcal{C}|) \cdot \log(|\mathcal{X}|))$ space and return $\hat{c}$ so that $\mathbf{Pr}_{x\sim\mathcal{D}_{\mathcal{X}}}[\hat{c}(x) \neq c(x)] \leqslant \varepsilon$.

Recently, Gonen, Lovett, and Moshkovitz [59] built up a connection between bounded space learning with the statistical query model introduced in Section 2.1.1. In particular, they use the concept of statistical query dimension (SQ-dimension) to characterize bounded space learning.

**Definition 5.0.2** (SQ-dimension)**.** *Let $\mathcal{D}_{\mathcal{X}}$ be a distribution over $\mathcal{X}$. Let $\langle\cdot,\cdot\rangle_{\mathcal{D}_{\mathcal{X}}}$ be the inner product over $\mathcal{C}$ with respect to $\mathcal{D}_{\mathcal{X}}$. That is, $\langle c_1, c_2\rangle_{\mathcal{D}_{\mathcal{X}}} = E_{x\sim\mathcal{D}_{\mathcal{X}}}[c_1(x)c_2(x)]$. Then $sq_{\mathcal{D}_{\mathcal{X}}}(\mathcal{C})$, the SQ-dimension of $\mathcal{C}$ with respect to $\mathcal{D}_{\mathcal{X}}$ is the largest integer $d$ so that there are distinct $c_1, \cdots, c_d \in \mathcal{C}$ so that for $i \neq j$, $\langle c_i, c_j\rangle_{\mathcal{D}_{\mathcal{X}}} \leqslant \frac{1}{d}$.*

For certain regime of parameters, Gonen, Lovett, and Moshkovitz gave the following characterization:

**Theorem 5.0.3** (Theorem 5, Theorem 6 in [59])**.** *Fix integer $N$. Let $|\mathcal{C}|, |\mathcal{X}| = poly(N)$. Let $\varepsilon = N^{-o(1)}$. Then the following statements are equivalent:*

- *There exists an algorithm that takes $N^{o(1)}$ samples and uses $o(\log^2 N)$ space and can return $\hat{c}$ so that $\mathbf{Pr}_{x\sim\mathcal{D}_{\mathcal{X}}}[\hat{c}(x) \neq c(x)] \leqslant \varepsilon$.*

- *For any distribution $Q$ over $\mathcal{X}$ satisfying $\varepsilon \leqslant \frac{Q(x)}{\mathcal{D}_{\mathcal{X}}(x)} \leqslant 1/\varepsilon$, $sq_Q(\mathcal{C}) \leqslant \mathsf{poly}(1/\varepsilon)$.*

Gonen, Lovett, and Moshkovitz conjectured that Theorem 5.0.3 holds for a much broader regime of parameters. They originally conjectured that a learning problem can be learned with $|\mathcal{C}|^{o(1)}$ samples and $o(\log(|\mathcal{C}|) \cdot \log(|\mathcal{X}|))$ space if and only if this learning problem has small statistical query dimension. However, as we have presented in Lemma 5.0.1, there exists a generic algorithm that runs with $O(|\mathcal{X}|^2)$ samples and $O(\log|\mathcal{C}|)$ space. Afterwards, Gonen et al. revised their conjecture to only work when $|\mathcal{X}|$ and $|\mathcal{C}|$ are polynomially related. This suggests that in the general setting, we need a better criteria for bounded space learning than the statistical query dimension.

# Chapter 6

# TOTAL LEAST SQUARES REGRESSION IN INPUT SPARSITY TIME

## *6.1 Introduction*

In the least squares regression problem, we are given an $m \times n$ matrix $A$ and an $m \times 1$ vector $b$, and we seek to find an $x \in \mathbb{R}^n$ that minimizes $\|Ax - b\|_2^2$. A natural geometric interpretation is that there is an unknown hyperplane in $\mathbb{R}^{n+1}$, specified by the normal vector $x$, for which we have $m$ points on this hyperplane, the $i$-th of which is given by $(A_i, \langle A_i, x \rangle)$, where $A_i$ is the $i$-th row of $A$. However, due to noisy observations, we do not see the points $(A_i, \langle A_i, x \rangle)$, but rather only see the point $(A_i, b_i)$, and we seek to find the hyperplane which best fits these points, where we measure (squared) distance only on the $(n+1)$-st coordinate. This naturally generalizes to the setting in which $B$ is an $m \times d$ matrix, and the true points have the form $(A_i, A_i X)$ for some unknown $n \times d$ matrix $X$. This setting is called multiple-response regression, in which one seeks to find $X$ to minimize $\|AX - B\|_F^2$, where for a matrix $Y$, $\|Y\|_F^2$ is its squared Frobenius norm, i. e., the sum of squares of each of its entries. This geometrically corresponds to the setting when the points live in a lower $n$-dimensional flat of $\mathbb{R}^{n+d}$, rather than in a hyperplane.

While extremely useful, in some settings the above regression model may not be entirely realistic. For example, it is quite natural that the matrix $A$ may also have been corrupted by measurement noise. In this case, one should also be allowed to first change entries of $A$, obtaining a new $m \times n$ matrix $\widehat{A}$, then try to fit $B$ to $\widehat{A}$ by solving a multiple-response regression problem. One should again be penalized for how much one changes the entries of $A$, and this leads to a popular formulation known as the *total least squares* optimization problem $\min_{\widehat{A},X} \|A - \widehat{A}\|_F^2 + \|\widehat{A}X - B\|_F^2$. Letting $C = [A, B]$, one can more compactly write this objective as $\min_{\widehat{C}=[\widehat{A},\widehat{B}]} \|C - \widehat{C}\|_F^2$, where it is required that the

columns of $\widehat{B}$ are in the column span of $\widehat{A}$. Total least squares can naturally capture many scenarios that least squares cannot. For example, imagine a column of $B$ is a large multiple $\lambda \cdot a$ of a column $a$ of $A$ that has been corrupted and sent to 0. Then in least squares, one needs to pay $\lambda^2 \|a\|_2^2$, but in total least squares one can "repair $A$" to contain the column $a$, and just pay $\|a\|_2^2$. We refer the reader to [99] for an overview of total least squares. There is also a large amount of work on total least squares with regularization [117, 98, 89].

Notice that $\widehat{C}$ has rank $n$, and therefore the optimal cost is at least $\|C - C_n\|_F^2$, where $C_n$ is the best rank-$n$ approximation to $C$. If, in the optimal rank-$n$ approximation $C_n$, one has the property that the last $d$ columns are in the column span of the first $n$ columns, then the optimal solution $\widehat{C}$ to total least squares problem is equal to $C_n$, and so the total least squares cost is the cost of the best rank-$n$ approximation to $C$. In this case, and only in this case, there is a closed-form solution. However, in general, this need not be the case, and $\|C - C_n\|_F^2$ may be strictly smaller than the total least squares cost. Fortunately, though, it cannot be much smaller, since one can take the first $n$ columns of $C_n$, and for each column that is not linearly independent of the remaining columns, we can replace it with an arbitrarily small multiple of one of the last $d$ columns of $C_n$ which is not in the span of the first $n$ columns of $C_n$. Iterating this procedure, we find that there is a solution to the total least squares problem which has cost which is arbitrarily close to $\|C - C_n\|_F^2$. We describe this procedure in more detail below.

The above procedure of converting a best rank-$n$ approximation to an arbitrarily close solution to the total least squares problem can be done efficiently given $C_n$, so this shows that one can get an arbitrarily good approximation by computing a truncated singular value decomposition (SVD), which is a standard way of solving for $C_n$ in $O(m(n+d)^2)$ time. However, given the explosion of large-scale datasets these days, this running time is often prohibitive, even for the simpler problem of multiple response least squares regression. Motivated by this, an emerging body of literature has looked at the *sketch-and-solve* paradigm, where one settles for randomized approximation algorithms which run in much faster, often input sparsity time. Here by input-sparsity, we mean in time

linear in the number $\mathsf{nnz}(C)$ of non-zero entries of the input description $C = [A, B]$. By now, it is known, for example, how to output a solution matrix $X$ to multiple response least squares regression satisfying $\|AX - B\|_F^2 \leqslant (1+\varepsilon)\min_{X'} \|AX' - B\|_F^2$, in $\mathsf{nnz}(A) + \mathsf{nnz}(B) + \mathsf{poly}(nd/\varepsilon)$ time. This algorithm works for arbitrary input matrices $A$ and $B$, and succeeds with high probability over the algorithm's random coin tosses. For a survey of this and related results, we refer the reader to [153].

Given the above characterization of total least squares as a low rank approximation problem, it is natural to ask if one can directly apply sketch-and-solve techniques to solve it. Indeed, for low rank approximation, it is known how to find a rank-$k$ matrix $\widehat{C}$ for which $\|C - \widehat{C}\|_F^2 \leqslant (1+\varepsilon)\|C - C_k\|_F^2$ in time $\mathsf{nnz}(C) + m \cdot k^2/\varepsilon$ , using the fastest known results [7]. Here, recall, we assume $m \geqslant n + d$. From an approximation point of view, this is fine for the total least squares problem, since this means after applying the procedure above to ensure the last $d$ columns of $\widehat{C}$ are in the span of the first $n$ columns, and setting $k = n$ in the low rank approximation problem, our cost will be at most $(1+\varepsilon)\|C - C_n\|_F^2 + \eta$, where $\eta$ can be made an arbitrarily small function of $n$. Moreover, the optimal total least squares cost is at least $\|C - C_n\|_F^2$, so our cost is a $(1+\varepsilon)$-relative error approximation, up to an arbitrarily small additive $\eta$.

Unfortunately, this approach is insufficient for total least squares, because in the total least squares problem one sets $k = n$, and so the running time for approximate low rank approximation becomes $\mathsf{nnz}(C) + m \cdot n^2/\varepsilon$. Since $n$ need not be that small, the $m \cdot n^2$ term is potentially prohibitively large. Indeed, if $d \leqslant n$, this may be much larger than the description of the input, which requires at most $mn$ parameters. Note that just outputting $\widehat{C}$ may take $m \cdot (n + d)$ parameters to describe. However, as in the case of regression, one is often just interested in the matrix $X$ or the hyperplane $x$ for ordinary least squares regression. Here the matrix $X$ for total least squares can be described using only $nd$ parameters, and so one could hope for a much faster running time.

### 6.1.1 Our Contributions

Our main contribution is to develop a $(1+\varepsilon)$-approximation to the total least squares regression problem, returning a matrix $X \in \mathbb{R}^{n\times d}$ for which there exist $\widehat{A} \in \mathbb{R}^{m\times n}$ and $\widehat{B} \in \mathbb{R}^{m\times d}$ for which $\widehat{A}X = \widehat{B}$ and $\|C - \widehat{C}\|_F^2 \leqslant (1+\varepsilon)\|C - C_n\|_F^2 + \eta$, where $C = [A, B]$, $\widehat{C} = [\widehat{A}, \widehat{B}]$, $C_n$ is the best rank-$n$ approximation to $C$, and $\eta$ is an arbitrarily small function of $n$. Importantly, we achieve a running time of $\widetilde{O}(\mathsf{nnz}(A) + \mathsf{nnz}(B)) + \mathsf{poly}(n/\varepsilon)\cdot d$.

Notice that this running time may be faster than the time it takes *even to write down* $\widehat{A}$ *and* $\widehat{B}$. Indeed, although one can write $A$ and $B$ down in $\mathsf{nnz}(A) + \mathsf{nnz}(B)$ time, it could be that the algorithm can only efficiently find an $\widehat{A}$ and a $\widehat{B}$ that are dense; nevertheless the algorithm does not need to write such matrices down, as it is only interested in outputting the solution $X$ to the equation $\widehat{A}X = \widehat{B}$. This is motivated by applications in which one wants generalization error. Given $X$, and a future $y \in \mathbb{R}^n$, one can compute $yX$ to predict the remaining unknown $d$ coordinates of the extension of $y$ to $n+d$ dimensions.

Our algorithm is inspired by the use of dimensionality reduction techniques for low rank approximation, such as fast oblivious "sketching" matrices, as well as leverage score sampling. The rough idea is to quickly reduce the low rank approximation problem to a problem of the form $\min_{\text{rank - }n\ Z\in\mathbb{R}^{d_1\times s_1}} \|(D_2CD_1)Z(S_1C) - D_2C\|_F$, where $d_1, s_1 = O(n/\varepsilon)$, $D_2$ and $D_1$ are row and column subset selection matrices, and $S_1$ is a so-called CountSketch matrix, which is a fast oblivious projection matrix. We describe the matrices $D_1, D_2$, and $S_1$ in more detail in the next section, though the key takeaway message is that $(D_2CD_1)$, $(S_1C)$, and $(D_2C)$ are each efficiently computable small matrices *with a number of non-zero entries no larger than that of* $C$. Now the problem is a small, rank-constrained regression problem for which there are closed form solutions for $Z$. We then need additional technical work, of the form described above, in order to find an $X \in \mathbb{R}^{n\times d}$ given $Z$, and to ensure that $X$ is the solution to an equation of the form $\widehat{A}X = \widehat{B}$. Surprisingly, fast sketching methods have not been applied to the total least squares problem before, and we consider this application to be one of the main contributions of this work.

We carefully bound the running time at each step to achieve $\widetilde{O}(\mathsf{nnz}(A) + \mathsf{nnz}(B) + \mathsf{poly}(n/\varepsilon)d)$ overall time, and prove its overall approximation ratio. Our main result is Theorem 6.5.10.

We also generalize the theorem to the important case of total least squares regression with *regularization*; see Theorem 6.6.7 for a precise statement.

We empirically validate our algorithm on real and synthetic data sets. As expected, on a number of datasets the total least squares error can be much smaller than the error of ordinary least squares regression. We then implement our fast total least squares algorithm, and show it is roughly $20 - 40$ times faster than computing the exact solution to total least squares, while retaining 95% accuracy.

### *6.2 Notation*

For a function $f$, we define $\widetilde{O}(f)$ to be $f \cdot \log^{O(1)}(f)$. In addition to $O(\cdot)$ notation, for two functions $f, g$, we use the shorthand $f \lesssim g$ (resp. $\gtrsim$) to indicate that $f \leqslant Cg$ (resp. $\geqslant$) for an absolute constant $C$. We use $f \eqsim g$ to mean $cf \leqslant g \leqslant Cf$ for constants $c, C$.

For vectors $x, y \in \mathbb{R}^n$, let $\langle x, y\rangle := \sum_{i=1}^n x_i y_i$ denote the inner product of $x$ and $y$. Let $\mathsf{nnz}(A)$ denote the number of nonzero entries of $A$. Let $A^\top$ denote the transpose of $A$. Let $A^\dagger$ denote the Moore-Penrose pseudoinverse of $A$ defined in Section 2.4, in contrast to true inverse $A^{-1}$ which only is defined when A is a full rank square matrix. Let $\|A\|_F$ denote the Frobenius norm of a matrix $A$, i. e., $\|A\|_F = (\sum_i \sum_j A_{i,j}^2)^{1/2}$.

### *6.3 Preliminaries*

Sketching matrices play an important role in our algorithm. Their usefulness will be further explained in Section 6.5. Here we present some detailed introduction.

### 6.3.1 Oblivious and Non-oblivious sketching matrices

In this section we introduce techniques in sketching. In order to optimize performance, we introduce multiple types of sketching matrices, which are used in Section 6.5. In Section 6.3.2, we provide the definition of CountSketch and Gaussian Transforms. In Section 6.3.3, we introduce leverage scores and sampling based on leverage scores.

### 6.3.2 CountSketch and Gaussian Transforms

The CountSketch matrix comes from the data stream literature [25, 139].

**Definition 6.3.1** (Sparse embedding matrix or CountSketch transform)**.** *A CountSketch transform is defined to be $\Pi = \Phi D \in \mathbb{R}^{m \times n}$ where $D$ and $\Phi$ are defined as follows: $D$ is an $n \times n$ random diagonal matrix with each diagonal entry independently chosen to be $+1$ or $-1$ with equal probability, and $\Phi \in \{0,1\}^{m \times n}$ is an $m \times n$ binary matrix with $\Phi_{h(i),i} = 1$ and all remaining entries $0$, where $h : [n] \to [m]$ is a random map such that for each $i \in [n]$, $h(i) = j$ with probability $1/m$ for each $j \in [m]$. For any matrix $A \in \mathbb{R}^{n \times d}$, $\Pi A$ can be computed in $O(\mathsf{nnz}(A))$ time.*

To obtain the optimal number of rows, we use a Gaussian matrix, which is another well-known oblivious sketching matrix.

**Definition 6.3.2** (Gaussian matrix or Gaussian transform)**.** *Let $S = \frac{1}{\sqrt{m}} \cdot G \in \mathbb{R}^{m \times n}$ where each entry of $G \in \mathbb{R}^{m \times n}$ is chosen independently from the standard Gaussian distribution. For any matrix $A \in \mathbb{R}^{n \times d}$, $SA$ can be computed in $O(m \cdot \mathsf{nnz}(A))$ time.*

We can combine CountSketch and Gaussian transforms to achieve the following:

**Definition 6.3.3** (CountSketch + Gaussian transform)**.** *Let $S' = S\Pi$, where $\Pi \in \mathbb{R}^{t \times n}$ is the CountSketch transform (defined in Definition 6.3.1) and $S \in \mathbb{R}^{m \times t}$ is the Gaussian transform (defined in Definition 6.3.2). For any matrix $A \in \mathbb{R}^{n \times d}$, $S'A$ can be computed in $O(\mathsf{nnz}(A) + dtm^{\omega-2})$ time, where $\omega \approx 2.373$ is the matrix multiplication exponent [39, 152], that is, multiplication of two $n$ by $n$ matrices can be done in $O(n^{\omega})$ time.*

### 6.3.3 Leverage Scores

We note that there are other ways of constructing sketching matrix, such as through sampling the rows of $A$ via a certain distribution and reweighting them. This is called *leverage score sampling* [52, 51, 53]. We first give the concrete definition of leverage scores.

**Definition 6.3.4** (Leverage scores)**.** *Let $U \in \mathbb{R}^{n\times k}$ have orthonormal columns with $n \geqslant k$. We will use the notation $p_i = u_i^2/k$, where $u_i^2 = \|e_i^\top U\|_2^2$ is referred to as the $i$-th leverage score of $U$.*

Next we explain the leverage score sampling. Given $A \in \mathbb{R}^{n\times d}$ with rank $k$, let $U \in \mathbb{R}^{n\times k}$ be an orthonormal basis of the column span of $A$, and for each $i$ let $k \cdot p_i$ be the squared row norm of the $i$-th row of $U$. Let $p_i$ denote the $i$-th leverage score of $U$. Let $\beta > 0$ be a constant and $q = (q_1, \cdots, q_n)$ denote a distribution such that, for each $i \in [n]$, $q_i \geqslant \beta p_i$. Let $s$ be a parameter. Construct an $n \times s$ sampling matrix $B$ and an $s \times s$ rescaling matrix $D$ as follows. Initially, $B = 0^{n\times s}$ and $D = 0^{s\times s}$. For the same column index $j$ of $B$ and of $D$, independently, and with replacement, pick a row index $i \in [n]$ with probability $q_i$, and set $B_{i,j} = 1$ and $D_{j,j} = 1/\sqrt{q_i s}$. We denote this procedure LEVERAGE SCORE SAMPLING according to the matrix $A$.

Leverage score sampling is efficient in the sense that the leverage scores can be efficiently approximated.

**Theorem 6.3.5** (Running time of over-estimation of leverage score, Theorem 14 in [105])**.** *For any $\varepsilon > 0$, with probability at least $2/3$, we can compute a $1 \pm \varepsilon$ approximation of all leverage scores of matrix $A \in \mathbb{R}^{n\times d}$ in time $\widetilde{O}(\mathsf{nnz}(A) + r^\omega \varepsilon^{-2\omega})$ where $r$ is the rank of $A$ and $\omega \approx 2.373$ is the matrix multiplication exponent.*

In Section 6.3.4 we show how to apply matrix sketching to solve regression problems faster. In Section 6.3.5, we give a structural result on rank-constrained approximation problems.

### 6.3.4 Multiple Regression

Linear regression is a fundamental problem in Machine Learning. There have been many attempts trying to speed up the running time of different kind of linear regression problems via sketching matrices [30, 101, 110, 95, 47, 4, 29]. A natural generalization of linear regression is multiple regression.

We first show how to use CountSketch to reduce the dimensionality of a multiple regression problem:

**Theorem 6.3.6** (Multiple regression, [153])**.** *Given $A \in \mathbb{R}^{n\times d}$ and $B \in \mathbb{R}^{n\times m}$, let $S \in \mathbb{R}^{s\times n}$ denote a sampling and rescaling matrix according to $A$. Let $X^*$ denote $\arg\min_X \|AX - B\|_F^2$ and $X'$ denote $\arg\min_X \|SAX - SB\|_F^2$. If $S$ has $s = O(d/\varepsilon)$ rows, then we have that*

$$\|AX' - B\|_F^2 \leqslant (1+\varepsilon)\|AX^* - B\|_F^2$$

*holds with probability at least* 0.999*.*

The following theorem says that leverage score sampling solves multiple response regression:

**Theorem 6.3.7** (See, e.g., the combination of Corollary C.30 and Lemma C.31 in [130])**.** *Given $A \in \mathbb{R}^{n\times d}$ and $B \in \mathbb{R}^{n\times m}$, let $D \in \mathbb{R}^{n\times n}$ denote a sampling and rescaling matrix according to $A$. Let $X^*$ denote $\arg\min_X \|AX - B\|_F^2$ and $X'$ denote $\arg\min_X \|DAX - SB\|_F^2$. If $D$ has $O(d\log d + d/\varepsilon)$ non-zeros in expectation, that is, this is the expected number of sampled rows, then we have that*

$$\|AX' - B\|_F^2 \leqslant (1+\varepsilon)\|AX^* - B\|_F^2$$

*holds with probability at least* 0.999*.*

### 6.3.5 Generalized Rank-Constrained Matrix Approximation

We now state a tool which has been used in several recent works [21, 128, 130].

**Theorem 6.3.8** (Generalized rank-constrained matrix approximation, Theorem 2 in [55])**.** *Given matrices $A \in \mathbb{R}^{n\times d}$, $B \in \mathbb{R}^{n\times p}$, and $C \in \mathbb{R}^{q\times d}$, let the singular value decomposition (SVD) of $B$ be $B = U_B\Sigma_B V_B^\top$ and the SVD of $C$ be $C = U_C\Sigma_C V_C^\top$. Then*

$$B^\dagger(U_B U_B^\top A V_C V_C^\top)_k C^\dagger = \underset{\text{rank}-k\ X\in\mathbb{R}^{p\times q}}{\arg\min} \|A - BXC\|_F$$

*where $(U_B U_B^\top A V_C V_C^\top)_k \in \mathbb{R}^{n\times d}$ is of rank at most $k$ and denotes the best* rank-$k$ *approximation to $U_B U_B^\top A V_C V_C^\top \in \mathbb{R}^{n\times d}$ in Frobenius norm.*

*Moreover, $(U_B U_B^\top A V_C V_C^\top)_k$ can be computed by first computing the SVD decomposition of $U_B U_B^\top A V_C V_C^\top$ in time $O(nd^2)$, then only keeping the largest $k$ coordinates. Hence $B^\dagger(U_B U_B^\top A V_C V_C^\top)_k C^\dagger$ can be computed in $O(nd^2 + np^2 + qd^2)$ time.*

### *6.4 Problem Formulation*

We first give the precise definition of the exact (i. e., non-approximate) version of the total least squares problem, and then define the approximate case.

**Definition 6.4.1** (Exact total least squares)**.** *Given two matrices $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{m\times d}$, let $C = [A,\ B] \in \mathbb{R}^{m\times(n+d)}$. The goal is to solve the following minimization problem:*

$$\min_{X\in\mathbb{R}^{n\times d}, \Delta A\in\mathbb{R}^{m\times n}, \Delta B\in\mathbb{R}^{m\times d}} \|[\Delta A,\ \Delta B]\|_F \tag{6.1}$$
$$\text{subject to } (A+\Delta A)X = (B+\Delta B)$$

Markovsky and Huffel [99] propose the following alternative formulation of total least squares problem.

$$\min_{\text{rank}-n\ C'\in\mathbb{R}^{m\times(n+d)}} \|C' - C\|_F \tag{6.2}$$

When program (6.1) has a solution $(X, \Delta A, \Delta B)$, we can see that (6.1) and (6.2) are in general equivalent by setting $C' = [A + \Delta A,\ B + \Delta B]$. However, there are cases when program (6.1) fails to have a solution, while (6.2) always has a solution.

As discussed, a solution to the total least squares problem can sometimes be written in closed form. Letting $C = [A,\ B]$, denote the singular value decomposition (SVD) of $C$ by $U\Sigma V^\top$, where $\Sigma = \mathsf{diag}(\sigma_1, \cdots, \sigma_{n+d}) \in \mathbb{R}^{m\times(n+d)}$ with $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_{n+d}$. Also we represent $(n+d)\times(n+d)$ matrix $V$ as $\begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix}$ where $V_{11} \in \mathbb{R}^{n\times n}$ and $V_{22} \in \mathbb{R}^{d\times d}$.

Clearly $\widehat{C} = U\,\mathsf{diag}(\sigma_1, \cdots, \sigma_n, 0, \cdots, 0)V^\top$ is a minimizer of program (6.2). But whether a solution to program (6.1) exists depends on the singularity of $V_{22}$. We introduce different cases of the solution to program (6.1), and discuss how our algorithm deals with each case.

### *6.4.1 Unique Solution*

We first consider the case when the Total Least Squares problem has a unique solution.

**Theorem 6.4.2** (Theorem 2.6 and Theorem 3.1 in [147])**.** *If $\sigma_n > \sigma_{n+1}$, and $V_{22}$ is non-singular, then the minimizer $\widehat{C}$ is given by $U\,\mathsf{diag}(\sigma_1, \cdots, \sigma_n, 0, \cdots, 0)V^\top$, and the optimal solution $\widehat{X}$ is given by $-V_{12}V_{22}^{-1}$.*

Our algorithm will first find a rank $n$ matrix $C' = [A',\ B']$ so that $\|C' - C\|_F$ is small, then solve a regression problem to find $X'$ so that $A'X' = B'$. In this sense, this is the most favorable case to work with, because a unique optimal solution $\widehat{C}$ exists, so if $C'$ approximates $\widehat{C}$ well, then the regression problem $A'X' = B'$ is solvable.

### *6.4.2 Solution exists, but is not unique*

If $\sigma_n = \sigma_{n+1}$, then it is still possible that the Total Least Squares problem has a unique solution, although this time, the solution $\widehat{X}$ is not unique. Theorem 6.4.3 is a generalization of Theorem 6.4.2.

**Theorem 6.4.3** (Theorem 3.9 in [147])**.** *Let $p \leqslant n$ be a number so that $\sigma_p > \sigma_{p+1} = \cdots = \sigma_{n+1}$. Let $V_p$ be the submatrix that contains the last $d$ rows and the last $n - p + d$ columns of $V$.*

*If $V_p$ is non-singular, then multiple minimizers $\widehat{C} = [\widehat{A},\ \widehat{B}]$ exist, and there exists $\widehat{X} \in \mathbb{R}^{n\times d}$ so that $\widehat{A}\widehat{X} = \widehat{B}$.*

We can also handle this case. As long as the Total Least Squares problem has a solution $\widehat{X}$, we are able to approximate it by first finding $C' = [A',\ B']$ and then solving a regression problem.

### *6.4.3 When a solution does not exist*

Notice that the cost $\|\widehat{C} - C\|_F^2$, where $\widehat{C}$ is the optimal solution to program (6.2), always lower bounds the cost of program (6.1). But there are cases where this cost is not approachable in program (6.1).

**Theorem 6.4.4** (Lemma 3.2 in [147]). *If $V_{22}$ is singular, letting $\widehat{C}$ denote $[\widehat{A},\ \widehat{B}]$, then $\widehat{A}X = \widehat{B}$ has no solution.*

Theorem 6.4.4 shows that even if we can compute $\widehat{C}$ precisely, we cannot output $X$, because the first $n$ columns of $\widehat{C}$ cannot span the rest $d$ columns. In order to generate a meaningful result, our algorithm will perturb $C'$ by an arbitrarily small amount so that $A'X' = B'$ has a solution. This will introduce an arbitrarily small additive error in addition to our relative error guarantee.

It is natural to consider the approximate version of total least squares:

**Definition 6.4.5** (Approximate total least squares problem). *Given two matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{m\times d}$, let $\mathrm{OPT} = \min_{\mathrm{rank}-n\ C'} \|C' - [A,\ B]\|_F$, for parameters $\varepsilon > 0, \delta > 0$. The goal is to output $X' \in \mathbb{R}^{n\times d}$ so that there exists $A' \in \mathbb{R}^{m\times n}$ such that*

$$\|[A',\ A'X'] - [A,\ B]\|_F \leqslant (1+\varepsilon)\,\mathrm{OPT} + \delta.$$

One could solve total least squares directly, but it is much slower than solving least squares (LS). We will use fast randomized algorithms, the basis of which are sampling and sketching ideas [30, 105, 101, 153, 115, 110, 128, 33, 37, 94, 126, 129, 130, 131, 46], to speed up solving total least squares in both theory and in practice.

**Algorithm 1:** Least Squares and Total Least Squares Algorithms

**Function** LEASTSQUARES$(A, B)$**:**
  $X \leftarrow \min_X \|AX - B\|_F$;
  $C_{\text{LS}} \leftarrow [A,\ AX]$;
  **return** $C_{\text{LS}}$
**Function** TOTALLEASTSQUARES$(A, B)$**:**
  $C_{\text{TLS}} \leftarrow \min_{\text{rank}-n\ C'} \|C - C'\|_F$;
  **return** $C_{\text{TLS}}$

The total least squares problem with regularization is also an important variant of this problem [88]. We consider the following version of the regularized total least squares problem.

**Definition 6.4.6** (Approximate regularized total least squares problem)**.** *Given two matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{m\times d}$ and $\lambda > 0$, let* $\text{OPT} = \min_{U\in\mathbb{R}^{m\times n}, V\in\mathbb{R}^{n\times(n+d)}} \|UV - [A,\ B]\|_F^2 + \lambda\|U\|_F^2 + \lambda\|V\|_F^2$*, for parameters $\varepsilon > 0, \delta > 0$. The goal is to output $X' \in \mathbb{R}^{n\times d}$ so that there exist $A' \in \mathbb{R}^{m\times n}$ , $U' \in \mathbb{R}^{m\times n}$ and $V' \in \mathbb{R}^{n\times(n+d)}$ satisfying $\|[A',\ A'X'] - U'V'\|_F^2 \leqslant \delta$ and $\|[A',\ A'X'] - [A,\ B]\|_F^2 \leqslant (1+\varepsilon)\,\text{OPT} + \delta$.*

Table 6.1: Notations in Algorithm 2

| Not. | Value | Comment | Matrix | Dim. | Comment |
|---|---|---|---|---|---|
| $s_1$ | $O(n/\varepsilon)$ | #rows in $S_1$ | $S_1$ | $\mathbb{R}^{s_1\times m}$ | CountSketch matrix |
| $d_1$ | $\widetilde{O}(n/\varepsilon)$ | #columns in $D_1$ | $D_1$ | $\mathbb{R}^{n\times d_1}$ | Leverage score sampling matrix |
| $d_2$ | $\widetilde{O}(n/\varepsilon)$ | #rows in $D_2$ | $D_2$ | $\mathbb{R}^{d_2\times m}$ | Leverage score sampling matrix |
| $s_2$ | $O(n/\varepsilon)$ | #rows in $S_2$ | $S_2$ | $\mathbb{R}^{s_2\times m}$ | CountSketch matrix for fast regression |
| | | | $Z_2$ | $\mathbb{R}^{s_1\times d_1}$ | Low rank approximation solution matrix |

### *6.5 Fast Total Least Squares Algorithm*

We present our algorithm in Algorithm 2 and give the analysis here. Readers can refer to Table 6.1 to check notations in Algorithm 2. To clearly give the intuition, we present a sequence of approximations, reducing the size of our problem step-by-step. We can focus on the case when $d \gg \Omega(n/\varepsilon)$ and the optimal solution $\widehat{C}$ to program (6.2) has the form $[\widehat{A},\ \widehat{A}\widehat{X}] \in \mathbb{R}^{m\times(n+d)}$. For the other case when $d = O(n/\varepsilon)$, we do not need to use the sampling matrix $D_1$. In the case when the solution does not have the form $[\widehat{A},\ \widehat{A}\widehat{X}]$, we need to include SPLIT in the algorithm, since it will perturb some columns in $\widehat{A}$ with arbitrarily small noise to make sure $\widehat{A}$ has rank $n$. By applying procedure SPLIT, we can handle all cases.

**Algorithm 2:** Our Fast Total Least Squares Algorithm

**Function** FASTTOTALLEASTSQUARES($A, B, n, d, \varepsilon, \delta$)**:**

  $s_1 \leftarrow O(n/\varepsilon)$, $s_2 \leftarrow O(n/\varepsilon)$, $d_1 \leftarrow \widetilde{O}(n/\varepsilon)$, $d_2 \leftarrow \widetilde{O}(n/\varepsilon)$;

  Choose $S_1 \in \mathbb{R}^{s_1\times m}$ to be a CountSketch matrix and compute $S_1C$ ▷ Definition 6.3.1

  **if** $d > \Omega(n/\varepsilon)$ **then**

    `/* Reduce` $n+d$ `to` $O(n/\varepsilon)$. `*/`

    Choose $D_1^\top \in \mathbb{R}^{d_1\times(n+d)}$ to be a leverage score sampling and rescaling matrix according to the rows of $(S_1C)^\top$, then compute $CD_1$;

  **else**

    `/* We do not need to use matrix` $D_1$ `*/`

    Choose $D_1^\top \in \mathbb{R}^{(n+d)\times(n+d)}$ to be the identity matrix;

  Choose $D_2 \in \mathbb{R}^{d_2\times m}$ to be a leverage score sampling and rescaling matrix according to the rows of $CD_1$;

  $Z_2 \leftarrow \min_{\mathrm{rank}-n\ Z\in\mathbb{R}^{d_1\times s_1}} \|D_2CD_1ZS_1C - D_2C\|_F$ ▷ Theorem 6.3.8

10

11 $\overline{A}, \overline{B}, \pi \leftarrow \textsc{Split}(CD_1, Z_2, S_1C, n, d, \delta/\mathsf{poly}(m))$, $X \leftarrow \min \|\overline{A}X - \overline{B}\|_F$;

12 **if** *Need* $C_{\mathrm{FTLS}}$ **then**

`/* For experiments to evaluate the cost */`

13 $\textsc{Evaluate}(CD_1, Z_2, S_1C, X, \pi, \delta/\mathsf{poly}(m))$

14 **return** $X$;

15 **Function** $\textsc{Split}(CD_1, Z_2, S_1C, n, d, \delta)$**:**

`/* Lemma 6.5.8 */`

16 Choose $S_2 \in \mathbb{R}^{s_2 \times m}$ to be a CountSketch matrix;

17 $\overline{C} \leftarrow (S_2 \cdot CD_1) \cdot Z_2 \cdot S_1C$ ▷ $\widehat{C} = CD_1Z_2S_1C$ ; $\overline{C} = S_2\widehat{C}$;

18 $\overline{A} \leftarrow \overline{C}_{*,[n]}$, $\overline{B} \leftarrow \overline{C}_{*,[n+d]\setminus[n]}$ ▷ $\widehat{A} = \widehat{C}_{*,[n]}, \widehat{B} = \widehat{C}_{*,[n+d]\setminus[n]}$; $\overline{A} = S_2\widehat{A}, \overline{B} = S_2\widehat{B}$;

19 $T \leftarrow \emptyset$, $\pi(i) = -1$ for all $i \in [n]$;

20 **for** $i = 1 \to n$ **do**

21 **if** $\overline{A}_{*,i}$ *is linearly dependent of* $\overline{A}_{*,[n]\setminus\{i\}}$ **then**

22 $j \leftarrow \min_{j \in [d]\setminus T}\{\overline{B}_{*,j}$ is linearly independent of $\overline{A}\}$, $\overline{A}_{*,i} \leftarrow \overline{A}_{*,i} + \delta \cdot \overline{B}_{*,j}$, $T \leftarrow T \cup \{j\}$, $\pi(i) \leftarrow j$

23 **return** $\overline{A}, \overline{B}, \pi$ ▷ $\pi : [n] \to \{-1\} \cup ([n+d]\setminus[n])$;

24 **Function** $\textsc{Evaluate}(CD_1, Z_2, S_1C, X, \pi, \delta)$**:**

25 $\widehat{C} \leftarrow CD_1Z_2S_1C$, $\widehat{A} \leftarrow \widehat{C}_{*,[n]}$, $\widehat{B} \leftarrow \overline{C}_{*,[n+d]\setminus[n]}$;

26 **for** $i = 1 \to n$ **do**

27 **if** $\pi(i) \neq -1$ **then**

28 $\widehat{A}_{*,i} \leftarrow \widehat{A}_{*,i} + \delta \cdot \widehat{B}_{*,\pi(i)}$

29 **return** $\|[\widehat{A},\ \widehat{A}X] - C\|_F$

Fix $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{m\times d}$. Let $\mathrm{OPT} = \min_{\mathrm{rank}-n\ C'\in\mathbb{R}^{m\times(n+d)}} \|C' - [A,\ B]\|_F$. By using techniques in low-rank approximation, we can find an approximation of a special form. More precisely, let $S_1 \in \mathbb{R}^{s_1\times m}$ be a CountSketch matrix with $s_1 = O(n/\varepsilon)$. Then we claim that it is sufficient to look at solutions of the form $US_1C$.

**Claim 6.5.1** (CountSketch matrix for low rank approximation problem)**.** *With probability* 0.98,

$$\min_{\mathrm{rank}-n\ U\in\mathbb{R}^{m\times s_1}} \|US_1C - C\|_F^2 \leqslant (1+\varepsilon)^2\,\mathrm{OPT}^2\,.$$

We shall mention that we cannot use leverage score sampling here, because taking leverage score sampling on matrix $C$ would take at least $\mathsf{nnz}(C) + (n+d)^2$ time, while we are linear in $d$ in the additive term in our running time $\widetilde{O}(\mathsf{nnz}(C)) + d\cdot\mathsf{poly}(n/\varepsilon)$.

*Proof.* Let $C^*$ be the optimal solution of $\min_{\mathrm{rank}-n\ C'\in\mathbb{R}^{m\times(n+d)}} \|C' - [A,\ B]\|_F$. Since $\mathrm{rank}(C^*) = n \ll m$, there exist $U^* \in \mathbb{R}^{m\times s_1}$ and $V^* \in \mathbb{R}^{s_1\times(n+d)}$ so that $C^* = U^*V^*$, and $\mathrm{rank}(U^*) = \mathrm{rank}(V^*) = n$. Therefore

$$\min_{V\in\mathbb{R}^{s_1\times(n+d)}} \|U^*V - C\|_F^2 = \mathrm{OPT}^2\,.$$

Now consider the problem formed by multiplying by $S_1$ on the left,

$$\min_{V\in\mathbb{R}^{s_1\times(n+d)}} \|S_1U^*V - S_1C\|_F^2.$$

Letting $V'$ be the minimizer to the above problem, we have

$$V' = (S_1U^*)^\dagger S_1C.$$

Thus, we have

$$\begin{aligned}\min_{\text{rank}-n\ U\in\mathbb{R}^{m\times s_1}} \|US_1C - C\|_F^2 &\leqslant \|U^*(S_1U^*)^\dagger S_1C - C\|_F^2 \\ &= \|U^*V' - C\|_F^2 \\ &\leqslant (1+\varepsilon)\|S_1U^*V' - S_1C\|_F^2 \\ &\leqslant (1+\varepsilon)\|S_1U^*V^* - S_1C\|_F^2 \\ &\leqslant (1+\varepsilon)^2\|U^*V^* - C\|_F^2 \\ &= (1+\varepsilon)^2\,\mathrm{OPT}^2\end{aligned}$$

where the first step uses the fact that $U^*(S_1U^*)^\dagger S_1 \in \mathbb{R}^{m\times s_1}$ with rank $n$, the second step is the definition of $V'$, the third step follows from the definition of the CountSketch matrix $S_1$ and Theorem 6.3.6, the fourth step uses the optimality of $V'$, and the fifth step again uses Theorem 6.3.6. □

Let $U_1$ be the optimal solution of the program $\min_{U\in\mathbb{R}^{m\times s_1}} \|US_1C - C\|_F^2$, i. e.,

$$U_1 = \arg\min_{\text{rank}-n\ U\in\mathbb{R}^{m\times s_1}} \|US_1C - C\|_F^2. \tag{6.3}$$

If $d$ is large compared to $n$, then program (6.3) is computationally expensive to solve. So we can apply sketching techniques to reduce the size of the problem. Let $D_1^\top \in \mathbb{R}^{d_1\times(n+d)}$ denote a leverage score sampling and rescaling matrix according to the columns of $S_1C$, with $d_1 = \widetilde{O}(n/\varepsilon)$ nonzero entries on the diagonal of $D_1$. Let $U_2 \in \mathbb{R}^{m\times s_1}$ denote the optimal solution to the problem $\min_{\text{rank}-n\ U\in\mathbb{R}^{m\times s_1}} \|US_1CD_1 - CD_1\|_F^2$, i. e.,

$$U_2 = \arg\min_{\text{rank}-n\ U\in\mathbb{R}^{m\times s_1}} \|US_1CD_1 - CD_1\|_F^2. \tag{6.4}$$

Then the following claim comes from the constrained low-rank approximation result (Theorem 6.3.8).

**Claim 6.5.2** (Solving regression with leverage score sampling)**.** *Let $U_1$ be defined in Eq.* (6.3)*, and let $U_2$ be defined in Eq.* (6.4)*. Then with probability* 0.98*,*

$$\|U_2S_1C - C\|_F^2 \leqslant (1+\varepsilon)^2\|U_1S_1C - C\|_F^2.$$

*Proof.* We have

$$\begin{aligned}\|U_2S_1C - C\|_F^2 &\leqslant (1+\varepsilon)\|U_2S_1CD_1 - CD_1\|_F^2 \\ &\leqslant (1+\varepsilon)\|U_1S_1CD_1 - CD_1\|_F^2 \\ &\leqslant (1+\varepsilon)^2\|U_1S_1C - C\|_F^2,\end{aligned}$$

where the first step uses the property of a leverage score sampling matrix $D_1$, the second step follows from the definition of $U_2$ (i. e., $U_2$ is the minimizer), and the last step follows from the property of the leverage score sampling matrix $D_1$ again. □

We now consider how to solve program (6.4). We observe that

**Claim 6.5.3.** $U_2 \in \mathrm{colspan}(CD_1)$.

We can thus consider the following relaxation: given $CD_1$, $S_1C$ and $C$, solve:

$$\min_{\mathrm{rank}-n\ Z\in\mathbb{R}^{d_1\times s_1}} \|CD_1ZS_1C - C\|_F^2. \tag{6.5}$$

By setting $CD_1Z = U$, we can check that program (6.5) is indeed a relaxation of program (6.4). Let $Z_1$ be the optimal solution to program (6.5). We show the following claim.

**Claim 6.5.4** (Approximation ratio of relaxation)**.** *With probability* 0.98*,*

$$\|CD_1Z_1S_1C - C\|_F^2 \leqslant (1+O(\varepsilon))\,\mathrm{OPT}^2\,.$$

*Proof.* From Claim 6.5.2 we have that $U_2 \in \mathrm{colspan}(CD_1)$. Hence we can choose $Z$ so that $CD_1Z = U_2$. Then by Claim 6.5.1 and Claim 6.5.2, we have

$$\|CD_1ZS_1C - C\|_F^2 = \|U_2S_1C - C\|_F^2 \leqslant (1+\varepsilon)^4\,\mathrm{OPT}^2\,.$$

Since $Z_1$ is the optimal solution, the objective value can only be smaller. □

However, program (6.5) still has a potentially large size, i. e., we need to work with an $m \times d_1$ matrix $CD_1$. To handle this problem, we again apply sketching techniques. Let

$D_2 \in \mathbb{R}^{d_2 \times m}$ be a leverage score sampling and rescaling matrix according to the matrix $CD_1 \in \mathbb{R}^{m \times d_1}$, so that $D_2$ has $d_2 = \widetilde{O}(n/\varepsilon)$ nonzeros on the diagonal. Now, we arrive at the small program that we are going to directly solve:

$$\min_{\text{rank}-n\ Z \in \mathbb{R}^{d_1 \times s_1}} \|D_2 C D_1 Z S_1 C - D_2 C\|_F^2. \tag{6.6}$$

We shall mention that here it is beneficial to apply leverage score sampling matrix because we only need to compute leverage scores of a smaller matrix $CD_1$, and computing $D_2C$ only involves sampling a small fraction of the rows of $C$. On the other hand, if we were to use the CountSketch matrix, then we would need to touch the whole matrix $C$ when computing $D_2C$. Overall, using leverage score sampling at this step can reduce the constant factor of the $\mathsf{nnz}(C)$ term in the running time, and may be useful in practice. Let rank-$n$ $Z_2 \in \mathbb{R}^{d_1 \times s_1}$ be the optimal solution to this problem.

**Claim 6.5.5** (Solving regression with a CountSketch matrix)**.** *With probability* 0.98*,*

$$\|CD_1 Z_2 S_1 C - C\|_F^2 \leqslant (1+\varepsilon)^2 \|CD_1 Z_1 S_1 C - C\|_F^2$$

*Proof.* Recall that $Z_1 = \arg\min_{\text{rank}-n\ Z \in \mathbb{R}^{d_1 \times s_1}} \|CD_1 Z S_1 C - C\|_F^2$. Then we have

$$\begin{aligned}
\|CD_1 Z_2 S_1 C - C\|_F^2 &\leqslant (1+\varepsilon)\|D_2 C D_1 Z_2 S_1 C - D_2 C\|_F^2 \\
&\leqslant (1+\varepsilon)\|D_2 C D_1 Z_1 S_1 C - D_2 C\|_F^2 \\
&\leqslant (1+\varepsilon)^2 \|CD_1 Z_1 S_1 C - C\|_F^2,
\end{aligned}$$

where the first step uses the property of the leverage score sampling matrix $D_2$, the second step follows from the definition of $Z_2$ (i. e., $Z_2$ is a minimizer), and the last step follows from the property of the leverage score sampling matrix $D_2$. □

Our algorithm thus far is as follows: we compute matrices $S_1$, $D_1$, $D_2$ accordingly, then solve program (6.6) to obtain $Z_2$. At this point, we are able to obtain the low rank approximation $\widehat{C} = CD_1 \cdot Z_2 \cdot S_1 C$. We show the following claim.

**Claim 6.5.6** (Analysis of $\widehat{C}$)**.** *With probability* 0.94*,*

$$\|\widehat{C} - C\|_F^2 \leqslant (1 + O(\varepsilon)) \operatorname{OPT}^2 .$$

*Proof.*

$$\begin{aligned} \|\widehat{C} - C\|_F^2 &= \|CD_1 \cdot Z_2 \cdot S_1 C - C\|_F^2 \\ &\leqslant (1+\varepsilon)^2 \|CD_1 Z_1 S_1 C - C\|_F^2 \\ &\leqslant (1 + O(\varepsilon)) \operatorname{OPT}^2 \end{aligned}$$

where the first step is the definition of $\widehat{C}$, the second step is Claim 6.5.5, and the last step is Claim 6.5.4. □

Let $\widehat{C} = [\widehat{A},\ \widehat{B}]$ where $\widehat{A} \in \mathbb{R}^{m \times n}$ and $\widehat{B} \in \mathbb{R}^{m \times d}$. However, if our goal is to only output a matrix $X$ so that $\widehat{A}X = \widehat{B}$, then we can do this faster by not computing or storing the matrix $\widehat{C}$. Let $S_2 \in \mathbb{R}^{s_2 \times m}$ be a CountSketch matrix with $s_2 = O(n/\varepsilon)$. We solve a regression problem:

$$\min_{X \in \mathbb{R}^{n \times d}} \|S_2 \widehat{A} X - S_2 \widehat{B}\|_F^2.$$

Notice that $S_2\widehat{A}$ and $S_2\widehat{B}$ are computed directly from $CD_1$, $Z_2$, $S_1C$ and $S_2$. Let $\overline{X}$ be the optimal solution to the above problem.

**Claim 6.5.7** (Approximation ratio guarantee)**.** *Assume* $\widehat{C} = [\widehat{A},\ \widehat{A}\widehat{X}]$ *for some* $\widehat{X} \in \mathbb{R}^{n \times d}$*. Then with probability at least* 0.9*,*

$$\|[\widehat{A},\ \widehat{A}\overline{X}] - [A,\ B]\|_F^2 \leqslant (1 + O(\varepsilon)) \operatorname{OPT}^2 .$$

*Proof.* By the condition that $\widehat{C} = [\widehat{A}\ \widehat{A}\widehat{X}]$, $\widehat{B} = \widehat{A}\widehat{X}$, hence $\widehat{X}$ is the optimal solution to the program $\min_{X \in \mathbb{R}^{n \times d}} \|\widehat{A}X - \widehat{B}\|_F^2$. Hence by Theorem 6.3.6, with probability at least 0.99,

$$\|\widehat{A}\overline{X} - \widehat{B}\|_F^2 \leqslant (1+\varepsilon)\|\widehat{A}\widehat{X} - \widehat{B}\|_F^2 = 0$$

Therefore

$$\|[\widehat{A},\ \widehat{A}\overline{X}] - [A,\ B]\|_F^2 = \|[\widehat{A},\ \widehat{B}] - C\|_F^2 = \|\widehat{C} - C\|_F^2.$$

Then Claim 6.5.7 follows from Claim 6.5.6. □

If the assumption $\widehat{C} = [\widehat{A},\ \widehat{A}\widehat{X}]$ in Claim 6.5.7 does not hold, then we need to apply procedure SPLIT. Because $\text{rank}(\widehat{C}) = n$ from our construction, if the first $n$ columns of $\widehat{C}$ cannot span the last $d$ columns, then the first $n$ columns of $\widehat{C}$ are not full rank. Hence we can keep adding a sufficiently small multiple of one of the last $d$ columns that cannot be spanned by the first $n$ columns until the first $n$ columns are full rank. Formally, we have

**Lemma 6.5.8** (Analysis of procedure SPLIT)**.** *Fix $s_1 = O(n/\varepsilon)$, $s_2 = O(n/\varepsilon)$, $d_1 = \widetilde{O}(n/\varepsilon)$. Given $CD_1 \in \mathbb{R}^{m\times d_1}$, $Z_2 \in \mathbb{R}^{d_1\times s_1}$, and $S_1C \in \mathbb{R}^{s_1\times(n+d)}$ so that $\widehat{C} := CD_1 \cdot Z_2 \cdot S_1C$ has rank $n$, procedure* SPLIT *(Algorithm 2) returns $\overline{A} \in \mathbb{R}^{s_2\times n}$ and $\overline{B} \in \mathbb{R}^{s_2\times d}$ in time $O(\mathsf{nnz}(C) + d\cdot\mathsf{poly}(n/\varepsilon))$ so that there exists $\overline{X} \in \mathbb{R}^{n\times d}$ satisfying $\overline{A}\cdot\overline{X} = \overline{B}$. Moreover, letting $\widehat{A}$ be the matrix computed in lines* (24) *to* (28)*, then with probability* 0.99*,*

$$\|[\widehat{A},\ \widehat{A}\overline{X}] - C\|_F \leqslant \|\widehat{C} - C\|_F + \delta.$$

*Proof.* **Proof of running time.** Let us first check the running time. We can compute $\overline{C} = S_2 \cdot \widehat{C}$ by first computing $S_2 \cdot CD_1$, then computing $(S_2CD_1)\cdot Z_2$, then finally computing $S_2CD_1Z_2S_1C$. Notice that $D_1$ is a leverage score sampling matrix, so $\mathsf{nnz}(CD_1) \leqslant \mathsf{nnz}(C)$. So by Definition 6.3.1, we can compute $S_2 \cdot CD_1$ in time $O(\mathsf{nnz}(C))$. All the other matrices have smaller size, so we can do matrix multiplication in time $O(d\cdot\mathsf{poly}(n/\varepsilon))$. Once we have $\overline{C}$ as defined in line (16), the independence between columns in $\overline{A}$ can be checked in time $O(s_2\cdot n)$. The FOR loop will be executed at most $n$ times, and inside each loop, line (21) will take at most $d$ linear independence checks. So the running time of the FOR loop is at most $O(s_2\cdot n)\cdot n\cdot d = O(d\cdot\mathsf{poly}(n/\varepsilon))$. Therefore the running time is as desired.

**Proof of Correctness.**

We next argue the correctness of procedure SPLIT. Since $\text{rank}(\widehat{C}) = n$, with high probability $\text{rank}(\overline{C}) = \text{rank}(S_2\cdot\widehat{C}) = n$. Notice that $\overline{B}$ is never changed in this subroutine. In order to show there exists an $X$ so that $\overline{A}X = \overline{B}$, it is sufficient to show that at the end of procedure SPLIT, $\text{rank}(\overline{A}) = \text{rank}(\overline{C})$, because this means that the columns of $\overline{A}$ span each of the columns of $\overline{C}$, including $\overline{B}$. Indeed, whenever $\text{rank}(\overline{A}_{*,[i]}) < i$, line (26) will be executed. Then by doing line (27), the rank of $\overline{A}$ will increase by 1, since by the choice of

$j$, $\overline{A}_{*,i} + \delta \cdot \overline{B}_{*,j}$ is independent form $\overline{A}_{*,[i-1]}$. Because $\mathrm{rank}(\overline{C}) = n$, at the end of the FOR loop we will have $\mathrm{rank}(\overline{A}) = n$.

Finally let us compute the cost. In line (10) we use $\delta / \mathsf{poly}(m)$, and thus

$$\|[\widehat{A},\ \widehat{B}] - \widehat{C}\|_F^2 \leqslant \frac{\delta^2}{\mathsf{poly}(m)} \cdot \|\widehat{B}\|_F^2 \leqslant \delta^2. \tag{6.7}$$

We know that $\overline{X}$ is the optimal solution to the program $\min_{X \in \mathbb{R}^{n \times d}} \|S_2 \widehat{A} X - S_2 \widehat{B}\|_F^2$. Hence by Theorem 6.3.6, with probability 0.99,

$$\|\widehat{A}\overline{X} - \widehat{B}\|_F^2 \leqslant (1+\varepsilon) \min_{X \in \mathbb{R}^{n \times d}} \|S_2 \widehat{A} X - S_2 \widehat{B}\|_F^2 = 0.$$

which implies $\widehat{A}\overline{X} = \widehat{B}$. Hence we have

$$\begin{aligned} \|[\widehat{A},\ \widehat{A}\overline{X}] - C\|_F &\leqslant \|[\widehat{A},\ \widehat{A}\overline{X}] - \widehat{C}\|_F + \|\widehat{C} - C\|_F \\ &= \|[\widehat{A},\ \widehat{B}] - \widehat{C}\|_F + \|\widehat{C} - C\|_F \\ &\leqslant \delta + \|\widehat{C} - C\|_F \end{aligned}$$

where the first step follows by triangle inequality, and the last step follows by (6.7). □

Now that we have $\overline{A}$ and $\overline{B}$, and we can compute $X$ by solving the regression problem $\min_{X \in \mathbb{R}^{n \times d}} \|\overline{A} X - \overline{B}\|_F^2$.

We next summarize the running time.

**Lemma 6.5.9** (Running time analysis)**.** *Procedure* FASTTOTALLEASTSQUARES *in Algorithm 2 runs in time* $\widetilde{O}(\mathsf{nnz}(A) + \mathsf{nnz}(B) + d \cdot \mathsf{poly}(n/\varepsilon))$.

*Proof.* We bound the time of each step:

1. Construct the $s_1 \times m$ CountSketch matrix $S_1$ and compute $S_1 C$ with $s_1 = O(n/\varepsilon)$. This step takes time $\mathsf{nnz}(C) + d \cdot \mathsf{poly}(n/\varepsilon)$.

2. Construct the $(n+d) \times d_1$ leverage sampling and rescaling matrix $D_1$ with $d_1 = \widetilde{O}(n/\varepsilon)$ nonzero diagonal entries and compute $CD_1$. This step takes time $\widetilde{O}(\mathsf{nnz}(C) + d \cdot \mathsf{poly}(n/\varepsilon))$.

3. Construct the $d_2 \times m$ leverage sampling and rescaling matrix $D_2$ with $d_2 = \widetilde{O}(n/\varepsilon)$ nonzero diagonal entries. This step takes time $\widetilde{O}(\mathsf{nnz}(C) + d \cdot \mathsf{poly}(n/\varepsilon))$ according to Theorem 6.3.5.

4. Compute $Z_2 \in \mathbb{R}^{d_1 \times s_1}$ by solving the rank-constrained system:

$$\min_{\text{rank}-n\ Z \in \mathbb{R}^{d_1 \times s_1}} \|D_2 C D_1 Z S_1 C - D_2 C\|_F^2.$$

Note that $D_2CD_1$ has size $\widetilde{O}(n/\varepsilon) \times \widetilde{O}(n/\varepsilon)$, $S_1C$ has size $O(n/\varepsilon) \times (n+d)$, and $D_2C$ has size $\widetilde{O}(n/\varepsilon) \times (n+d)$, so according to Theorem 6.3.8, we have an explicit closed form for $Z_2$, and the time taken is $d \cdot \mathsf{poly}(n/\varepsilon)$.

5. Run procedure SPLIT to get $\overline{A} \in \mathbb{R}^{s_2 \times n}$ and $\overline{B} \in \mathbb{R}^{s_2 \times d}$ with $s_2 = O(n/\varepsilon)$. By Lemma 6.5.8, this step takes time $O(\mathsf{nnz}(C) + d \cdot \mathsf{poly}(n/\varepsilon))$.

6. Compute $X$ by solving the regression problem $\min_{X \in \mathbb{R}^{n \times d}} \|\overline{A}X - \overline{B}\|_F^2$ in time $O(d \cdot \mathsf{poly}(n/\varepsilon))$. This is because $X = (\overline{A})^\dagger \overline{B}$, and $\overline{A}$ has size $O(n/\varepsilon) \times n$, so we can compute $(\overline{A})^\dagger$ in time $O((n/\varepsilon)^\omega) = \mathsf{poly}(n/\varepsilon)$, and then compute $X$ in time $O((n/\varepsilon)^2 \cdot d)$ since $\overline{B}$ is an $O(n/\varepsilon) \times d$ matrix.

Notice that $\mathsf{nnz}(C) = \mathsf{nnz}(A) + \mathsf{nnz}(B)$, so we have the desired running time. □

To summarize, Theorem 6.5.10 shows the performance of our algorithm.

**Theorem 6.5.10** (Main Result). *Given two matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{m \times d}$, letting*

$$\text{OPT} = \min_{\text{rank}-n\ C' \in \mathbb{R}^{m \times (n+d)}} \|C' - [A,\ B]\|_F,$$

*we have that for any $\varepsilon \in (0,1)$, there is an algorithm (procedure* FASTTOTALLEASTSQUARES *in Algorithm 2) that runs in $\widetilde{O}(\mathsf{nnz}(A) + \mathsf{nnz}(B)) + d \cdot \mathsf{poly}(n/\varepsilon))$ time and outputs a matrix $X \in \mathbb{R}^{n \times d}$ such that there is a matrix $\widehat{A} \in \mathbb{R}^{m \times n}$ satisfying that*

$$\|[\widehat{A},\ \widehat{A}X] - [A,\ B]\|_F \leqslant (1+\varepsilon)\,\text{OPT} + \delta$$

*holds with probability at least $9/10$, where $\delta > 0$ is arbitrarily small.*

*Proof.* The running time follows from Lemma 6.5.9. For the approximation ratio, let $\widehat{A}$, $\overline{A}$ be defined as in Lemma 6.5.8. From Lemma 6.5.8, there exists $\overline{X} \in \mathbb{R}^{n\times d}$ satisfying $\overline{A}\overline{X} = \overline{B}$. Since $X$ is obtained from solving the regression problem $\|\overline{A}X - \overline{B}\|_F^2$, we also have $\overline{A}X = \overline{B}$. Hence with probability 0.9,

$$\|[\widehat{A}, \widehat{A}X] - C\|_F \leqslant \delta + \|\widehat{C} - C\|_F \leqslant \delta + (1 + O(\varepsilon))\,\mathrm{OPT},$$

where the first step uses Lemma 6.5.8 and the second step uses Claim 6.5.6. Rescaling $\varepsilon$ gives the desired statement. □

**Remark 6.5.11.** *The success probability* $9/10$ *in Theorem 6.5.10 can be boosted to* $1-\delta$ *for any* $\delta > 0$ *in a standard way. Namely, we run our FTLS algorithm* $O(\log(1/\delta))$ *times where in each run we use independent randomness, and choose the solution found with the smallest cost. Note that for any fixed output* $X$*, the cost* $\|[\bar{A}, \bar{A}X] - [A, B]\|_F$ *can be efficiently approximated. To see this, let* $S$ *be a CountSketch matrix with* $O(\varepsilon^{-2})$ *rows. Then* $\|S[\bar{A}, \bar{A}X] - S[A, B]\|_F = (1 \pm \varepsilon)\|[\bar{A}, \bar{A}X] - [A, B]\|_F$ *with probability* $9/10$ *(see, for example Lemma 40 of [30] ). We can compute* $\|S[\bar{A}, \bar{A}X] - S[A, B]\|_F$ *in time* $O(d \cdot poly(n/\varepsilon))$*, and applying* $S$ *can be done in* $nnz(A) + nnz(B)$ *time. We can then amplify the success probability by taking* $O(\log(1/\delta))$ *independent estimates and taking the median of the estimates. This is a* $(1 \pm \varepsilon)$*-approximation with probability at least* $1 - O(\delta/\log(1/\delta))$*. We run our FTLS algorithm* $O(\log(1/\delta))$ *times, obtaining outputs* $X^1, \ldots, X^{O(\log(1/\delta))}$ *and for each* $X^i$*, apply the method above to estimate its cost. Since for each* $X^i$ *our estimate to the cost is within* $1 \pm \varepsilon$ *with probability at least* $1 - O(\delta/(\log(1/\delta))$*, by a union bound the estimates for all* $X^i$ *are within* $1 \pm \varepsilon$ *with probability at least* $1 - \delta/2$*. Since also the solution with minimal cost is a* $1 \pm \varepsilon$ *approximation with probability at least* $1 - \delta/2$*, by a union bound we can achieve* $1 - \delta$ *probability with running time* $\widetilde{O}(\log^2(1/\delta)) \cdot (nnz(A) + nnz(B) + d \cdot poly(n/\varepsilon)))$*.*

### *6.6 Extension to regularized total least squares problem*

We further generalize our algorithm to handle regularization. In this section we provide Algorithm 3, our algorithm for the regularized total least squares problem and prove its

correctness. Recall that our regularized total least squares problem is defined as follows.

$$\mathrm{OPT} := \min_{\widehat{A}\in\mathbb{R}^{m\times n}, X\in\mathbb{R}^{n\times d}, U\in\mathbb{R}^{m\times n}, V\in\mathbb{R}^{n\times(n+d)}} \|UV - [A,\ B]\|_F^2 + \lambda\|U\|_F^2 + \lambda\|V\|_F^2 \qquad (6.8)$$
$$\text{subject to } [\widehat{A},\ \widehat{A}X] = UV$$

**Definition 6.6.1** (Statistical Dimension, e.g., see [7]). *For $\lambda > 0$ and rank $k$ matrix $A$, the* statistical dimension *of the ridge regression problem with regularizing weight $\lambda$ is defined as*

$$\mathrm{sd}_\lambda(A) := \sum_{i\in[k]} \frac{1}{1+\lambda/\sigma_i^2}$$

*where $\sigma_i$ is the $i$-th singular value of $A$ for $i \in [k]$.*

Notice that $\mathrm{sd}_\lambda(A)$ is decreasing in $\lambda$, so we always have $\mathrm{sd}_\lambda(A) \leqslant \mathrm{sd}_0(A) = \mathrm{rank}(A)$.

**Lemma 6.6.2** (Exact solution of low rank approximation with regularization, Lemma 27 of [7]). *Given positive integers $n_1, n_2, r, s, k$ and parameter $\lambda \geqslant 0$. For $C \in \mathbb{R}^{n_1\times r}$, $D \in \mathbb{R}^{s\times n_2}$, $B \in \mathbb{R}^{n_1\times n_2}$, the problem of finding*

$$\min_{Z_R\in\mathbb{R}^{r\times k}, Z_S\in\mathbb{R}^{k\times s}} \|CZ_RZ_SD - B\|_F^2 + \lambda\|CZ_R\|_F^2 + \lambda\|Z_SD\|_F^2,$$

*and the minimizing of $CZ_R \in \mathbb{R}^{n_1\times k}$ and $Z_SD \in \mathbb{R}^{k\times n_2}$, can be solved in*

$$O(n_1 r\cdot\mathrm{rank}(C) + n_2 s\cdot\mathrm{rank}(D) + \mathrm{rank}(D)\cdot n_1(n_2 + r_C))$$

*time.*

**Theorem 6.6.3** (Sketching for solving ridge regression, Theorem 19 in [7]). *Fix $m \geqslant n$. For $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times d}$ and $\lambda > 0$, consider the rigid regression problem*

$$\min_{X\in\mathbb{R}^{n\times d}} \|AX - B\|_F^2 + \lambda\|X\|_F^2.$$

*Let $S \in \mathbb{R}^{s\times m}$ be a CountSketch matrix with $s = \widetilde{O}(\mathrm{sd}_\lambda(A)/\varepsilon) = \widetilde{O}(n/\varepsilon)$, then with probability 0.99,*

$$\min_{X\in\mathbb{R}^{n\times d}} \|SAX - SB\|_F^2 + \lambda\|X\|_F^2 \leqslant (1+\varepsilon)\min_{X\in\mathbb{R}^{n\times d}} \|AX - B\|_F^2 + \lambda\|X\|_F^2$$

**Algorithm 3:** Our Fast Total Least Squares Algorithm with Regularization

**Function** FASTREGULARIZEDTOTALLEASTSQUARES($A, B, n, d, \lambda\varepsilon, \delta$)**:**

/* Theorem 6.6.7 */

$s_1 \leftarrow \widetilde{O}(n/\varepsilon), s_2 \leftarrow \widetilde{O}(n/\varepsilon), , s_3 \leftarrow \widetilde{O}(n/\varepsilon), d_1 \leftarrow \widetilde{O}(n/\varepsilon)$;

Choose $S_1 \in \mathbb{R}^{s_1 \times m}$ to be a CountSketch matrix, then compute $S_1 C$;

Choose $S_2 \in \mathbb{R}^{s_2 \times (n+d)}$ to be a CountSketch matrix, then compute $CS_2^\top$;

Choose $D_1 \in \mathbb{R}^{d_1 \times m}$ to be a leverage score sampling and rescaling matrix according to the rows of $CS_2^\top$;

$\widehat{Z}_1, \widehat{Z}_2 \leftarrow \arg\min_{Z_1 \in \mathbb{R}^{n \times s_1}, Z_2 \in \mathbb{R}^{s_2 \times n}} \|D_1 CS_2^\top Z_2 Z_1 S_1 C - D_1 C\|_F^2 + \lambda \|D_1 CS_2^\top Z_2\|_F^2 + \lambda \|Z_1 S_1 C\|_F^2$ ▷ Theorem 6.6.2;

$\overline{A}, \overline{B}, \pi \leftarrow \text{SPLIT}(CS_2^\top, \widehat{Z}_1, \widehat{Z}_2, S_1 C, n, d, \delta/\mathsf{poly}(m))$, $X \leftarrow \min \|\overline{A}X - \overline{B}\|_F$;

**return** $X$;

**Function** SPLIT($CS_2^\top, \widehat{Z}_1, \widehat{Z}_2, S_1 C, n, d, \delta$)**:**

/* Lemma 6.5.8 */

Choose $S_3 \in \mathbb{R}^{s_3 \times m}$ to be a CountSketch matrix;

$\overline{C} \leftarrow (S_3 \cdot CS_2^\top) \cdot \widehat{Z}_2 \cdot \widehat{Z}_1 \cdot S_1 C$ ▷ $\widehat{C} = CS_2^\top \widehat{Z}_2 \widehat{Z}_1 S_1 C$ ; $\overline{C} = S_3 \widehat{C}$;

$\overline{A} \leftarrow \overline{C}_{*,[n]}, \overline{B} \leftarrow \overline{C}_{*,[n+d]\setminus[n]}$ ▷ $\widehat{A} = \widehat{C}_{*,[n]}, \widehat{B} = \widehat{C}_{*,[n+d]\setminus[n]}$; $\overline{A} = S_3 \widehat{A}, \overline{B} = S_3 \widehat{B}$;

$T \leftarrow \emptyset$, $\pi(i) = -1$ for all $i \in [n]$;

**for** $i = 1 \to n$ **do**

  **if** $\overline{A}_{*,i}$ *is linearly dependent of* $\overline{A}_{*,[n]\setminus\{i\}}$ **then**

    $j \leftarrow \min_{j \in [d]\setminus T} \{\overline{B}_{*,j}$ is linearly independent of $\overline{A}\}$, $\overline{A}_{*,i} \leftarrow \overline{A}_{*,i} + \delta \cdot \overline{B}_{*,j}$, $T \leftarrow T \cup \{j\}$, $\pi(i) \leftarrow j$

**return** $\overline{A}, \overline{B}, \pi$ ▷ $\pi : [n] \to \{-1\} \cup ([n+d]\setminus[n])$

*Moreover, SA, SB can be computed in time*

$$O(\mathsf{nnz}(A) + \mathsf{nnz}(B)) + \widetilde{O}\left((n+d)(\mathrm{sd}_\lambda(A)/\varepsilon + \mathrm{sd}_\lambda(A)^2)\right).$$

We claim that it is sufficient to look at solutions of the form $CS_2^\top Z_2 Z_1 S_1 C$.

**Claim 6.6.4** (CountSketch matrix for low rank approximation problem)**.** *Given matrix* $C \in \mathbb{R}^{m\times(n+d)}$*. Let* OPT *be defined as in* (6.8)*. For any* $\varepsilon > 0$*, let* $S_1 \in \mathbb{R}^{s_1\times m}$*,* $S_2 \in \mathbb{R}^{s_2\times m}$ *be the sketching matrices defined in Algorithm 3, then with probability* 0.98*,*

$$\min_{Z_1\in\mathbb{R}^{n\times s_1}, Z_2\in\mathbb{R}^{s_2\times n}} \|CS_2^\top Z_2 Z_1 S_1 C - C\|_F^2 + \lambda\|CS_2^\top Z_2\|_F^2 + \lambda\|Z_1 S_1 C\|_F^2 \leqslant (1+\varepsilon)^2\,\mathrm{OPT}\,.$$

*Proof.* Let $U^* \in \mathbb{R}^{m\times n}$ and $V^* \in \mathbb{R}^{n\times(n+d)}$ be the optimal solution to the program (6.8). Consider the following optimization problem:

$$\min_{V\in\mathbb{R}^{n\times(n+d)}} \|U^*V - C\|_F^2 + \lambda\|V\|_F^2 \tag{6.9}$$

Clearly $V^* \in \mathbb{R}^{n\times(n+d)}$ is the optimal solution to program (6.9), since for any solution $V \in \mathbb{R}^{n\times(n+d)}$ to program (6.9) with cost $c$, $(U^*, V)$ is a solution to program (6.8) with cost $c + \lambda\|U^*\|_F^2$.

Program (6.9) is a ridge regression problem. Hence we can take a CountSketch matrix $S \in \mathbb{R}^{s_1\times m}$ with $s_1 = \widetilde{O}(n/\varepsilon)$ to obtain

$$\min_{V\in\mathbb{R}^{n\times(n+d)}} \|S_1U^*V - S_1C\|_F^2 + \lambda\|V\|_F^2 \tag{6.10}$$

Let $V_1 \in \mathbb{R}^{n\times(n+d)}$ be the minimizer of the above program, then we know

$$V_1 = \begin{bmatrix} S_1U^* \\ \sqrt{\lambda}I_n \end{bmatrix}^\dagger \begin{bmatrix} S_1C \\ 0 \end{bmatrix},$$

which means $V_1 \in \mathbb{R}^{n\times(n+d)}$ lies in the row span of $S_1C \in \mathbb{R}^{s_1\times(n+d)}$. Moreover, by Theorem 6.6.3, with probability at least 0.99 we have

$$\|U^*V_1 - C\|_F^2 + \lambda\|V_1\|_F^2 \leqslant (1+\varepsilon)\|U^*V^* - C\|_F^2 + \lambda\|V^*\|_F^2 \tag{6.11}$$

Now consider the problem

$$\min_{U\in\mathbb{R}^{m\times n}} \|UV_1 - C\|_F^2 + \lambda\|U\|_F^2 \tag{6.12}$$

Let $U_0 \in \mathbb{R}^{m\times n}$ be the minimizer of program (6.12). Similarly, we can take a CountSketch matrix $S_2 \in \mathbb{R}^{s_2\times(n+d)}$ with $s_2 = \widetilde{O}(n/\varepsilon)$ to obtain

$$\min_{U\in\mathbb{R}^{m\times n}} \|UV_1S_2^\top - CS_2^\top\|_F^2 + \lambda\|U\|_F^2 \tag{6.13}$$

Let $U_1 \in \mathbb{R}^{m\times n}$ be the minimizer of program (6.13), then we know

$$U_1^\top = \begin{bmatrix} S_2V_1^\top \\ \sqrt{\lambda}I_n \end{bmatrix}^\dagger \begin{bmatrix} S_2C^\top \\ 0 \end{bmatrix},$$

which means $U_1 \in \mathbb{R}^{m\times n}$ lies in the column span of $CS_2^\top \in \mathbb{R}^{m\times s_2}$. Moreover, with probability at least 0.99 we have

$$\begin{aligned} \|U_1V_1 - C\|_F^2 + \lambda\|U_1\|_F^2 &\leqslant (1+\lambda)\cdot(\|U_0V_1 - C\|_F^2 + \lambda\|U_0\|_F^2) \\ &\leqslant (1+\lambda)\cdot(\|U^*V_1 - C\|_F^2 + \lambda\|U^*\|_F^2) \end{aligned} \tag{6.14}$$

where the first step we use Theorem 6.6.3 and the second step follows that $U_0$ is the minimizer.

Now let us compute the cost.

$$\begin{aligned} &\|U_1V_1 - C\|_F^2 + \lambda\|U_1\|_F^2 + \lambda\|V_1\|_F^2 \\ =\ &\lambda\|V_1\|_F^2 + (\|U_1V_1 - C\|_F^2 + \lambda\|U_1\|_F^2) \\ \leqslant\ &\lambda\|V_1\|_F^2 + (1+\varepsilon)(\|U^*V_1 - C\|_F^2 + \lambda\|U^*\|_F^2) \\ \leqslant\ &(1+\varepsilon)\cdot\Big(\lambda\|U^*\|_F^2 + (\|U^*V_1 - C\|_F^2 + \lambda\|V_1\|_F^2)\Big) \\ \leqslant\ &(1+\varepsilon)\cdot\Big(\lambda\|U^*\|_F^2 + (1+\varepsilon)^2\cdot(\|U^*V^* - C\|_F^2 + \lambda\|V^*\|_F^2)\Big) \\ \leqslant\ &(1+\varepsilon)^2\cdot(\|U^*V^* - C\|_F^2 + \lambda\|U^*\|_F^2 + \lambda\|V^*\|_F^2) \\ =\ &(1+\varepsilon)^2\,\mathrm{OPT} \end{aligned}$$

where the second step follows from (6.14), the fourth step follows from (6.11), and the last step follows from the definition of $U^* \in \mathbb{R}^{m\times n}, V^* \in \mathbb{R}^{n\times(n+d)}$.

Finally, since $V_1 \in \mathbb{R}^{n\times(n+d)}$ lies in the row span of $S_1C \in \mathbb{R}^{s_1\times(n+d)}$ and $U_1 \in \mathbb{R}^{m\times n}$ lies in the column span of $CS_2^\top \in \mathbb{R}^{m\times s_2}$, there exists $Z_1^* \in \mathbb{R}^{n\times s_1}$ and $Z_2^* \in \mathbb{R}^{s_2\times n}$ so that

$V_1 = Z_1^* S_1 C \in \mathbb{R}^{n\times(n+d)}$ and $U_1 = CS_2^\top Z_2^* \in \mathbb{R}^{m\times n}$. Then the claim stated just follows from $(Z_1^*, Z_2^*)$ are also feasible. □

Now we just need to solve the optimization problem

$$\min_{Z_1\in\mathbb{R}^{n\times s_1}, Z_2\in\mathbb{R}^{s_2\times n}} \|CS_2^\top Z_2 Z_1 S_1 C - C\|_F^2 + \lambda\|CS_2^\top Z_2\|_F^2 + \lambda\|Z_1 S_1 C\|_F^2 \tag{6.15}$$

The size of this program is quite huge, i. e., we need to work with an $m \times d_2$ matrix $CS_2^\top$. To handle this problem, we again apply sketching techniques. Let $D_1 \in \mathbb{R}^{d_1\times m}$ be a leverage score sampling and rescaling matrix according to the matrix $CS_2 \in \mathbb{R}^{m\times s_2}$, so that $D_1$ has $d_1 = \widetilde{O}(n/\varepsilon)$ nonzeros on the diagonal. Now, we arrive at the small program that we are going to directly solve:

$$\min_{Z_1\in\mathbb{R}^{n\times s_1}, Z_2\in\mathbb{R}^{s_2\times n}} \|D_1 CS_2^\top Z_2 Z_1 S_1 C - D_1 C\|_F^2 + \lambda\|D_1 CS_2^\top Z_2\|_F^2 + \lambda\|Z_1 S_1 C\|_F^2 \tag{6.16}$$

We have the following approximation guarantee.

**Claim 6.6.5.** *Let $(Z_1^*, Z_2^*)$ be the optimal solution to program* (6.15)*. Let $(\widehat{Z}_1, \widehat{Z}_2)$ be the optimal solution to program* (6.16)*. With probability* 0.96*,*

$$\begin{aligned}&\|CS_2^\top \widehat{Z}_2\widehat{Z}_1 S_1 C - C\|_F^2 + \lambda\|CS_2^\top \widehat{Z}_2\|_F^2 + \lambda\|\widehat{Z}_1 S_1 C\|_F^2\\ \leqslant&(1+\varepsilon)^2(\|CS_2^\top Z_2^* Z_1^* S_1 C - C\|_F^2 + \lambda\|CS_2^\top Z_2^*\|_F^2 + \lambda\|Z_1^* S_1 C\|_F^2)\end{aligned}$$

*Proof.* This is because

$$\begin{aligned}&\|CS_2^\top \widehat{Z}_2\widehat{Z}_1 S_1 C - C\|_F^2 + \lambda\|CS_2^\top \widehat{Z}_2\|_F^2 + \lambda\|\widehat{Z}_1 S_1 C\|_F^2\\ \leqslant&(1+\varepsilon)\left(\|D_1 CS_2^\top \widehat{Z}_2\widehat{Z}_1 S_1 C - D_1 C\|_F^2 + \lambda\|D_1 CS_2^\top \widehat{Z}_2\|_F^2\right) + \lambda\|\widehat{Z}_1 S_1 C\|_F^2\\ \leqslant&(1+\varepsilon)\left(\|D_1 CS_2^\top \widehat{Z}_2\widehat{Z}_1 S_1 C - D_1 C\|_F^2 + \lambda\|D_1 CS_2^\top \widehat{Z}_2\|_F^2 + \lambda\|\widehat{Z}_1 S_1 C\|_F^2\right)\\ \leqslant&(1+\varepsilon)\left(\|D_1 CS_2^\top Z_2^* Z_1^* S_1 C - D_1 C\|_F^2 + \lambda\|D_1 CS_2^\top Z_2^*\|_F^2 + \lambda\|Z_1^* S_1 C\|_F^2\right)\\ \leqslant&(1+\varepsilon)^2\left(\|CS_2^\top Z_2^* Z_1^* S_1 C - C\|_F^2 + \lambda\|CS_2^\top Z_2^*\|_F^2 + \lambda\|Z_1^* S_1 C\|_F^2\right)\end{aligned}$$

where the first step uses property of the leverage score sampling matrix $D_1$, the third step follows from $(\widehat{Z}_1, \widehat{Z}_2)$ are minimizers of program (6.16), and the fourth step again uses property of the leverage score sampling matrix $D_1$. □

Let $\widehat{U} = CS_2^\top \widehat{Z}_2, \widehat{V} = \widehat{Z}_1 S_1 C$ and $\widehat{C} = \widehat{U}\widehat{V}$. Combining Claim 6.6.4 and Claim 6.6.5 together, we get with probability at least 0.91,

$$\|\widehat{U}\widehat{V} - [A,\ B]\|_F^2 + \lambda\|\widehat{U}\|_F^2 + \lambda\|\widehat{V}\|_F^2 \leqslant (1+\varepsilon)^4\,\mathrm{OPT} \tag{6.17}$$

If the first $n$ columns of $\widehat{C}$ can span the whole matrix $\widehat{C}$, then we are in good shape. In this case we have:

**Claim 6.6.6** (Perfect first $n$ columns)**.** *Let $S_3 \in \mathbb{R}^{s_3 \times m}$ be the CountSketch matrix defined in Algorithm 3. Write $\widehat{C}$ as $[\widehat{A},\ \widehat{B}]$ where $\widehat{A} \in \mathbb{R}^{m\times n}$ and $\widehat{B} \in \mathbb{R}^{m\times d}$. If there exists $\widehat{X} \in \mathbb{R}^{n\times d}$ so that $\widehat{B} = \widehat{A}\widehat{X}$, let $\bar{X} \in \mathbb{R}^{n\times d}$ be the minimizer of $\min_{X\in\mathbb{R}^{n\times d}} \|S_3\widehat{A}X - S_3\widehat{B}\|_F^2$, then with probability 0.9,*

$$\|[\widehat{A}, \widehat{A}\bar{X}] - [A,\ B]\|_F^2 + \lambda\|\widehat{U}\|_F^2 + \lambda\|\widehat{V}\|_F^2 \leqslant (1+\varepsilon)^4\,\mathrm{OPT}$$

*Proof.* We have with probability 0.99,

$$\|\widehat{A}\bar{X} - \widehat{B}\|_F^2 \leqslant (1+\varepsilon)\|\widehat{A}\widehat{X} - \widehat{B}\|_F^2 = 0$$

where the first step follows from Theorem 6.3.6 and the second step follows from the assumption. Recall that $\widehat{C} = \widehat{U}\widehat{V}$, so

$$\begin{aligned}&\|[\widehat{A}, \widehat{A}\bar{X}] - [A,\ B]\|_F^2 + \lambda\|\widehat{U}\|_F^2 + \lambda\|\widehat{V}\|_F^2\\ =&\|\widehat{U}\widehat{V} - [A,\ B]\|_F^2 + \lambda\|\widehat{U}\|_F^2 + \lambda\|\widehat{V}\|_F^2 \leqslant (1+\varepsilon)^4\,\mathrm{OPT}\end{aligned}$$

where the last step uses (6.17). □

However, if $\widehat{C}$ does not have nice structure, then we need to apply our procedure SPLIT, which would introduce the additive error $\delta$. Overall, by rescaling $\varepsilon$, our main result is summarized as follows.

**Theorem 6.6.7** (Algorithm for the regularized total least squares problem)**.** *Given two matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{m\times d}$ and $\lambda > 0$, letting*

$$\mathrm{OPT} = \min_{U\in\mathbb{R}^{m\times n}, V\in\mathbb{R}^{n\times(n+d)}} \|UV - [A,\ B]\|_F^2 + \lambda\|U\|_F^2 + \lambda\|V\|_F^2,$$

*we have that for any* $\varepsilon \in (0,1)$*, there is an algorithm that runs in*

$$\widetilde{O}(\mathsf{nnz}(A) + \mathsf{nnz}(B) + d \cdot \mathsf{poly}(n/\varepsilon))$$

*time and outputs a matrix* $X \in \mathbb{R}^{n\times d}$ *such that there is a matrix* $\widehat{A} \in \mathbb{R}^{m\times n}$, $\widehat{U} \in \mathbb{R}^{m\times n}$ *and* $\widehat{V} \in \mathbb{R}^{n\times(n+d)}$ *satisfying that* $\|[\widehat{A},\ \widehat{A}X] - \widehat{U}\widehat{V}\|_F^2 \leqslant \delta$ *and*

$$\|[\widehat{A},\ \widehat{A}X] - [A,\ B]\|_F + \lambda\|\widehat{U}\|_F^2 + \lambda\|\widehat{V}\|_F^2 \leqslant (1+\varepsilon)\,\mathrm{OPT} + \delta$$

### *6.7 Experiments*

We conducted several experiments to verify the running time and optimality of our fast total least squares algorithm 2. Let us first recall the multiple-response regression problem. Let $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{m\times d}$. In this problem, we want to find $X \in \mathbb{R}^{n\times d}$ so that $AX \sim B$. The least squares method (LS) solves the following optimization program:

$$\begin{aligned} c_{\mathrm{LS}} := \min_{X\in\mathbb{R}^{n\times d}, \Delta B\in\mathbb{R}^{m\times d}} & \|\Delta B\|_F^2, \\ \text{subject to } & AX = B + \Delta B. \end{aligned}$$

On the other hand, the total least squares method (TLS) solves the following optimization program:

$$c_{\mathrm{TLS}} := \min_{\mathrm{rank}-n\ C'\in\mathbb{R}^{m\times(n+d)}} \|C' - [A\ B]\|_F.$$

The fast total least squares method (FTLS) returns $X \in \mathbb{R}^{n\times d}$, which provides an approximation $C' = [\widehat{A}\ \widehat{A}X]$ to the TLS solution, and the cost is computed as $c_{\mathrm{FTLS}} = \|C' - C\|_F^2$.

Our numerical tests are carried out on an Intel Xeon E7-8850 v2 server with 2.30GHz and 4GB RAM under Matlab R2017b. The code can be found at `https://github.com/yangxinuw/total_least_squares_code`. We also put the main Matlab code in Appendix A.

### 6.7.1 A Toy Example

We first run our FTLS algorithm on the following toy example, for which we have the analytical solution exactly. Let $A \in \mathbb{R}^{3\times 2}$ be $A_{11} = A_{22} = 1$ and 0 everywhere else. Let $B \in \mathbb{R}^{3\times 1}$ be $B_3 = 3$ and 0 everywhere else. The cost of LS is 9, since $AX$ can only have non-zero entries on the first 2 coordinates, so the 3rd coordinate of $AX - B$ must have absolute value 3. Hence the cost is at least 9. Moreover, a cost 9 can be achieved by setting $X = 0$ and $\Delta B = -B$. However, for the TLS algorithm, the cost is only 1. Consider $\Delta A \in \mathbb{R}^{3\times 2}$ where $A_{11} = -1$ and 0 everywhere else. Then $C' := [(A + \Delta A),\ B]$ has rank 2, and $\|C' - C\|_F = 1$.

We first run experiments on this small matrix. Since we know the solution of LS and TLS exactly in this case, it is convenient for us to compare their results with that of the FTLS algorithm. When we run the FTLS algorithm, we sample 2 rows in each of the sketching algorithms.

The experimental solution of LS is $C_{\text{LS}} = \text{diag}(0, 1, 3)$ which matches the theoretical solution. The cost is 9. The experimental solution of TLS is $C_{\text{TLS}} = \text{diag}(1, 1, 0)$ which also matches the theoretical result. The cost is 1.

FTLS is a randomized algorithm, so the output varies. We post several outputs:

$$C_{\text{FTLS}} = \begin{bmatrix} .06 & -.01 & .25 \\ -.01 & .99 & .00 \\ .76 & .01 & 2.79 \end{bmatrix}, \begin{bmatrix} .14 & -.26 & -.22 \\ -.26 & .91 & -.06 \\ -.67 & -.20 & 2.82 \end{bmatrix}$$

These solutions have cost of 1.55 and 1.47.

We run the FTLS multiple times to analyze the distribution of costs. Experimental result, which can be found in Figure 6.1, shows that FTLS is a stable algorithm, and consistently performs better than LS.

We also consider the generalization of this example with larger dimension. Let $A \in \mathbb{R}^{m\times n}$ be $A_{ii} = 1$ for $i = 1, \cdots, n$ and 0 everywhere else. Let $B \in \mathbb{R}^{m\times 1}$ be $B_{n+1} = 3$ and 0 everywhere else.

The cost of LS is 9, since $AX$ can only have non-zero entries on the first $n$ coordinates, so the $(n+1)$-th coordinate of $AX - B$ must have absolute value 3. Hence the cost is at least 9. Moreover, a cost 9 can be achieved by setting $X = 0$ and $\Delta B = -B$.

However, for the TLS algorithm, the cost is only 1. Consider $\Delta A \in \mathbb{R}^{m \times n}$ where $A_{11} = -1$ and 0 everywhere else. Then $C' := [(A + \Delta A),\ B]$ does have rank $n$, and $\|C' - C\|_F = 1$.

For a concrete example, we set $m = 10$, $n = 5$. That is,

$$C := [A,\ B] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

When we run the FTLS algorithm, we sample 6 rows in all the sketching algorithms.

The experimental solution of LS is $C_{\text{LS}}$ which is the same as the theoretical solution. The cost is 9. The experimental solution of TLS is $C_{\text{TLS}}$ which is also the same as the theoretical result. The cost is 1.

$$C_{\text{LS}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad C_{\text{TLS}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

FTLS is a randomized algorithm, so the output varies. We post several outputs:

$$C_{\text{FTLS}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.1 & -0.3 \\ 0 & 0 & 0 & 0 & -0.9 & 2.7 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

This solution has a cost of 4.3.

$$\widehat{C}_{\text{FTLS}} = \begin{bmatrix} 0.5 & -0.5 & 0 & 0 & 0 & 0 \\ -0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.09 & 0.09 & 0 & 0.27 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.82 & 0.82 & 0 & 2.45 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

This solution has a cost of 5.5455.

$$C_{\text{FTLS}} = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -0.9 & 0 & 2.7 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

This solution has a cost of 3.4.

We also consider a second small toy example. Let $A$ still be a $10 \times 5$ matrix and $B$ be a $10 \times 1$ vector. Each entry $A(i,j)$ is chosen i.i.d. from the normal distribution $N(0,1)$, and each entry $B(i)$ is chosen from $N(0,3)$. Because entries from $A$ and $B$ have different variance, we expect the results of LS and TLS to be quite different. When we run the FTLS algorithm, we sample 6 rows.

We run FTLS 1000 times, and compute the distribution of costs. Figure 6.1, the results

of this experiment, again demonstrates the stability of the algorithm. In both figures, the $x$-axis is the cost of the FTLS algorithm, measured by $\|C' - C\|_F^2$ where $C'$ is the output of our FTLS algorithm; the $y$-axix is the frequency of each cost that is grouped in suitable range.

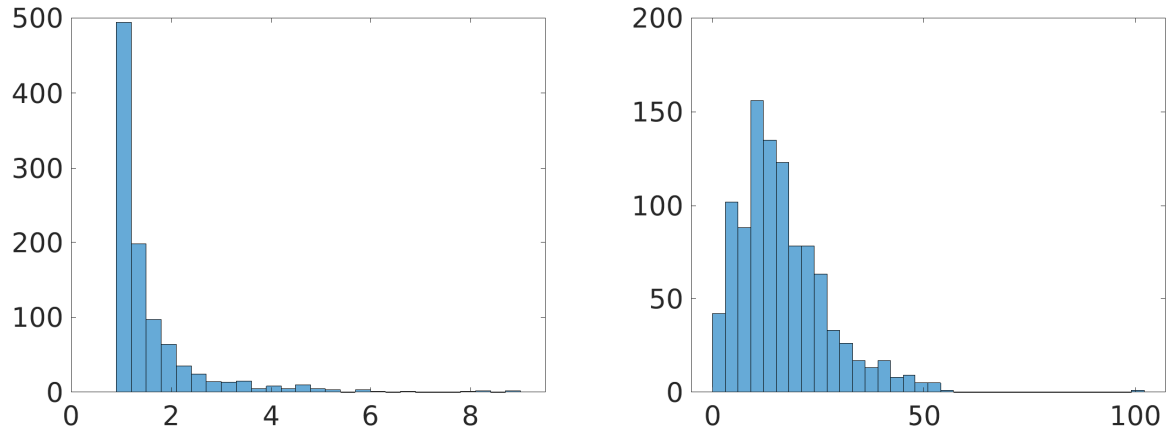


Figure 6.1: Cost distribution of our fast least squares algorithm on toy examples. The $x$-axis is the cost for FTLS. (Note that we want to minimize the cost); the $y$-axis is the frequency of each cost. (Left) First toy example, TLS cost is 1, LS cost is 9. (Right) Second toy example, TLS cost is 1.30, LS cost is 40.4

### *6.7.2 Large Scale Problems*

We have already seen that FTLS works pretty well on small matrices. We next show that the fast total least squares method also provides a good estimate for large scale regression problems. The setting for matrices is as follows: for $k = 5, 10, \cdots, 100$, we set $A$ to be a $20k \times 2k$ matrix where $A(i, i) = 1$ for $i = 1, \cdots, 2k$ and 0 everywhere else, and we set $B$ to be a $20k \times 1$ vector where $B(2k + 1) = 3$ and 0 elsewhere. As in the small case, the cost of TLS is 1, and the cost of LS is 9.

Recall that in the FTLS algorithm, we use CountSketch/leverage scores sampling/-Gaussian sketches to speed up the algorithm. In the experiments, we take sample density

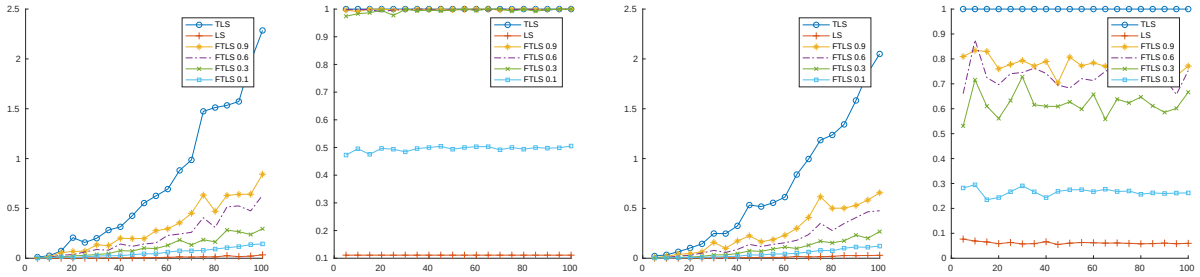


Figure 6.2: Running time and accuracy of our FTLS algorithms. The left 2 figures are for the sparse matrix. The right 2 pictures are for the Gaussian matrix. (Left) The $y$-axis is the running time of each algorithm (counted in seconds); the $x$-axis is the size of the matrix. (Right) The $y$-axis is cost-TLS/cost-other, where cost-other is the cost achieved by other algorithms. (Note we want to minimize the cost); the $x$-axis is the size of the matrix.

$\rho = 0.1, 0.3, 0.6, 0.9$ respectively to check our performance. The left 2 pictures in Figure 6.2 show the running time together with the ratio TLS/FTLS for different sample densities.

We can see that the running time of FTLS is significantly smaller than that of TLS. This is because the running time of TLS depends heavily on $m$, the size of matrix $A$. When we apply sketching techniques, we significantly improve our running time. The fewer rows we sample, the faster the algorithm runs. We can see that FTLS has pretty good performance; even with 10% sample density, FTLS still performs better than LS. Moreover, the more we sample, the better accuracy we achieve.

The above matrix is extremely sparse. We also consider another class of matrices. For $k = 5, 10, \cdots, 100$, we set $A$ to be a $20k \times 2k$ matrix where $A(i, j) \sim N(0, 1)$; we set $B$ to be a $20k \times 1$ vector where $B(i) \sim N(0, 3)$. As in previous experiments, we take sample densities of $\rho = 0.1, 0.3, 0.6, 0.9$, respectively, to check our performance. The results of this experiment are shown in the right 2 pictures in Figure 6.2.

We see that compared to TLS, our FTLS sketching-based algorithm significantly reduces the running time. FTLS is still slower than LS, though, because in the FTLS algorithm we still need to solve a LS problem of the same size. However, as discussed, LS is

| Method | Cost | C-std | Time | T-std |
|---|---|---|---|---|
| TLS | 0.10 | 0 | 1.12 | 0.05 |
| LS | $10^6$ | 0 | 0.0012 | 0.0002 |
| FTLS 0.9 | 0.10 | 0.0002 | 0.16 | 0.0058 |
| FTLS 0.6 | 0.10 | 0.0003 | 0.081 | 0.0033 |
| FTLS 0.3 | 0.10 | 0.0007 | 0.046 | 0.0022 |
| FTLS 0.1 | 0.10 | 0.0016 | 0.034 | 0.0024 |

| Method | Cost | C-std | Time | T-std |
|---|---|---|---|---|
| TLS | 0.93 | 0 | 1.36 | 0.16 |
| LS | 666 | 0 | 0.0012 | 0.001 |
| FTLS 0.9 | 0.93 | 0.0032 | 0.30 | 0.025 |
| FTLS 0.6 | 0.94 | 0.0050 | 0.17 | 0.01 |
| FTLS 0.3 | 0.95 | 0.01 | 0.095 | 0.005 |
| FTLS 0.1 | 0.99 | 0.03 | 0.074 | 0.004 |

| Method | Cost | C-std | Time | T-std |
|---|---|---|---|---|
| TLS | 1.85 | 0 | 29.44 | 1.44 |
| LS | 2794 | 0 | 0.0022 | 0.001 |
| FTLS 0.9 | 1.857 | 0.001 | 3.12 | 0.081 |
| FTLS 0.6 | 1.858 | 0.002 | 1.62 | 0.054 |
| FTLS 0.3 | 1.864 | 0.006 | 0.77 | 0.027 |
| FTLS 0.1 | 1.885 | 0.019 | 0.60 | 0.017 |

| Method | Cost | C-std | Time | T-std |
|---|---|---|---|---|
| TLS | 0.550 | 0 | 125.38 | 82.9 |
| LS | 303 | 0 | 0.019 | 0.02 |
| FTLS 0.9 | 0.553 | 0.003 | 21.313 | 1.867 |
| FTLS 0.6 | 0.558 | 0.011 | 13.115 | 1.303 |
| FTLS 0.3 | 0.558 | 0.054 | 7.453 | 1.237 |
| FTLS 0.1 | 0.732 | 0.227 | 4.894 | 0.481 |

Table 6.2: Up Left: Airfoil Self-Noise. Up Right: Red wine. Down Left: White wine. Down Right: Insurance Company Benchmark. C-std is the standard deviation for cost. T-std is the standard deviation for running time.

inadequate in a number of applications as it does not allow for changing the matrix $A$. The accuracy of our FTLS algorithms is also shown.

We also conducted experiments on real datasets from the UCI Machine Learning Repository [45]. We choose datasets with regression task. Each dataset consists of input data and output data. To turn it into a total least squares problem, we simply write down the input data as a matrix $A$ and the output data as a matrix $B$, then run the corresponding algorithm on $(A, B)$. We have four real datasets : Airfoil Self-Noise [143] in Table 6.2(a), Wine Quality Red wine [145, 40] in Table 6.2(b), Wine Quality White wine [145, 40] in Table 6.2(c), Insurance Company Benchmark (COIL 2000) Data Set [144, 112] From the results,, we see that FTLS also performs well on real data: when FTLS samples

10% of the rows, the result is within 5% of the optimal result of TLS, while the running time is $20 - 40$ times faster. In this sense, FTLS achieves the advantages of both TLS and LS: FTLS has almost the same accuracy as TLS, while FTLS is significantly faster.

# Chapter 7

# A NEAR-OPTIMAL ALGORITHM FOR APPROXIMATING THE JOHN ELLIPSOID

## *7.1 Introduction*

Let $P = \{x \in \mathbb{R}^n : Ax \leqslant b\}$ be a polytope where $P$ has nonzero, finite Euclidean volume. The classical theorem of Fritz John [72] states that if $E \subseteq P$ is the ellipsoid of maximal volume contained in $P$, then $P \subseteq nE$, where $nE$ represents a dilation of the ellipsoid $E$ by a factor of $n$ about its center. Moreover, if $P$ is symmetric, then $P \subseteq \sqrt{n}E$. The maximal volume inscribed ellipsoid (MVIE) $E$ is called the John Ellipsoid, and we are interested in the problem of approximating $E$ when the polytope $P$ is centrally symmetric, i. e. $P$ can be expressed as $P = \{x \in \mathbb{R}^n : -\mathbf{1}_m \leqslant Ax \leqslant \mathbf{1}_m\}$ where $A \in \mathbb{R}^{m \times n}$ and $A$ has rank $n$.

The problem of computing the ellipsoid of maximal volume inside polytope given by a set of inequalities has a wealth of different applications, including sampling and integration [148, 28], linear bandits [23, 68], linear programming [93], cutting plane methods [80] and differential privacy [108].

Computing the John Ellipsoid additionally has applications in the field of experimental design, a classical problem in statistics [6]. Specifically, in the D-optimal design problem one seeks to maximize the determinant of the Fisher information matrix [83, 6], which turns out to be equivalent to finding the John Ellipsoid of a symmetric polytope. While this equivalence is known, e.g. [141], we include it in Section 7.3 for completeness. The problem of D-optimal design has received recent attention in the machine learning community, e.g. [2, 151, 97].

### 7.1.1 Our Contribution

Our main contribution is to develop an approximation algorithm for computing the John Ellipsoid inside a centrally symmetric polytope given by a set of inequalities. Previously, for solving the MVIE problem or its dual equivalent D-optimal design problem, researchers have developed various algorithms, such as first-order methods [81, 85, 42], and second-order interior-point methods [107, 137]. Instead of using traditional optimization methods, we apply a very simple fixed point iteration. The analysis is also simple and clean, yet the convergence rate is very fast. We state our main result as follows.

**Theorem 7.1.1** (Informal)**.** *Given $A \in \mathbb{R}^{m\times n}$, let $P$ be a centrally symmetric polytope defined as $\{x \in \mathbb{R}^n : -\mathbf{1}_m \leqslant Ax \leqslant \mathbf{1}_m\}$. For $\eta \in (0,1)$, there is an algorithm (Algorithm 4) that runs in time $O(\eta^{-1}mn^2\log(m/n))$, returning an ellipsoid $Q$ so that $\frac{1}{\sqrt{1+\eta}}\cdot Q \subseteq P \subseteq \sqrt{n}\cdot Q$.*

In Lemma 7.3.3, we show that our ellipsoid is $\eta$-close to the John Ellipsoid in a certain sense. However, if we want to get a $(1-\varepsilon)$-approximation to the maximal volume, as we discuss in Lemma 7.3.3, if we set $\eta = \varepsilon/n$,then Algorithm 4 runs in time $O(\varepsilon^{-1}mn^3\log(\frac{m}{n}))$, and when $\varepsilon$ is constant, this is comparable with the best known results $O(mn^3/\varepsilon)$ [85, 142].

Furthermore, we use sketching ideas from randomized linear algebra to speed up the algorithm so that the running time does not depend on $m$ explicitly. This will make sense if $A$ is a sparse matrix. Our result is stated as follows.

**Theorem 7.1.2** (Informal)**.** *Given $A \in \mathbb{R}^{m\times n}$, let $P$ be a centrally symmetric polytope defined as $\{x \in \mathbb{R}^n : -\mathbf{1}_m \leqslant Ax \leqslant \mathbf{1}_m\}$. For $\eta \in (0,1)$ and $\delta \in (0,1)$, there is an algorithm (Algorithm 5) that runs within $O(\frac{1}{\eta}\log\frac{m}{\delta})$ iterations, returning an ellipsoid $Q$ so that with probability at least $1-\delta$, $\frac{1}{\sqrt{1+\eta}}\cdot Q \subseteq P \subseteq \sqrt{n}\cdot Q$. Moreover, each iteration involves in solving $O(\frac{1}{\eta})$ linear systems of the form $A^\top WAx = b$ where $W$ is some diagonal matrix.*

Algorithm 5 is near optimal, because in order to verify the correctness of the result, we need to compute the *leverage scores* of some weighted version of $A$. The best known algo-

rithm for approximating leverage scores needs to solve $\widetilde{O}(\frac{1}{\eta^2})$ many linear systems [133, 50, 30, 105]. One key advantage of our algorithm is that it reduces the problem of computing the John ellipsoid to solving a relatively small number of linear systems. Therefore, it allows the user to apply a linear systems solver tailored for the given matrix $A$. For example, if $A$ is tall, one can apply sketching techniques to solve the linear systems in nearly linear time [153]; if each row of A has only two non-zeros, one can apply Laplacian solvers [41, 79, 35, 87, 86]. In the matlab code for this computation that is included in Appendix B of this thesis, we use the Cholesky decomposition which is very fast for many sparse matrices $A$ in practice.

Finally, we validate our algorithm on both synthetic and real data sets. Experiments show that our algorithm outperform all previous algorithms, which matches the theoretical calculation. We shall stress that our program can handle very large sparse matrices, which seems to be difficult for previous implementations.

### *7.1.2 Related Works*

There is a long line of research on computing the maximal volume ellipsoid inside polytopes given by a list of linear inequalities. We note that Khachiyan and Todd [82] presented a linear time reduction from the problem of computing a minimum volume enclosing ellipsoid (MVEE) of a set of points to the maximal volume inscribed ellipsoid problem; therefore, these algorithms also apply for approximating the John Ellipsoid.

Using an interior-point algorithm, Nesterov and Nemirovskii [107] showed that a $1+\varepsilon$ approximation of MVEE can be computed in time $O(m^{2.5}(n^2+m)\log(\frac{m}{\varepsilon}))$. Khachiyan and Todd [82] subsequently improved the runtime to $O(m^{3.5}\log(\frac{m}{\varepsilon})\cdot\log(\frac{n}{\varepsilon}))$. Later on, Nemirovski [106] and Anstreicher [5] independently obtained an $O(m^{3.5}\log\frac{m}{\varepsilon})$ algorithm. The best previous algorithms by Kumar and Yildirim, Todd and Yıldırım [85, 142] run in time $O(mn^3/\varepsilon)$. We refer readers to [141] for a comprehensive introduction and overview.

Computing the minimum volume enclosing ellipsoid of a set of points is the dual prob-

lem of D-optimal design. By generalizing the smoothness condition on first order methods, Lu, Freund and Nesterov [97] managed to solve D-optimal design problem within $O(\frac{m}{\varepsilon}\log(\frac{n}{\varepsilon}))$ many iterations. However, in the dense case, their iteration costs $O(mn^2)$ time, which leads to larger running time comparing to [85, 142]. Gutman and Peña [65] applied Bregman proximal method on the D-optimal design problem and observe accelerated convergence rate in their numerical experiments; however, they did not prove that their experimental parameter settings satisfy the assumption of their algorithm[1].

A natural version of the D-optimal design problem is to require an integral solution. The integral variant is shown to be **NP**-hard [24], although recently approximation algorithms have been developed [2, 122]. In our context, this means the weight vector $w \in \mathbb{R}^m$ is the integral optimal solution to (7.2), where the sum of the weights is some specified integral parameter $k$.

Several Markov chains for sampling convex bodies have well understood performance guarantees based upon the roundedness of the convex body. If $B_n \subseteq K \subseteq R \cdot B_n$, then the mixing time of hit-and-run and the ball walk are both $O(n^2R^2)$ steps [96, 73]. Thus, placing a convex body in John position guarantees the walks mix in $O(n^4)$ steps, and $O(n^3)$ steps if the body is symmetric; this transformation is used in practice with the convex body to be a polytope [66]. Generating the John Ellipsoid, with a fixed center point, has also been employed as a proposal distribution for a Markov chain [28, 64].

We build our even faster algorithm via sketching techniques. Sketching has been successfully applied to speed up different problems, such as linear programs [94], clustering [34, 132], low rank approximations [30, 105, 20, 31, 115, 128], linear regression [30, 105, 36, 32, 110, 4], total least regression [48], tensor regression [95, 47] and tensor decomposition [150, 127, 130]. Readers may refer to [153] for a comprehensive survey on sketching technique. We use sketching techniques to speed up computing leverage scores. This idea was first used in [133].

---

[1]We are grateful to Gutman and Peña who provided us their code for testing, which had D-optimal design as only one application.

Previous research on the MVEE problem did take advantage of the sparsity of the input matrix $A$, and to the best of our knowledge, our algorithm is the first one that is able to deal with large sparse input. It would be interesting if we could apply sketching techniques to further speed up existing algorithms.

*Relation with [38]*

Our work is greatly inspired by the paper [38] by Cohen and Peng. The $\ell_p$ *Lewis Weights* $\overline{w}$ for matrix $A \in \mathbb{R}^{m\times n}$ is defined as the unique vector $\overline{w}$ so that for $i \in [m]$,

$$a_i^\top \left(A^\top \mathsf{diag}(\overline{w})^{1-2/p} A\right)^{-1} a_i = \overline{w}_i^{2/p}.$$

It is known that computing the $\ell_\infty$ Lewis Weight is equivalent to computing the maximal volume inscribed ellipsoid. Cohen and Peng propose an algorithm for approximating Lewis Weights for all $p < 4$. Their algorithm is an iterative algorithm that is very similar to our Algorithm 4, and the convergence is proved by arguing the iteration mapping is contractive. The main difference is that they outputs the weights in the last round, while our Algorithm 4 takes the average over all rounds and outputs the averaging weights, which allows us to conduct a convexity analysis and deal with the $\ell_\infty$ case.

## *7.2 Preliminaries*

In this section we introduce notations and preliminaries. We use $N(\mu, \sigma^2)$ to represent the normal distribution with mean $\mu$ and variance $\sigma^2$.

### *7.2.1 Multivariate Calculus*

Let $f : \mathbb{R}^m \to \mathbb{R}^n$ be a differentiable function. The *directional derivative* of $f$ in the direction $h$ is defined as

$$Df(x)[h] := \frac{\mathsf{d}f(x+th)}{\mathsf{d}t}\Big|_{t=0}.$$

We can also define a high order directional derivative as

$$D^k f(x)[h_1, \cdots, h_k] := \frac{\mathsf{d}^k f(x + \sum_{i=1}^k t_i h_i)}{\mathsf{d}t_1 \mathsf{d}t_2 \cdots \mathsf{d}t_k}\Big|_{t_1=0,\ldots,t_k=0}.$$

The following two properties of directional derivatives will be useful.

**Proposition 7.2.1.**

- *(Chain rule)* $Df(g(x))[h] = f'(g(x)) \cdot Dg(x)[h]$.
- *Let* $f(X) = X^{-1}$ *where* $X \in \mathbb{R}^{n\times n}$. *For* $H \in \mathbb{R}^{n\times n}$, $Df(X)[H] = X^{-1}HX^{-1}$.

### *7.2.2 Gamma Function*

The Gamma ($\Gamma$) function is well-known and defined as

$$\Gamma(z) = \int_0^{+\infty} x^{z-1} e^{-x} \mathsf{d}x.$$

We need the following result on the Gamma function.

**Lemma 7.2.2** (Corollary 1 of [71])**.** *For all* $x > 0$ *and* $0 \leqslant y \leqslant 1$,

$$x(x+y)^{y-1} \leqslant \frac{\Gamma(x+y)}{\Gamma(x)} \leqslant x^y.$$

### *7.2.3 Tail Bound for $\chi^2$ Distribution*

The chi-square ($\chi^2$) distribution is the distribution of a summation of the squares of independent standard normal random variables.

**Definition 7.2.3** ($\chi^2$ distribution)**.** *Fix integer* $n \geqslant 1$. *Let* $X_1, \cdots, X_n$ *be independent random variables where* $X_i \sim N(0,1)$. *Let* $Y = \sum_{i=1}^n X_i^2$. *Then we call the distribution of* $Y$ *the* $\chi^2$ *distribution of degrees* $n$. *Moreover, the probability density function of* $Y$ *is*

$$f(x) = \frac{1}{2^{n/2}\Gamma(n/2)} x^{n/2-1} e^{-x/2}.$$

We need the following version of concentration for the $\chi^2$ distribution.

**Lemma 7.2.4** (Lemma 1 in [91])**.** *Let* $X \sim \chi^2(n)$ *be a* $\chi^2$ *distribution with* $n$ *degrees of freedom. Then for* $t > 0$,

$$\mathbf{Pr}[X - n \geqslant 2\sqrt{nt} + 2t] \leqslant e^{-t}.$$

### *7.3 Problem Formulation*

In this section, we formally define the problem of computing the John Ellipsoid of a symmetric polytope. For fixed integers $m, n$, let $a_1, \cdots, a_m$ be $m$ vectors in $\mathbb{R}^n$. Let $P = \{x \in \mathbb{R}^n : |a_i^\top x| \leqslant 1, i \in [m]\}$ be a symmetric convex polytope, where $[m]$ denotes the set $\{1, 2, \cdots, m\}$. We assume that $A = (a_1\, a_2\, \cdots\, a_m)^\top$ has full rank. By symmetry, we know that the maximal volume ellipsoid inside the polytope should be centered at the origin. Any ellipsoid $E$ centered at the origin can be expressed by $x^\top G^{-2} x \leqslant 1$, where $G$ is a positive definite matrix. Note that the volume of $E$ is proportional to $\det G$, and an ellipsoid $E$ is contained in polytope $P$ if and only if for $i \in [m]$, $\max_{x \in E} |a_i^\top x| \leqslant 1$. For any $x \in E$, we can write $x = Gy$ where $\|y\|_2 \leqslant 1$. Hence

$$\max_{x \in E} |a_i^\top x| = \max_{\|y\|_2 \leqslant 1} |a_i^\top G y| = \max_{\|y\|_2 \leqslant 1} \|G a_i\|_2 \cdot \|y\|_2 = \|G a_i\|_2$$

Therefore, we can compute the John Ellipsoid of $P$ by solving the following optimization program:

$$\begin{aligned} \text{Maximize} \quad & \log\det G, \\ \text{subject to:} \quad & G \succeq 0, \\ & \|G a_i\|_2 \leqslant 1, \quad \forall i \in [m]. \end{aligned} \tag{7.1}$$

It turns out that the optimal ellipsoid satisfies $G^{-2} = A^\top \mathsf{diag}(w) A$, where $w \in \mathbb{R}^m_{\geqslant 0}$ is the optimal solution of the program

$$\begin{aligned} \text{Minimize} \quad & \sum_{i=1}^m w_i - \log\det\left(\sum_{i=1}^m w_i a_i a_i^\top\right) - n, \\ \text{subject to:} \quad & w_i \geqslant 0, \quad \forall i \in [m]. \end{aligned} \tag{7.2}$$

Actually program (7.2) is the Lagrange dual of program (7.1). Moreover, we have the following optimality criteria for $w$ in the above program.

**Lemma 7.3.1** (Optimality criteria, Proposition 2.5 in [141])**.** *A weight $w$ is optimal for pro-*

*gram* (7.2) *if and only if*

$$\begin{aligned}
&\sum_{i=1}^{m} w_i = n, \\
&a_i^\top \left( \sum_{i=1}^{m} w_i a_i a_i^\top \right)^{-1} a_i = 1, && \text{if } w_i \neq 0; \\
&a_i^\top \left( \sum_{i=1}^{m} w_i a_i a_i^\top \right)^{-1} a_i < 1, && \text{if } w_i = 0.
\end{aligned}$$

Computing the John Ellipsoid is closely related to the D-optimal design problem [6, 22, 141, 65]. For the D-optimal design problem, we are given input $X \in \mathbb{R}^{n \times m}$ where $m > n$, and we want to solve program,

$$\begin{aligned}
\text{Maximize} \quad & \log\det\left(X \mathsf{diag}(v) X^\top\right), \\
\text{subject to:} \quad & v_i \geqslant 0, \ \forall i \in [m] \\
& \sum_{i=1}^{m} v_i = 1
\end{aligned} \tag{7.3}$$

We emphasize that program (7.3) and program (7.2) are equivalent, in the following sense. By Lemma 7.3.1, we can rewrite program (7.2) as minimizing $n \log n - \log\det(A^\top \mathsf{diag}(w) A)$, subject to $w_i \geqslant 0$ for $i \in [m]$ and $\sum_{i=1}^{m} w_i = n$. By setting $v_i = \frac{w_i}{n}$, we obtain program (7.3). Thus, optimal solutions to programs (7.2) and (7.3) are equivalent up to a multiplicative factor $n$.

We can also talk about an approximate John Ellipsoid.

**Definition 7.3.2.** *For $\varepsilon > 0$, we say $w \in \mathbb{R}^m_{\geqslant 0}$ is a $(1+\varepsilon)$-approximation of program* (7.2) *if $w$ satisfies*

$$\begin{aligned}
&\sum_{i=1}^{m} w_i = n, \\
&a_i^\top \left( \sum_{i=1}^{m} w_i a_i a_i^\top \right)^{-1} a_i \leqslant 1 + \varepsilon, \qquad \forall i \in [m].
\end{aligned}$$

Lemma 7.3.3 gives a geometric interpretation of the approximation factor in Definition 7.3.2. Recall that the exact John Ellipsoid $Q^*$ of $P$ satisfies $Q^* \subseteq P \subseteq \sqrt{n} \cdot Q^*$.

**Lemma 7.3.3** ($(1+\varepsilon)$-approximation is good rounding)**.** *Let $w$ be a $(1+\varepsilon)$-approximation of* (7.2)*. Define $Q$ as $\{x : x^\top A^\top \mathsf{diag}(w) A x \leqslant 1\}$. Then*

$$\frac{1}{\sqrt{1+\varepsilon}} \cdot Q \subseteq P \subseteq \sqrt{n} \cdot Q.$$

*Moreover,* $\mathbf{vol}\left(\frac{1}{\sqrt{1+\varepsilon}} Q\right) \geqslant e^{-n\varepsilon/2} \cdot \mathbf{vol}\left(Q^*\right)$.

*Proof.* Let $G = \left(A^\top \mathsf{diag}(w) A\right)^{-\frac{1}{2}}$ and suppose that $x \in \frac{1}{\sqrt{1+\varepsilon}} Q$. Then, we have that $x^\top G^{-2} x \leqslant \frac{1}{1+\varepsilon}$. So,

$$|Ax|_i = \langle a_i, x \rangle = \langle G a_i, G^{-1} x \rangle \leqslant \|G a_i\|_2 \|G^{-1} x\|_2 \leqslant \frac{\|G a_i\|_2}{\sqrt{1+\varepsilon}}.$$

Since $\|G a_i\|_2^2 = a_i^\top (\sum_{i=1}^m w_i a_i a_i^\top)^{-1} a_i \leqslant 1+\varepsilon$, then $|Ax|_i \leqslant 1$ and $x \in P$.

On the other hand, for $x \in P$, we have that $|Ax|_i \leqslant 1$. Hence

$$x^\top G^{-2} x = x^\top A^\top \mathsf{diag}(w) A x = \sum_{i=1}^m w_i |Ax|_i^2 \leqslant \sum_{i=1}^m w_i = n.$$

So $P \subseteq \sqrt{n} \cdot Q$.

Finally, since $\frac{1}{\sqrt{1+\varepsilon}} \cdot Q$ is contained in $P$, $G' = ((1+\varepsilon) A^\top \mathsf{diag}(w) A)^{-\frac{1}{2}}$ is a feasible solution to program (7.1). Moreover $w$ is a feasible solution to program (7.2). So by duality of program (7.1) and (7.2), we see that the duality gap is at most

$$\left(n - \log\det\left(\sum_{i=1}^m w_i a_i a_i^\top\right) - n\right) - \log\det\left((1+\varepsilon)\sum_{i=1}^m w_i a_i a_i^\top\right)^{-1} = n\log(1+\varepsilon) \leqslant n\varepsilon.$$

Let the matrix representation of $Q^*$ be $x^\top G_*^{-2} x \leqslant 1$, then by optimality of $G_*$ and the duality gap, we have

$$\log\det\left((G')^{-2}\right) \geqslant \log\det(G_*^2) - n\varepsilon.$$

Since $\mathbf{vol}(\frac{1}{\sqrt{1+\varepsilon}} \cdot Q)$ is proportional to $\det(G')^{-1}$, we conclude that $\mathbf{vol}(\frac{1}{\sqrt{1+\varepsilon}} Q) \geqslant e^{-n\varepsilon/2} \mathbf{vol}(Q^*)$.

□

**Algorithm 4:** Approximate John Ellipsoid inside symmetric polytopes

**Input:** A symmetric polytope given by $-\mathbf{1}_m \leqslant Ax \leqslant \mathbf{1}_m$, where $A \in \mathbb{R}^{m\times n}$ has rank $n$.

**Result:** Approximate John Ellipsoid inside the polytope.

Initialize $w_i^{(1)} = \frac{n}{m}$ for $i = 1, \cdots, m$.

**for** $k = 1, \cdots, T-1$, **do**

  **for** $i = 1, \cdots, m$ **do**

    `// We can use sketch techniques to further speed up.`

    $w_i^{(k+1)} = w_i^{(k)} \cdot a_i^\top (A^\top \mathsf{diag}(w^{(k)}) A)^{-1} a_i$

$w_i = \frac{1}{T}\sum_{k=1}^{T} w_i^{(k)}$ for $i = 1, \cdots, m$.

$W = \mathsf{diag}(w)$. (i. e. $W$ is a diagonal matrix with the entries of $w$)

**return** $A^\top W A$

### *7.4 Main Algorithm*

In this section, we present Algorithm 4 for approximating program (7.2) and analyze its performance.

Let $\sigma : \mathbb{R}^m \to \mathbb{R}^m$ be the function defined as $\sigma(v) = (\sigma_1(v), \sigma_2(v), \cdots, \sigma_m(v))$ where for $i \in [m]$,

$$\sigma_i(v) = a_i^\top \left( \sum_{j=1}^{m} v_j a_j a_j^\top \right)^{-1} a_i = a_i^\top (A^\top \mathsf{diag}(v) A)^{-1} a_i. \tag{7.4}$$

Let $w^*$ be the optimal solution to program (7.2). By Lemma (7.3.1), $w^*$ satisfies $w_i^*(1 - \sigma_i(w^*)) = 0$, or equivalently

$$w_i^* = w_i^* \cdot \sigma_i(w^*) \tag{7.5}$$

Inspired by (7.5), we use the fixed point iteration $w_i^{(k+1)} = w_i^{(k)} \cdot \sigma_i(w^{(k)})$ for $k \in [T-1]$ and $i \in [m]$. Observe that $w_i^{(k)}$ has very nice properties. Actually, by setting $B^{(k)} = \sqrt{\mathsf{diag}(w^{(k)})} \cdot A$, we can rewrite $w_i^{(k)}$ as $(B_i^{(k)})^\top \left( (B^{(k)})^\top B^{(k)} \right)^{-1} B_i^{(k)}$, hence $w_i^{(k)}$ is actually

the *leverage score* of the $i$-th row of the matrix $B^{(k)}$ [36]. From the well-known properties of leverage scores, we have

**Lemma 7.4.1** (Properties of leverage scores, e.g. see Section 3.3 of [36])**.** *For $k \in [T]$ and $i \in [m]$, we have $0 \leqslant w_i^{(k)} \leqslant 1$. Moreover, $\sum_{i=1}^m w_i^{(k)} = n$.*

In order to show that Algorithm 4 provides a good approximation of the John Ellipsoid, in the sense of Definition 7.3.2, we need to argue that for the output $w$ of Algorithm 4, $\sigma_i(w) \leqslant 1 + \varepsilon$. Our main result is the following theorem.

**Theorem 7.4.2** (Main Result)**.** *Let $w$ be the output of Algorithm 4 in line* (5)*. For all $\varepsilon \in (0, 1)$, when $T = \frac{2}{\varepsilon} \log \frac{m}{n}$, we have for $i \in [m]$,*

$$\sigma_i(w) \leqslant 1 + \varepsilon.$$

*Moreover,*

$$\sum_{i=1}^m w_i = n.$$

*Therefore, Algorithm 4 provides a $(1 + \varepsilon)$-approximation to program* (7.2)*.*

We now analyze the running time of Algorithm 4.

**Theorem 7.4.3** (Performance of Algorithm 4)**.** *For all $\varepsilon \in (0, 1)$, we can find a $(1 + \varepsilon)$-approximation of John Ellipsoid inside a symmetric convex polytope in time $O\left(\varepsilon^{-1} mn^2 \log \frac{m}{n}\right)$.*

*Proof.* The main loop is executed $T = O(\frac{1}{\varepsilon} \log(\frac{m}{n}))$ times, and inside each loop, we can first use $O(mn)$ time to compute $B^{(k)} := (W^{(k)})^{\frac{1}{2}} A$, then compute $(B^{(k)})^\top B^{(k)}$ in $O(mn^2)$ time. To see why we introduce $B^{(k)}$, observe that $(B^{(k)})^\top B^{(k)} = A^\top W^{(k)} A$. Now we can compute the Cholesky decomposition of $(B^{(k)})^\top B^{(k)}$ in time $O(n^3)$, and use the Cholesky decomposition to compute $c_i := ((B^{(k)})^\top B^{(k)})^{-1} a_i = (A^\top W^{(k)} A)^{-1} a_i$ in time $O(n^2)$ for each $i \in [m]$. Finally, we can compute $w_i^{(k+1)}$ by computing $w_i^{(k)} \cdot a_i^\top c_i$ in time $O(n)$. This is valid since $w_i^{(k)} \cdot a_i^\top c_i = w_i^{(k)} \cdot a_i^\top (A^\top W^{(k)} A)^{-1} a_i = w_i^{(k)} \sigma_i(w^{(k)})$. To summarize, in each iteration we use $O(mn^2 + n^3 + mn^2) = O(mn^2)$ time, hence the overall running time is as stated. □

Now we turn to proving Theorem 7.4.2. The proof of Theorem 7.4.2 relies on the following important observation.

**Lemma 7.4.4** (Convexity)**.** *For $i = 1, \cdots, m$, let $\phi_i : \mathbb{R}^m \to \mathbb{R}$ be the function defined as*

$$\phi_i(v) = \log \sigma_i(v) = \log \left( a_i^\top \left( \sum_{j=1}^m v_j a_j a_j^\top \right)^{-1} a_i \right).$$

*Then $\phi_i$ is convex.*

*Proof.* We first prove a strengthened result: for fixed $a \in \mathbb{R}^n$, the function $f : S_{++}^n \to \mathbb{R}$ defined as $f(M) = \log(a^\top M^{-1} a)$ is convex. Here $S_{++}^n$ is the set of all positive definite $n \times n$ matrices. Notice that $S_{++}^n$ is an open set, so we can differentiate $f$.

We argue that it is sufficient to show for all $M \in GL_n$ and all $H \in \mathbb{R}^{n \times n}$, the second order directional derivative $D^2 f(M)[H, H]$ is non-negative. This is because $D^2 f(M)[H, H] = H^\top \nabla^2 f(M) H$. So if for all $H$ we have $H^\top \nabla^2 f(M) H \geqslant 0$, then $\nabla^2 f(M) \succeq 0$, which is precisely the convexity condition.

Let us do some computation with Proposition 7.2.1.

$$Df(M)[H] = -\frac{a^\top M^{-1} H M^{-1} a}{a^\top M^{-1} a},$$

and

$$D^2 f(M)[H, H] = \frac{2a^\top M^{-1} H M^{-1} H M^{-1} a \cdot a^\top M^{-1} a - (a^\top M^{-1} H M^{-1} a)^2}{(a^\top M^{-1} a)^2}.$$

By Cauchy-Schwartz inequality, we have

$$\begin{aligned} a^\top M^{-1} H M^{-1} H M^{-1} a \cdot a^\top M^{-1} a =& \|M^{-\frac{1}{2}} H M^{-1} a\|_2^2 \cdot \|M^{-\frac{1}{2}} a\|_2^2 \\ \geqslant& (\langle M^{-\frac{1}{2}} H M^{-1} a, M^{-\frac{1}{2}} a \rangle)^2 \\ =& (a^\top M^{-1} H M^{-1} a)^2. \end{aligned}$$

Hence $D^2 f(M)[H, H] \geqslant \frac{a^\top M^{-1} H M^{-1} H M^{-1} a \cdot a^\top M^{-1} a}{(a^\top M^{-1} a)^2} \geqslant 0$ for all $M \in GL_n$ and all $H \in \mathbb{R}^{n \times n}$.

Now we are ready to work on $\phi_i$. For all $v, v'$ in the domain of $\phi_i$, let $M = \sum_{i=1}^m v_i a_i a_i^\top$ and $M' = \sum_{i=1}^m v_i' a_i a_i^\top$. Then for all $\lambda \in [0,1]$,

$$\begin{aligned}
\phi_i(\lambda v + (1-\lambda)v') &= \log a_i^\top \left( \sum_{i=1}^m (\lambda v_i + (1-\lambda)v_i') a_i a_i^\top \right)^{-1} a_i \\
&= \log a_i^\top \left( \lambda \sum_{i=1}^m v_i a_i a_i^\top + (1-\lambda) \sum_{i=1}^m v_i' a_i a_i^\top \right)^{-1} a_i \\
&= f(\lambda M + (1-\lambda)M') \\
&\leqslant \lambda f(M) + (1-\lambda) f(M') \qquad \text{because } f \text{ is convex} \\
&= \lambda \phi_i(v) + (1-\lambda)\phi_i(v').
\end{aligned}$$

So $\phi_i$ is also convex. □

Now that $\phi_i$ is convex, we can apply Jensen's inequality to get Lemma 7.4.5.

**Lemma 7.4.5** (Telescoping)**.** *Fix $T$ as the number of times that the main loop is executed in Algorithm 4. Let $w$ be the output in line* (5) *of Algorithm 4. Then for $i \in [m]$,*

$$\phi_i(w) \leqslant \frac{1}{T} \log \frac{m}{n}$$

*Proof.* Recall that $w = \frac{1}{T} \sum_{k=1}^T w^{(k)}$. By Lemma 7.4.4, $\phi_i$ is convex, and so

$$\begin{aligned}
\phi_i(w) &= \phi_i \left( \frac{1}{T} \sum_{k=1}^T w^{(k)} \right) \\
&\leqslant \frac{1}{T} \sum_{k=1}^T \phi_i(w^{(k)}) && \text{by Jensen's inequality} \\
&= \frac{1}{T} \sum_{k=1}^T \log \sigma_i(w^{(k)}) && \text{by definition of } \phi_i \text{ function} \\
&= \frac{1}{T} \sum_{k=1}^T \log \frac{w_i^{(k+1)}}{w_i^{(k)}} \\
&= \frac{1}{T} \log \frac{w_i^{(T+1)}}{w_i^{(1)}} \\
&\leqslant \frac{1}{T} \log \frac{m}{n} && \text{by Lemma 7.4.1 and the initialization of } w^{(1)}
\end{aligned}$$

□

Now we are ready to prove Theorem 7.4.2.

*Proof of Theorem 7.4.2.* Set $T = \frac{2}{\varepsilon}\log\frac{m}{n}$. By Lemma 7.4.5, we have for $i \in [m]$,

$$\log \sigma_i(w) = \phi_i(w) \leqslant \frac{1}{T}\log\frac{m}{n} = \frac{\varepsilon}{2} \leqslant \log(1+\varepsilon)$$

where the last step uses the fact that when $0 < \varepsilon < 1$, $\frac{\varepsilon}{2} \leqslant \log(1+\varepsilon)$. This gives us $\sigma_i(w) \leqslant 1+\varepsilon$.

On the other hand, from Lemma 7.4.1 we have $\sum_{i=1}^m w_i^{(k)} = n$. Hence

$$\sum_{i=1}^m w_i = \sum_{i=1}^m \frac{1}{T}\sum_{k=1}^T w_i^{(k)} = n$$

□

### *7.5 Faster Algorithm for Computing John Ellipsoid for Sparse Matrix*

It is possible to further improve Algorithm 4 by applying sketching techniques from randomized linear algebra. In this section we present our accelerated algorithm, Algorithm 5 and analyze its performance. Recall that Algorithm 4 uses the iterating rule $w \leftarrow w \cdot \sigma(w)$ where $\sigma_i(w) = a_i^\top (A^\top \mathsf{diag}(w) A)^{-1} a_i$. With our setting of $B^{(k)}$, we have

$$(B^{(k)})^\top B^{(k)} = A^\top \sqrt{W^{(k)}} \cdot \sqrt{W^{(k)}} A = A^\top W^{(k)} A.$$

So $\widehat{w}_i^{(k+1)} = \|B^{(k)}((B^{(k)})^\top B^{(k)})^{-1}(\sqrt{w_i^{(k)}} a_i)\|_2^2 = w_i \cdot \sigma_i(w)$ is just what we do in Algorithm 4. Hence from Lemma 7.4.1, we obtain the following properties about $\widehat{w}_i^{(k)}$.

**Proposition 7.5.1** (Bound on $\widehat{w}^{(k)}$)**.** *For completeness, define $\widehat{w}^{(1)} = w^{(1)}$. For $k \in [T]$ and $i \in [m]$, $0 \leqslant \widehat{w}_i^{(k)} \leqslant 1$. Moreover, $\sum_{i=1}^m \widehat{w}_i^{(k)} = n$.*

However, $B^{(k)}$ is an $m$ by $n$ matrix, so it is computationally expensive to compute $\widehat{w}_i^{(k+1)}$. The trick we use here, which was first introduced by Spielman and Srivastava [133], is to introduce a random Gaussian matrix $S$ with $s$ rows to speed up the computation. Of course, this will introduce extra error, however we can prove that the overall result has good concentration.

**Algorithm 5:** Faster Algorithm for approximating John Ellipsoid inside symmetric polytopes

**Input:** A symmetric polytope given by $-\mathbf{1}_m \leqslant Ax \leqslant \mathbf{1}_m$, where $A \in \mathbb{R}^{m \times n}$

**Result:** Approximate John Ellipsoid inside the polytope

1 initialize $w_i^{(1)} = \frac{n}{m}$ for $i = 1, \cdots, m$.

2 **for** $k = 1, \cdots, T-1$, **do**

3 $W^{(k)} = \mathsf{diag}(w^{(k)})$.

4 $B^{(k)} = \sqrt{W^{(k)}} A$.

5 Let $S^{(k)} \in \mathbb{R}^{s \times m}$ be a random matrix where each entry is chosen i.i.d from $N(0,1)$, i. e. the standard normal distribution.

6 **for** $i = 1, \cdots, m$ **do**

`// Ideally we want to compute`

// $\widehat{w}_i^{(k+1)} = \|B^{(k)}((B^{(k)})^\top B^{(k)})^{-1}(\sqrt{w_i^{(k)}} a_i)\|_2^2$.

`// But this is expensive, so we use sketching technique to speed up.`

7 $w_i^{(k+1)} = \frac{1}{s} \cdot \|S^{(k)} B^{(k)}((B^{(k)})^\top B^{(k)})^{-1}(\sqrt{w_i^{(k)}} a_i)\|_2^2$.

8 $w_i = \frac{1}{T} \sum_{k=1}^{T} w_i^{(k)}$ for $i = 1, \cdots, m$.

9 $v_i = \frac{n}{\sum_{j=1}^{m} w_j} w_i$ for $i = 1, \cdots, m$.

10 $V = \mathsf{diag}(v)$.

11 **return** $A^\top V A$

### *7.5.1 Approximation Guarantee*

Since Algorithm 5 is a randomized algorithm, we need to argue that for the output $v$ of Algorithm 5, $\sigma_i(v) \leqslant 1 + \varepsilon$ with high probability. Our main result in this section is

**Theorem 7.5.2** (Main result). *Let $w$ be the output in line* (8) *of Algorithm 5. For all $\varepsilon, \delta \in (0,1)$,*

*when* $T = \frac{10}{\varepsilon}\log\frac{m}{\delta}$ *and* $s = \frac{80}{\varepsilon}$, *we have*

$$\mathbf{Pr}[\forall i \in [m], \sigma_i(w) \leqslant 1+\varepsilon] \geqslant 1-\delta.$$

*Moreover, before rescaling at the end,*

$$\mathbf{Pr}\left[\sum_{i=1}^{m} w_i \leqslant (1+\varepsilon)n\right] \geqslant 1-\delta.$$

By scaling $w$ so that $\sum_{i=1}^{m} w_i = n$, we have

**Theorem 7.5.3** (Approximation guarantee)**.** *For all* $\varepsilon, \delta \in (0,1)$, *When* $T = O(\frac{1}{\varepsilon}\log\frac{m}{\delta})$ *and* $s = O(\frac{1}{\varepsilon})$, *Algorithm 5 provides a* $(1+\varepsilon)^2$*-approximation to program* (7.2) *with probability at least* $1-2\delta$.

*Proof.* On line (9) of Algorithm 5 we set $v = \frac{n}{\sum_{i=1}^{m} w_i} w$, hence $\sum_{i=1}^{m} v_i = n$.

From Theorem 7.5.2, with probability at least $1-2\delta$, $\forall i \in [m], \sigma_i(w) \leqslant 1+\varepsilon$, and $\sum_{i=1}^{m} w_i \leqslant (1+\varepsilon)n$. Therefore for all $i \in [m]$,

$$\begin{aligned}
\sigma_i(v) &= a_i^\top (A^\top \mathsf{diag}(v) A)^{-1} a_i \\
&= a_i^\top \left(\frac{n}{\sum_{i=1}^{m} w_i} \cdot A^\top \mathsf{diag}(w) A\right)^{-1} a_i \\
&= \frac{\sum_{i=1}^{m} w_i}{n} \sigma_i(w) \\
&\leqslant (1+\varepsilon)^2.
\end{aligned}$$

□

From now on we focus on proving Theorem 7.5.2. Recall that $\phi_i(w) = \log \sigma_i(w)$ for $i \in [m]$. Similar to Lemma 7.4.5, we can prove the following lemma with the convexity of $\phi_i$.

**Lemma 7.5.4** (Telescoping)**.** *Fix* $T$ *as the number of main loops executed in Algorithm 5. Let* $w$ *be the output at line* (8) *of Algorithm 5. Then for* $i \in [m]$,

$$\phi_i(w) \leqslant \frac{1}{T}\log\frac{m}{n} + \frac{1}{T}\sum_{k=1}^{T} \log\frac{\widehat{w}_i^{(k)}}{w_i^{(k)}}.$$

*Proof.* Recall that $w = \frac{1}{T}\sum_{k=1}^{T} w^{(k)}$. By Lemma 7.4.4, $\phi_i$ is convex, so we can apply Jensen's inequality to obtain

$$\begin{aligned}
\phi_i(w) &\leqslant \frac{1}{T}\sum_{k=1}^{T} \phi_i(w^{(k)})) && \text{Jensen's inequality} \\
&= \frac{1}{T}\sum_{k=1}^{T} \log \sigma_i(w^{(k)}) && \text{by definition of } \phi_i \text{ function} \\
&= \frac{1}{T}\sum_{k=1}^{T} \log \frac{\widehat{w}_i^{(k+1)}}{w_i^{(k)}} && \widehat{w}_i^{(k+1)} = w_i^{(k)}\sigma_i(w^{(k)}) \\
&= \frac{1}{T}\sum_{k=1}^{T} \log \frac{\widehat{w}_i^{(k+1)}}{\widehat{w}_i^{(k)}} \frac{\widehat{w}_i^{(k)}}{w_i^{(k)}} \\
&= \frac{1}{T}(\sum_{k=1}^{T} \log \frac{\widehat{w}_i^{(k+1)}}{\widehat{w}_i^{(k)}} + \sum_{k=1}^{T} \log \frac{\widehat{w}_i^{(k)}}{w_i^{(k)}}) \\
&= \frac{1}{T}\log \frac{\widehat{w}_i^{(T+1)}}{\widehat{w}_i^{(1)}} + \frac{1}{T}\sum_{k=1}^{T} \log \frac{\widehat{w}_i^{(k)}}{w_i^{(k)}} \\
&\leqslant \frac{1}{T}\log \frac{m}{n} + \frac{1}{T}\sum_{k=1}^{T} \log \frac{\widehat{w}_i^{(k)}}{w_i^{(k)}}. && \text{by Proposition 7.5.1 and the initialization of } w^{(1)}
\end{aligned}$$

□

From Lemma 7.5.4, we can bound the expectation of $\phi_i$ directly.

**Lemma 7.5.5** (Expectation of $\log \sigma_i$)**.** *If $s$ is even, then*

$$\mathbf{E}[\phi_i(w)] = \mathbf{E}\left[\log a_i^\top \left(\sum_j w_j a_j a_j^\top\right)^{-1} a_i\right] \leqslant \frac{1}{T}\log\frac{m}{n} + \frac{2}{s}.$$

*where the randomness is taken over the sketching matrices $\{S^{(k)}\}_{k=1}^{T-1}$.*

*Proof.* Recall the update rule

$$w_i^{(k+1)} = \frac{1}{s}\cdot \|S^{(k)}B^{(k)}((B^{(k)})^\top B^{(k)})^{-1}(\sqrt{w_i^{(k)}}a_i)\|_2^2.$$

Let $y_i^{(k)} = B^{(k-1)}((B^{(k-1)})^\top B^{(k-1)})^{-1}(\sqrt{w_i^{(k-1)}}a_i)$ be a vector of size $m$. Then $\widehat{w}_i^{(k)} = \|y_i^{(k)}\|_2^2$, and $w_i^{(k)} = \frac{1}{s}\|S^{(k)}y_i^{(k)}\|_2^2$, where each entry of $S^{(k)}$ is chosen i.i.d from $N(0,1)$.

Fix $y_i^{(k)}$. Let us consider the distribution of $w_i^{(k)}$. We first consider 1 coordinate of $S^{(k)}y_i^{(k)}$, which is $(S^{(k)}y_i^{(k)})_j = \sum_{t=1}^m S_{jt}^{(k)}(y_i^{(k)})_t$. Since each $S_{jt}^{(k)}$ is chosen from $N(0,1)$, $S_{jt}^{(k)}(y_i^{(k)})_t$ follows the distribution $N(0,(y_i^{(k)})_t^2)$, and $(S^{(k)}y_i^{(k)})_j$ follows the distribution $N(0,\sum_{t=1}^m (y_i^{(k)})_t^2) = N(0,\|y_i^{(k)}\|_2^2) = \|y_i^{(k)}\|_2 \cdot N(0,1)$. Hence $w_i^{(k)}$ follows the distribution of $\frac{1}{s}\|y_i^{(k)}\|_2^2 \cdot \chi^2(s)$ where $\chi^2(s)$ is $\chi^2$-distribution with $s$ degree of freedom.

Hence if we only consider the randomness of matrix $S^{(k)}$, then we have

$$\mathop{\mathbf{E}}_{S}\left[\log\frac{\widehat{w}_i^{(k)}}{w_i^{(k)}}\right] = \mathop{\mathbf{E}}_{z\sim\chi^2(s)}\left[\log\frac{\|y_i^{(k)}\|_2^2}{\frac{1}{s}\|y_i^{(k)}\|_2^2\cdot z}\right] = \mathop{\mathbf{E}}_{z\sim\chi^2(s)}[\log\frac{s}{z}].$$

Hence by the probability density function of the $\chi^2$-distribution, assuming $s$ is even, we have that

$$\mathop{\mathbf{E}}_{z\sim\chi^2(s)}[\log z] = \int_0^\infty \frac{1}{2^{s/2}\Gamma(s/2)}x^{s/2-1}e^{-x/2}\log x\mathrm{d}x = \sum_{i=1}^{s/2-1}\frac{1}{i} - \gamma + \log 2.$$

The last equation use 4.352-2 of the 5th edition of [62], and $\gamma$ is the Euler constant. Hence

$$\mathop{\mathbf{E}}_{z\sim\chi^2(s)}[\log\frac{s}{z}] = \log s - \left(\sum_{i=1}^{s/2-1}\frac{1}{i} - \gamma + \log 2\right) \leqslant \log s - (\log\frac{s}{2} + \gamma - \gamma + \log 2 - \frac{2}{s}) = \frac{2}{s}.$$

by the fact that $\sum_{i=1}^k \frac{1}{i} \geqslant \log k + \gamma$.

Therefore we have that

$$\mathbf{E}[\phi_i(w)] \leqslant \frac{1}{T}\log\frac{m}{n} + \frac{2}{s}.$$

□

Since $\phi_i(w) = \log\sigma_i(w)$, Lemma 7.5.5 already provides some concentration results on $\sigma_i(w)$. We can also prove stronger type of concentration by bounding the moments of $\sigma_i(w)$ directly.

**Lemma 7.5.6** (Moments of $\sigma_i$)**.** *For $\alpha > 0$, if $\frac{s}{2} > \frac{\alpha}{T}$, then*

$$\mathbf{E}[\sigma_i(w)^\alpha] = \mathbf{E}\left[(a_i^\top(\sum_j w_j a_j a_j^\top)^{-1}a_i)^\alpha\right] \leqslant (\frac{m}{n})^{\frac{\alpha}{T}}\cdot(1+\frac{2\alpha}{sT-2\alpha})^T.$$

*Proof.* By Lemma 7.5.4 we have

$$\sigma_i(w)^\alpha \leqslant (\frac{m}{n})^{\frac{\alpha}{T}} \cdot \prod_{k=1}^{T} (\frac{\widehat{w}^{(k)}}{w^{(k)}})^{\frac{\alpha}{T}}.$$

Fix $k$ and $\widehat{w}^{(k)}$. Similar to the proof of Lemma 7.5.5, for the moment let us only consider the randomness of $S^{(k)}$, then we have

$$\mathop{\mathbf{E}}_{S^{(k)}} \left[ (\frac{\widehat{w}^{(k)}}{w^{(k)}})^{\frac{\alpha}{T}} \right] = \mathop{\mathbf{E}}_{z \sim \chi^2(s)} \left[ (\frac{s}{z})^{\frac{\alpha}{T}} \right].$$

Hence we have

$$\begin{aligned}
\mathop{\mathbf{E}}_{S^{(k)}} \left[ (\frac{\widehat{w}^{(k)}}{w^{(k)}})^{\frac{\alpha}{T}} \right] &= \int_0^\infty (\frac{s}{x})^{\frac{\alpha}{T}} \cdot \frac{1}{2^{s/2}\Gamma(s/2)} x^{s/2-1} e^{-x/2} \mathrm{d}x \\
&= \frac{s^{\frac{\alpha}{T}}}{2^{s/2}\Gamma(s/2)} \int_0^\infty x^{\frac{s}{2}-\frac{\alpha}{T}-1} e^{-x/2} \mathrm{d}x \\
&= \frac{s^{\frac{\alpha}{T}}}{2^{s/2}\Gamma(s/2)} \cdot \frac{\Gamma(s/2-\alpha/T)}{(1/2)^{s/2-\alpha/T}} \\
&= (s/2)^{\frac{\alpha}{T}} \cdot \frac{\Gamma(s/2-\alpha/T)}{s/2}.
\end{aligned}$$

where the third line uses 3.381-4 in [62] and the condition that $\frac{s}{2} > \frac{\alpha}{T}$.

By Lemma 7.2.2,

$$\frac{\Gamma(\frac{s}{2})}{\Gamma(\frac{s}{2}-\frac{\alpha}{T})} \geqslant \frac{(\frac{s}{2}-\frac{\alpha}{T})}{(\frac{s}{2})^{1-\frac{\alpha}{T}}},$$

which gives us

$$\mathop{\mathbf{E}}_{S^{(k)}} \left[ (\frac{\widehat{w}^{(k)}}{w^{(k)}})^{\frac{\alpha}{T}} \right] \leqslant (\frac{s}{2})^{\frac{\alpha}{T}} \cdot \frac{(\frac{s}{2})^{1-\frac{\alpha}{T}}}{(\frac{s}{2}-\frac{\alpha}{T})} = \frac{\frac{s}{2}}{\frac{s}{2}-\frac{\alpha}{T}}.$$

Because each $S^{(k)}$ matrix is independent to each other, we have

$$\mathbf{E}[\sigma_i(w)^\alpha] \leqslant (\frac{m}{n})^{\frac{\alpha}{T}} \cdot (\frac{\frac{s}{2}}{\frac{s}{2}-\frac{\alpha}{T}})^T = (\frac{m}{n})^{\frac{\alpha}{T}} \cdot (1 + \frac{2\alpha}{sT-2\alpha})^T.$$

□

Now we are ready to prove Theorem 7.5.2.

*Proof of Theorem 7.5.2.* Set $\alpha = \frac{3}{\varepsilon}\log\frac{m}{\delta}$. We can verify that

$$\alpha \geqslant \frac{\log(m/\delta)}{\log\frac{1+\varepsilon}{1+\varepsilon/4}}. \tag{7.6}$$

Notice that in this setting, we have $sT \geqslant 4\alpha$. Therefore, for $i \in [m]$, by Markov's inequality, we have

$$\begin{aligned}
\mathbf{Pr}[\sigma_i(w) \geqslant 1+\varepsilon] &= \mathbf{Pr}[\sigma_i(w)^\alpha \geqslant (1+\varepsilon)^\alpha] \\
&\leqslant \frac{\mathbf{E}[\sigma_i(w)^\alpha]}{(1+\varepsilon)^\alpha} \\
&\leqslant \frac{(\frac{m}{n})^{\frac{\alpha}{T}} \cdot (1+\frac{2\alpha}{sT-2\alpha})^T}{(1+\varepsilon)^\alpha} && \text{by Lemma 7.5.6} \\
&\leqslant \frac{(\frac{m}{n})^{\frac{\alpha}{T}} \cdot (1+\frac{2\alpha}{sT/2})^T}{(1+\varepsilon)^\alpha} && \text{because } sT \geqslant 4\alpha \\
&\leqslant \frac{(\frac{m}{n})^{\frac{\alpha}{T}} e^{\frac{4\alpha}{s}}}{(1+\varepsilon)^\alpha} && \text{Here we use } 1+x \leqslant e^x
\end{aligned}$$

With our choice of $s, T$, we can check that for sufficiently large $m, n$,

$$(\frac{m}{n})^{\frac{1}{T}} = (\frac{m}{n})^{\frac{\varepsilon/10}{\log(m/\delta)}} \leqslant 1+\varepsilon/10,$$

and

$$e^{\frac{4}{s}} = e^{\frac{\varepsilon}{20}} \leqslant 1+\varepsilon/10.$$

Hence

$$\mathbf{Pr}[\sigma_i(w) \geqslant 1+\varepsilon] \leqslant \left(\frac{(1+\varepsilon/10)^2}{1+\varepsilon}\right)^\alpha \leqslant \left(\frac{1+\varepsilon/4}{1+\varepsilon}\right)^\alpha.$$

Then by (7.6), we have

$$\mathbf{Pr}[\sigma_i(w) \geqslant 1+\varepsilon] \leqslant \frac{\delta}{m}.$$

By a union bound, we have that

$$\mathbf{Pr}[\exists i \in [m], \sigma_i(w) \geqslant 1+\varepsilon] \leqslant \delta.$$

Then let us prove the second part in Theorem 7.5.2. Fix $k$. Recall that $B^{(k)} = \sqrt{W^{(k)}}A$. Let $D^{(k)} = B^{(k)}((B^{(k)})^\top B^{(k)})^{-1}(B^{(k)})^\top$, then we can check that $D^{(k)}$ is an orthogonal projection matrix, because

$$(D^{(k)})^2 = \left(B^{(k)}((B^{(k)})^\top B^{(k)})^{-1}(B^{(k)})^\top\right) \cdot \left(B^{(k)}((B^{(k)})^\top B^{(k)})^{-1}(B^{(k)})^\top\right) = D^{(k)}.$$

Since $\mathsf{rank}(D^{(k)}) = n$, we can diagonalize $D^{(k)}$ as $D^{(k)} = \Lambda^{-1}E_n\Lambda$ where $\Lambda$ is an $m \times m$ orthogonal matrix, and $E_n \in \mathbb{R}^{m\times m}$ is a diagonal matrix where the first $n$ diagonal entries are 1 and all the other entries are 0. So we can rewrite the update rule as

$$\begin{aligned} w_i^{(k+1)} &= \frac{1}{s} \cdot \|S^{(k)}B^{(k)}((B^{(k)})^\top B^{(k)})^{-1}(\sqrt{w_i^{(k)}}a_i)\|_2^2 \\ &= \frac{1}{s} \cdot \left((S^{(k)}D^{(k)})^\top (S^{(k)}D^{(k)})\right)_{ii}. \end{aligned}$$

Therefore

$$\begin{aligned} \sum_{i=1}^m w_i^{(k+1)} =& \frac{1}{s} \cdot \sum_{i=1}^m \left((S^{(k)}D^{(k)})^\top (S^{(k)}D^{(k)})\right)_{ii} \\ =& \frac{1}{s} \cdot \mathsf{Tr}\left[((S^{(k)}D^{(k)})^\top (S^{(k)}D^{(k)})\right] \\ =& \frac{1}{s} \cdot \mathsf{Tr}\left[(D^{(k)})^\top (S^{(k)})^\top S^{(k)}D^{(k)}\right] \\ =& \frac{1}{s} \cdot \mathsf{Tr}\left[(S^{(k)})^\top S^{(k)}(D^{(k)})^2\right] && D^{(k)} \text{ is symmetric} \\ =& \frac{1}{s} \cdot \mathsf{Tr}\left[(S^{(k)})^\top S^{(k)}D^{(k)}\right] && (D^{(k)})^2 = D^{(k)} \\ =& \frac{1}{s} \cdot \mathsf{Tr}\left[S^{(k)}\Lambda^{-1}E_n\Lambda(S^{(k)})^\top\right]. && \text{diagonalization of } D^{(k)} \end{aligned}$$

Let $\widehat{S}^{(k)} = S^{(k)}\Lambda^{-1}$. Here $\Lambda$ is a function of $A, S^{(1)}, \cdots, S^{(k-1)}$, but $\Lambda$ does not depend on $S^{(k)}$. Notice that the Gaussian distribution is invariant under orthogonal transform, and $S^{(k)}$ is independent to previous $S$ matrices. We conclude that $\widehat{S}^{(k)}$ also has i.i.d entries that follows the distribution $N(0,1)$, and $\widehat{S}^{(k)}$ is independent to previous randomness.

So we have

$$\sum_{i=1}^m w_i^{(k+1)} = \frac{1}{s} \cdot \mathsf{Tr}[\widehat{S}^{(k)}E_n(\widehat{S}^{(k)})^\top] = \frac{1}{s}\sum_{p=1}^s\sum_{q=1}^n(\widehat{S}^{(k)})_{pq}^2.$$

Namely, the distribution of $\sum_{i=1}^{m} w_i^{(k+1)}$ is $\frac{1}{s}\chi^2(ns)$. Because for different $k$, the random choices are independent, we have $\sum_{i=1}^{m} w_i = \frac{1}{T}(\sum_{k=1}^{T}\sum_{i=1}^{m} w_i^{(k)})$ follows the distribution $\frac{1}{sT}\chi^2(nsT)$. So we can set $t = \frac{1}{16}\varepsilon^2 \cdot nsT = \Theta(n \log \frac{m}{\delta})$ in Lemma 7.2.4 to see that for sufficiently large $m, n$,

$$\begin{aligned}
\mathbf{Pr}\left[\sum_{i=1}^{m} w_i^{(k+1)} \geqslant (1+\varepsilon)n\right] =& \mathbf{Pr}_{z\sim\frac{1}{sT}\chi^2(nsT)}\left[z \geqslant (1+\varepsilon)n\right] && \text{set } z = \sum_{i=1}^{m} w_i^{(k+1)} \\
=& \mathbf{Pr}_{z\sim\chi^2(nsT)}\left[z \geqslant (1+\varepsilon)nsT\right] && \text{rescale } z \\
=& \mathbf{Pr}_{z\sim\chi^2(nsT)}\left[z - nsT \geqslant \varepsilon \cdot nsT\right] \\
\leqslant& \mathbf{Pr}_{z\sim\chi^2(nsT)}\left[z - nsT \geqslant 2t + 2\sqrt{nsT \cdot t}\right] \\
\leqslant& e^{-t} \leqslant \delta. && \text{by Lemma 7.2.4}
\end{aligned}$$

□

### 7.5.2 *Runtime analysis*

We now analyze the running time of Algorithm 5. The main loop is executed $T$ times, and inside each loop, we can first use $O(s \cdot \mathsf{nnz}(B^{(k)}))$ time to compute $S^{(k)}B^{(k)}$, then solve $s$ linear systems to compute $F^{(k)} := S^{(k)}B^{(k)}((B^{(k)})^\top B^{(k)})^{-1}$. Finally we can compute all of $\|F^{(k)} \cdot (w_i^k)^{1/2} \cdot a_i\|_2^2$ in $O(s \cdot \mathsf{nnz}(A))$ time by first computing matrix $S^{(k)} \cdot \sqrt{W^{(k)}} \cdot A^\top$. Recall that $B^{(k)} = \sqrt{W^{(k)}}A$, so $\mathsf{nnz}(B^{(k)}) \leqslant \mathsf{nnz}(A)$. Notice that it takes at least $\mathsf{nnz}(A)$ time to solve the linear system $A^\top W^{(x)} Ax = b$. Together with Theorem 7.5.3, this gives us

**Theorem 7.5.7.** *For all $\varepsilon, \delta \in (0,1)$, we can find a $(1+\varepsilon)$-approximation of the John Ellipsoid inside a symmetric polytope within $O\left(\frac{1}{\varepsilon}\log\frac{m}{\delta}\right)$ iterations with probability at least $1-\delta$. Moreover, each iteration involves solving $O(\frac{1}{\varepsilon})$ linear systems of the form $A^\top WAx = b$ for some diagonal matrix $W$.*

| | FixedPoint | | Todd | | ABPGLS | | Zhang | |
|---|---|---|---|---|---|---|---|---|
| | Its | Time(s) | Its | Time(s) | Its | Time(s) | Its | Time(s) |
| Scaled Boxes | 3 | 0.0069 | 1 | 0.028 | 2 | 4.2 | 3 | 0.045 |
| random Gaussian | 6 | 0.022 | 4.7e+02 | 0.12 | 5 | 1.5 | 4 | 0.99 |
| Tangent to ellipsoid | 3 | 0.011 | 5.8e+02 | 0.15 | 2 | 2 | 2 | 0.36 |
| Scaled balls | 4 | 0.025 | 5.9e+02 | 0.2 | 9 | 18 | 4 | 4.7 |
| Two ellipsoids | 15 | 0.088 | 5.8e+02 | 0.19 | 30 | 31 | 6 | 7.8 |

Table 7.1: Number of iterations and total time for each algorithm on the set of examples. Each tested polytope is in dimension 1000 and each algorithm was run until $\varepsilon < 0.10$.

| | FixedPoint | | Todd | | ABPGLS | | Zhang | |
|---|---|---|---|---|---|---|---|---|
| | Its | Time(s) | Its | Time(s) | Its | Time(s) | Its | Time(s) |
| Scaled Boxes | 3 | 0.0053 | 1 | 3.8 | Inf[a][b] | Inf | 3 | 3.1 |
| Random Gaussian | 1 | 0.054 | 5e+02 | 8.1 | 2 | 9.9 | 1 | 0.68 |
| Tangent to ellipsoid | 1 | 0.26 | 8e+03 | 94 | 2 | 9.4e+02 | 1 | 17 |
| Scaled balls | 3 | 0.27 | 7.5e+02 | 11 | Inf | Inf | 3 | 12 |
| Two ellipsoids | 10 | 6.4 | 7.8e+03 | 1.2e+02 | Inf | Inf | 3 | 8.3e+02 |

[a] If an entry has "Inf", this means the program did not terminate with 30 minutes of computational time.

[b] We run all the algorithms as provided, hence we suspect their algorithms can be better by parameter tuning and/or preprocessing.

### *7.6 Experimental Performance*

In this section, we do a brief comparison of our algorithm to existing algorithms/-tools for the maximum volume ellipsoid problem (or equivalently, algorithms for optimal D-design). The experimental results show that our algorithm performs favorably in the varied instances to the three tested packages, which we denote as "Todd" [141],

Table 7.2: Number of iterations and total time for each algorithm on the set of examples. Each tested polytope is in dimension 1000 and each algorithm was run until $\varepsilon < 0.001$.

| | FixedPoint | | Todd | | ABPGLS | | Zhang | |
|---|---|---|---|---|---|---|---|---|
| | Its | Time(s) | Its | Time(s) | Its | Time(s) | Its | Time(s) |
| Scaled Boxes | 6 | 0.0045 | 1 | 3.2 | Inf | Inf | 4 | 4.3 |
| Random Gaussian | 5 | 0.25 | 2e+03 | 19 | 6 | 36 | 3 | 3.2 |
| Tangent to ellipsoid | 34 | 9.4 | 1.9e+04 | 2.2e+02 | Inf | Inf | 5 | 2.7e+02 |
| Scaled balls | 1.2e+02 | 9.7 | 3.8e+03 | 37 | Inf | Inf | 7 | 29 |
| Two ellipsoids | 2.3e+02 | 1.3e+02 | 3.2e+04 | 4.7e+02 | Inf | Inf | Inf | Inf |

Table 7.3: Number of iterations and total time for each algorithm on various netlib examples. If a row has multiple instances, the reported iteration counts and times are reported as mean/median of the instances. maros_r7 has $m = 9408, n = 3136$, osa_07 has $m = 25067, n = 1118$, qap12 has $m = 8856, n = 2794$. Each algorithm was run until $\varepsilon < 0.10$.

| | FixedPoint | | Todd | | ABPGLS | | Zhang | |
|---|---|---|---|---|---|---|---|---|
| | Its | Time(s) | Its | Time(s) | Its | Time(s) | Its | Time(s) |
| Netlib small | 7.1/6.5 | 0.12/0.02 | 230/58 | 4.8/0.31 | 29/29 | 57/8.2 | 3.9/3.5 | 9.1/0.43 |
| maros_r7 | 3 | 11.1 | 1 | 137 | Inf | Inf | 4 | 197 |
| osa_07 | 12 | 2.6 | 2404 | 67 | Inf | Inf | Inf | Inf |
| qap12 | 8 | 38.5 | 7751 | 570 | 7 | 610 | 4 | 145 |

“APBGLS" [65], and “Zhang" [155]. We denote our algorithm as “FixedPoint", and it is available for reference in Appendix B. The theoretical gains of the algorithm appear to transfer to an improved practical algorithm. We test 7 types of instances:

1. **Scaled Boxes**: Two hypercubes of different side lengths. We tested $[-5, 5]^n \cap$

$[-10, 10]^n$.

2. **Random Gaussian**: $A \in \mathbb{R}^{m \times n}$, and each entry of $A$ is an iid standard Gaussian. We set $m = 3n/2$.

3. **Tangent to ellipsoid**: $A \in \mathbb{R}^{m \times n}$, and each facet of $A$ is tangent to a random ellipsoid (defined by an $n \times n$ Gaussian matrix). We set $m = 10n$.

4. **Scaled balls**: $A \in \mathbb{R}^{m \times n}$. Balls of radius $r_1, r_2$ each having $m/2$ facets tangent to the ball, where each facet is chosen according to the uniform measure from the respective sphere. We set $m = 3n$, $r_1 = \sqrt{n}/2$, and $r_2 = \sqrt{n}/2 + 10$.

5. **Two ellipsoids**: $A$ consists of two independent realizations of the class **Tangent to ellipsoid**. We set $m = 20n$.

6. **Netlib small**: As a form of "real-world" and sparse examples, we consider a subset of Netlib polytopes. In this class, we consider those instances that (i) have a full-dimensional dual that only contains inequality constraints and (ii) have $m + n < 10^4$. We symmetrize the polytope with respect to the analytic center in the dual space. $39$[2] of the 109 models in the dataset satisfy these conditions.

7. **Netlib large**: We consider those Netlib polytopes that satisfy the same conditions as **Netlib small**, but have $m + n \in [10^4, 5 \cdot 10^4]$. 5 Netlib polytopes satisfy these conditions. [3]

---

[2]However, the reported results in Table 7.3 are only on 35 of these 39 models. For two of the instances, our algorithm fails due to numerical issues in the Cholesky factorization; this could likely be fixed by padding the matrix $B' \cdot B$ with a small multiple of the identity matrix, but we ignored this issue for cleanliness of the code. Two other instances take too long for the other implementations. Thus, all 4 sets of results for **Netlib small** were run on the same 35 models.

[3]However, similar to the **Netlib small** case, two of the cases fail our method due to numerical issues with the Cholesky factorization.

# Chapter 8

# EXPERIMENTAL DESIGN WITH CONSTRAINTS

## *8.1 Introduction*

In this chapter, we study how extra knowledge can help with the linear bandits problem. Consider the *best arm identification* problem, where we have a $d$ dimensional vector $\theta$, and we want to know which coordinate is the largest. We are allowed to query one coordinate at one time. If $\theta$ can be arbitrary, then an information theory lower bound suggests that we need to at least make $d$ queries. However, the sample complexity can drop drastically if prior constraints on $\theta$ are given. For example, if we know $\theta$ is the evaluation of some one-dimensional convex function over $d$ equally distributed values, then we can use binary search to reduce the sample complexity to $O(\log d)$ queries.

We define the problem setup in Section 8.2, and review an existing algorithm on the unconstrained problem in Section 8.3. In Section 8.4, we present our main contribution, a novel algorithm that takes advantage of the constraint set. We also analyze the regret and the sample complexity of this algorithm. Our result heavily depends on the geometry of the constraint set, hence in Section 8.5, we examine the data-specific parameters for some common convex sets.

## *8.2 Problem Setup and Background*

We study the *pure exploration for linear bandit problem*, which is defined as follows. Let $\mathcal{X} := (x_1, \ldots, x_n) \subset \mathbb{R}^d$ be a finite collection of $n$ vectors in $\mathbb{R}^d$. Let $K \subset \mathbb{R}^d$ be a convex set, and $\theta_* \in K$. Consider the following game between a player and a stochastic "nature" where $\mathcal{X}, K$ are known to the player, but $\theta_*$ is unknown to the player. At each time $t \in \mathbb{N}$, the player selects an $x_t \in \mathcal{X}$, and nature then reveals $y_t = \langle x_t, \theta_* \rangle + \eta_t$ where $\eta_t \sim \mathcal{N}(0, 1)$.

At the end of each round the player optionally decides to terminate and outputs some $\widehat{x} \in \mathcal{X}$.

**Definition 8.2.1.** *For any $\delta \in (0,1)$, we say an algorithm (player) is $\delta$-PAC for $(\mathcal{X}, K)$ if for any $\theta_* = \theta \in K$ we have that $\mathbf{P}(\widehat{x} = \arg\max_{x \in \mathcal{X}} \langle x, \theta \rangle) \geqslant 1 - \delta$.*

The sample complexity of the algorithm is the number of samples that the algorithm has seen when it terminates. The best arm identification problem seeks an algorithm that is $\delta$-PAC and terminates as soon as possible.

As an online learning problem, we can also talk about the *regret* of the player. Let $x_* = \arg\max_{x \in \mathcal{X}} \langle x, \theta_* \rangle$. Assume that the algorithm runs for $T$ rounds and selects $x_1, \cdots, x_T$. Then the regret of the algorithm is measured as $\sum_{k=1}^{T} \langle x_* - x_k, \theta_* \rangle$.

When $K = \mathbb{R}^d$, the linear bandits problem has been extensively studied and nearly optimal algorithms are known [125, 138, 154, 54, 43, 140]. Soare et al. [125] first related the linear bandits problem to G-optimal design, and obtained a complexity upper bound of $O(d^2 + \frac{d}{\Delta_{\min}^2} \log n)$, where $\Delta_{\min} = \min_{x' \in \mathcal{X}, x' \neq x_*} \langle x_* - x', \theta_* \rangle$. The $d^2$ term in the upper bound was later improved by Tao et al. [138], and in a recent work of Fiez et al. [54], a sample complexity that is up to logarithm factors of the information theoretical lower bound was obtained. Soare [124] gave the first analysis on the lower bound of sample complexity, and Degenne et al. [43] developed an algorithm whose sample complexity in the asymptotic regime matches the constant of the lower bound.

As for the regret, based on the *Upper Confidence Bound* (UCB) method, an upper bound of $O(d\sqrt{T} \log T)$ regret is known [90]. Using G-optimal design with elimination, it is also known to achieve a regret bound of $O(\sqrt{dT \log n})$. The first bound is independent on $n = |\mathcal{X}|$, while the second bound is better when $n < 2^d$.

The linear bandits problem relies on properties of unconstrained least squares, or linear regression. A related, comparatively harder problem is the *shape restricted regression*, where the learner wants to recover an unknown function from noisy observations. This problem becomes non-trivial when the learner knows some structural information on the

unknown function beforehand. In our setting, this is equivalent to the constraint set $K$ being the evaluations of all functions with desired property, $\theta_*$ being the evaluation of the unknown function, and $\mathcal{X}$ being the canonical basis. This problem has been studied for unimodal functions [26], isotonic functions and convex functions [14].

However, to the best of our knowledge very little is known in the general constrained case, that is, an arbitrary convex set $K$. Our result is the first to involve the geometry of the constraints in its analysis. Our main contribution is that we design a novel algorithm that takes advantage of the geometry of $K$, and we analyze the regret as well as the sample complexity of our algorithm.

### *8.3 Unconstrained Case*

In this section, we review an almost optimal algorithm in the unconstrained case $K = \mathbb{R}^d$ based on optimal experimental design proposed by Fiex, Jain, Jamieson and Ratlif [54], which greatly inspires this work. Here we present the high level idea of their algorithm, Randomized Adaptive Gap Elimination (RAGE). Assume that we have $T$ samples $(x_1, y_1), \cdots, (x_T, y_T)$. We know that $y = X\theta_* + \eta$, where $y = (y_1, \cdots, y_T) \in \mathbb{R}^T$, $\eta = (\eta_1, \cdots, \eta_T) \sim \mathcal{N}(0, I_T)$, and $X = \begin{pmatrix} x_1^\top \\ x_2^\top \\ \vdots \\ x_T^\top \end{pmatrix} \in \mathbb{R}^{T \times d}$. The maximum likelihood estimator $\widehat{\theta}$ is then computed as

$$\widehat{\theta} = (X^\top X)^{-1} X^\top y = \theta_* + (X^\top X)^{-1} X^\top \eta.$$

Hence, $\widehat{\theta}$ is distributed as $\mathcal{N}(\theta_*, (X^\top X)^{-1})$.

The strategy of RAGE is to use $\widehat{\theta}$ to eliminate $x' \in \mathcal{X}$ as candidates for $x_*$. Indeed, if we find some other $z \in \mathcal{X}$ so that $\langle x, \widehat{\theta} \rangle > \langle x', \widehat{\theta} \rangle$, then $x'$ is probably not the optimal $x_*$ if $\widehat{\theta}$ well approximates $\theta_*$. How confident can we be of this? We have the following

computation:

$$\langle x, \theta_* \rangle - \langle x', \theta_* \rangle = \langle x, \widehat{\theta} \rangle - \langle x', \widehat{\theta} \rangle + \langle x - x', \theta_* - \widehat{\theta} \rangle,$$

and

$$\begin{aligned} \langle x - x', \theta_* - \widehat{\theta} \rangle &\geqslant - \|x - x'\|_{(X^\top X)^{-1}} \cdot \|\theta_* - \widehat{\theta}\|_{(X^\top X)} \\ &= - \|x - x'\|_{(X^\top X)^{-1}} \cdot \|\xi\|_2. \end{aligned}$$

Here $\xi \in \mathbb{R}^d$, and the last step uses the fact when $\xi \sim \mathcal{N}(0, I_d)$ and $\eta \sim \mathcal{N}(0, I_T)$, $(X^\top X)^{-\frac{1}{2}}\xi$ and $(X^\top X)^{-1}X^\top \eta$ are identically distributed, because they are both distributed as $\mathcal{N}(0, (X^\top X)^{-1})$.

Hence when $\langle x, \widehat{\theta} \rangle - \langle x', \widehat{\theta} \rangle$ is bigger than $\|x - x'\|_{(X^\top X)^{-1}} \cdot \|\xi\|_2$, we can safely eliminate $x'$ from being candidate for $x_*$. Since $\xi$ is a Gaussian vector, $\|\xi\|_2$ has good concentration. Therefore we just need to find the optimal $T$ samples so that $\|x - x'\|_{(X^\top X)^{-1}}$ is as small as possible. To avoid a discrete optimization problem, we consider the continuous version of it, which is

$$\min_{\lambda \in \Delta_{\mathcal{X}}} \max_{x', x \in \mathcal{X}} \|x' - x\|^2_{(\sum_i \lambda_i x_i x_i^\top)^{-1}}.$$

Here $\Delta_{\mathcal{X}}$ is the set of all probability distributions over $\mathcal{X}$. This is a convex program, which can be efficiently solved in polynomial time. Now it is clear how to get these samples: we first compute the good distribution $\lambda_t$ over $\mathcal{X}$, then at step $t$ we just choose $x_t$ according to $\lambda_t$.

One remaining issue is that we do not know the true value of $\langle x, \theta_* \rangle - \langle x', \theta_* \rangle$, hence it may be difficult to determine the number of samples so that $\|x - x'\|_{(X^\top X)^{-1}}$ is small enough. Nevertheless, the RAGE algorithm proceeds in rounds by setting a threshold $\varepsilon_t$ in round $t$, and selecting enough many samples so that all $x'$ with $\langle x_*, \theta_* \rangle - \langle x', \theta_* \rangle > \varepsilon_t$ are eliminated before round $t$. By halving $\varepsilon_t$ each time, RAGE guarantees that at the end it can report the optimal $x_*$.

### 8.4 A novel algorithm for constrained experimental design

We now focus on the case where $K \subsetneq \mathbb{R}^d$ and $\mathcal{X} = \mathcal{X}$. Again assume that we have $T$ samples $(x_1, y_1), \cdots, (x_T, y_T)$, and we still have $y = X\theta_* + \eta$. However, different from the previous analysis, the least squares solution $\widehat{\theta} = (X^\top X)^{-1} X^\top y$ may not be feasible since $\widehat{\theta}$ may not lie inside $K$. Nevertheless, we can still consider the maximum likelihood estimator, which now becomes

$$\bar{\theta} := \arg\min_{\theta \in K} \sum_{i=1}^{T} (y_i - \langle x_i, \theta \rangle)^2 = \arg\min_{\theta \in K} \|\theta - \widehat{\theta}\|_{(X^\top X)}.$$

In the above analysis, we used the fact that

$$|\langle x - x', \theta_* - \widehat{\theta} \rangle| \leqslant \|x - x'\|_{(X^\top X)^{-1}} \cdot \|\theta_* - \widehat{\theta}\|_{(X^\top X)}.$$

In the unconstrained case, $\|\theta_* - \widehat{\theta}\|_{(X^\top X)}$ was easy to analyze, namely $\|\theta_* - \widehat{\theta}\|_{(X^\top X)} = \|\xi\|_2$. In the constrained case, we expect $\|\theta_* - \widehat{\theta}\|_{(X^\top X)}$ to be much smaller since it takes advantage of $K$. This intuition is proved by Chatterjee [27] and Bellec [14], who showed that the geometry property of $K$ plays an important role in the concentration. For our purpose, we show the following result built on top of [14].

**Lemma 8.4.1.** *Let $K$ be a convex set in $\mathbb{R}^d$. Assume that $y = X\theta_* + \eta$ where $y \in \mathbb{R}^T$, $X \in \mathbb{R}^{T \times d}$, $\theta_* \in K$ and $\eta \sim \mathcal{N}(0, I_T)$. Let $\widehat{\theta} = (X^\top X)^{-1} X^\top y$ and $\bar{\theta} = \arg\min_{\theta \in K} \|\theta - \widehat{\theta}\|_{(X^\top X)}$.*

*Assume that there exists $t_*(K, X^\top X) > 0$ that satisfies*

$$\mathop{\mathbf{E}}_{\xi \sim \mathcal{N}(0, I_d)} \left[ \sup_{\theta, \theta' \in K : \|\theta - \theta'\|_{X^\top X} \leqslant t_*(K, X^\top X)} \langle (X^\top X)^{1/2} (\theta - \theta'), \xi \rangle \right] \leqslant t_*(K, X^\top X)^2 / 2.$$

*Then for any $v > 0$, with probability at least $1 - e^{-v}$,*

$$\|\bar{\theta} - \theta_*\|_{X^\top X}^2 \leqslant (t_*(K, X^\top X) + \sqrt{2v})^2$$

*Proof.* The proof resembles the proof of Theorem 2.3 in [14]. For simplicity, let $t = t_*(K, X^\top X)$. Define the random variable $Z$ as $Z = \sup_{\theta, \theta' \in K : \|\theta - \theta'\|_{X^\top X} \leqslant t} \langle (X^\top X)^{1/2} (\theta - \theta'), \xi \rangle$. Since for any valid $\theta, \theta'$ we have

$$\mathbf{E}[\langle (X^\top X)^{1/2}(\theta - \theta'), \xi \rangle^2] = \|\theta - \theta'\|^2_{X^\top X} \leqslant t^2,$$

by Theorem 5.8 of [19], we have for any $u > 0$, with probability at least $1 - e^{-\frac{u^2}{2t^2}}$,

$$Z \leqslant \mathbf{E}[Z] + u.$$

Plugging in $u = t\sqrt{2v}$, we have with probability at least $1 - v$,

$$Z \leqslant \mathbf{E}[Z] + t\sqrt{2v} \leqslant t^2/2 + t\sqrt{2v},$$

where the last step follows from the assumption on $t$. Denote this event by $\Omega$.

Since $\bar{\theta} = \arg\min_{\theta \in K} \|\theta - \widehat{\theta}\|_{(X^\top X)}$, and $K$ is convex, we have

$$\|\bar{\theta} - \widehat{\theta}\|^2_{(X^\top X)} \leqslant \|\theta_* - \widehat{\theta}\|^2_{(X^\top X)} - \|\theta_* - \bar{\theta}\|^2_{(X^\top X)}.$$

As a consequence, we have

$$\begin{aligned}
\|\theta_* - \bar{\theta}\|^2_{(X^\top X)} &\leqslant \|\theta_* - \widehat{\theta}\|^2_{(X^\top X)} - \|\bar{\theta} - \widehat{\theta}\|^2_{(X^\top X)} \\
&= 2(\theta_* - \bar{\theta})(X^\top X)(\theta_* - \widehat{\theta}) - \|\theta_* - \bar{\theta}\|^2_{(X^\top X)}.
\end{aligned}$$

We argue that when $\Omega$ happens, $\|\theta_* - \bar{\theta}\|_{(X^\top X)} \leqslant t + \sqrt{2v}$. If $\|\theta_* - \bar{\theta}\|_{(X^\top X)} \leqslant t$, then this is clearly true. Otherwise, let $\theta' = \frac{t}{\|\theta_* - \bar{\theta}\|_{(X^\top X)}}(\theta_* - \bar{\theta}) + \bar{\theta}$. We can verify that $\theta' \in K$, and $\|\theta' - \bar{\theta}\|_{(X^\top X)} = t$. Therefore

$$\begin{aligned}
\|\theta_* - \bar{\theta}\|^2_{(X^\top X)} &\leqslant 2(\theta_* - \bar{\theta})(X^\top X)(\theta_* - \widehat{\theta}) - \|\theta_* - \bar{\theta}\|^2_{(X^\top X)} \\
&= \frac{2\|\theta_* - \bar{\theta}\|_{(X^\top X)}}{t}(\theta' - \bar{\theta})X^\top \eta - \|\theta_* - \bar{\theta}\|^2_{(X^\top X)} \\
&\leqslant \frac{2\|\theta_* - \bar{\theta}\|_{(X^\top X)}}{t} Z - \|\theta_* - \bar{\theta}\|^2_{(X^\top X)} \\
&= (\frac{X}{t})^2 - (\frac{X}{t} - \|\theta_* - \bar{\theta}\|_{(X^\top X)})^2 \\
&\leqslant (\frac{X}{t})^2 \\
&\leqslant (t + \sqrt{2v})^2,
\end{aligned}$$

where the last step follows from the event $\Omega$. This completes the proof of Lemma 8.4.1. □

Lemma 8.4.1 provides an upper bound for the quantity $\|\bar{\theta} - \theta_*\|_{X^\top X}$. But the upper bound depends on the samples $X$, which is awkward. So we just take the worst case upper bound, which is defined as follows.

$$\gamma(K, \delta) := \max_{\|\lambda\|_1 = T; \forall i \lambda_i \geqslant 0} t_*(K, (\sum_i \lambda_i x_i x_i^\top)) + \sqrt{2\log(2/\delta)}.$$

Now it turns to bound the quantity $|\langle x - x', \theta_* - \bar{\theta}\rangle|$ as in the unconstrained case, because when this term is sufficiently small, we can safely eliminate $x'$ when there exists another $x$ so that $\langle x - x', \bar{\theta}\rangle$ is bigger than some threshold. By the triangle inequality, it is sufficient to have an upper bound on $\max_{x \in \mathcal{X}} |\langle x, \theta_* - \bar{\theta}\rangle|$. But $\theta_*$ is not known to the learner, which does not help us to choose the best $N$ samples. Therefore we use the condition that $\|\bar{\theta} - \theta_*\|_{X^\top X} \leqslant \gamma(K, \delta)$ to get the following relaxation:

$$\beta(N, \delta) := \min_{\lambda \in \Delta_{\mathcal{X}}} \max_{x \in \mathcal{X}} \sup_{\theta, \theta' \in K : \sqrt{N}\|\theta - \theta'\|_{(\sum_i \lambda_i x_i x_i^\top)} \leqslant \gamma(K, \delta)} \sqrt{N}\langle x, \theta - \theta'\rangle. \tag{8.1}$$

Now it is time to introduce our main algorithm, Algorithm 6. Like RAGE for the unconstrained case, Algorithm 6 also proceeds in rounds, and in round $t$ there is a threshold proportional to $\frac{\beta(N,\delta)}{\sqrt{N}}$. Algorithm 6 aims that in round $t$, all $x'$ with $\langle x_*, \theta_*\rangle - \langle x', \theta_*\rangle$ being large for the threshold in round $t$ are eliminated. To achieve this goal, Algorithm 6 estimates a number of samples $N$ so that $\frac{\beta(N,\delta)}{\sqrt{N}}$ is small enough, and use the best distribution on $\mathcal{X}$ that achieves $\beta(N, \delta)$ to select the $N$ samples. Then Algorithm 6 computes $\bar{\theta}_t$ which is the constrained least squares as the estimator in round $t$, and use $\bar{\theta}_t$ to update the set $\widehat{\mathcal{X}}_t$.

Notice that there is some discrepancy between continuous distribution to integral samples. To get rid of this issue, we adopt the rounding procedure from [3]. We summarize the guarantee we need in the following proposition. For more details, readers may refer to [3].

**Proposition 8.4.2.** *Let $\pi \in \mathbb{R}^n_{\geqslant 0}$ satisfy $\|\pi\|_1 = N$. Then there exists an integer vector $s$ so that $\|s\|_1 \leqslant N + 1620d$, and*

$$\begin{aligned}
&\max_{x\in\mathcal{X}} \sup_{\theta,\theta'\in K:\|\theta-\theta'\|_{(\sum_i s_i x_i x_i^\top)}\leqslant\gamma(K,\delta/k^2)} \langle x, \theta-\theta'\rangle \\
\leqslant &\frac{4}{3}\max_{x\in\mathcal{X}} \sup_{\theta,\theta'\in K:\|\theta-\theta'\|_{(\sum_i \pi_i x_i x_i^\top)}\leqslant\gamma(K,\delta/k^2)} \langle x, \theta-\theta'\rangle.
\end{aligned}$$

**Algorithm 6:** Constrained Experimental Design

Initialize $\widehat{\mathcal{X}}_1 = \mathcal{X}$.
**for** $k = 1, \cdots,$ **do**
  $N_k = 1$.
  **repeat**
    $N_k \leftarrow 4N_k$.
    Compute $\lambda_k$ from the convex program

$$\min_{\lambda\in\Delta_{\mathcal{X}}} \max_{x\in\mathcal{X}} \sup_{\theta,\theta'\in K:\sqrt{N_k}\|\theta-\theta'\|_{(\sum_i \lambda_i x_i x_i^\top)}\leqslant\gamma(K,\delta/k^2)} \sqrt{N_k}\langle x, \theta-\theta'\rangle. \tag{8.2}$$

    Compute $\beta_k$ as the objective value.
  **until** $\frac{\beta_k}{\sqrt{N_k}} \leqslant 2^{-k}$;
  Compute $\pi_k = N_k \cdot \lambda_k$. Compute $s_k \in \mathbb{N}^n$ be the rounding of $\pi_k$.
  `/* By Proposition 8.4.2,` $\|s_k\|_1 \leqslant N_k + 1620d$. `*/`
  Obtain $X_k \in \mathbb{R}^{\|s_k\|_1\times d}$ and $y_k \in \mathbb{R}^{\|s_k\|_1}$ by querying $x_i$ for $s_{k,i}$ times.
  Compute $\widehat{\theta}_k = (X_k^\top X_k)^{-1} X_k^\top y$ and $\bar{\theta}_k = \arg\min_{\theta\in K} \|\theta - \widehat{\theta}_k\|_{(X_k^\top X_k)}$.
  $\widehat{\mathcal{X}}_{k+1} \leftarrow \widehat{\mathcal{X}}_k \backslash \{x' \in \widehat{\mathcal{X}}_k : \exists x \in \widehat{\mathcal{X}}_k, \langle x - x', \bar{\theta}_k\rangle \geqslant \frac{6\beta_k}{\sqrt{N_k}}\}$.
  **if** $|\widehat{\mathcal{X}}_{k+1}| = 1$ **then**
    **return** $\widehat{\mathcal{X}}_{k+1}$

*Moreover, given $\pi$, $s$ can be computed in time polynomially in $N, d$.*

We now present the regret of Algorithm 6.

**Theorem 8.4.3.** *For sufficiently large number $T$, with probability at least $1-\delta$, the regret of Algorithm 6 after running for $T$ steps is at most $36\beta\sqrt{T}+14580d$, where $\beta$ is defined as $\beta = \beta_{\log T}$.*

*Proof.* Let $\Omega$ be the event that for all $k = 1, 2, \cdots$, $\|\bar{\theta}_k - \theta_*\|_{X_k^\top X_k} \leqslant \gamma(K, \delta/k^2)$. By Lemma 8.4.1, this occurs with probability at least $1 - \sum_{k=1}^{\infty} e^{-\log \frac{2k^2}{\delta}} \geqslant 1-\delta$. In the remaining of the proof we assume that $\Omega$ happens.

Let $\nu > 0$ be some parameter to be optimized later. At stage $k+1$, if we choose $x' \neq x_*$, we will incur a loss of $\langle x_* - x', \theta_* \rangle$. But with the construction of $\widehat{\mathcal{X}}_{k+1}$, it is guaranteed that $|\langle x_* - x', \bar{\theta}_k \rangle| \leqslant \frac{6\beta_k}{\sqrt{N_k}}$. We then argue that $\langle x_* - x', \theta_* \rangle \leqslant \frac{9\beta_k}{\sqrt{N_k}}$. Otherwise, if $\langle x_* - x', \theta_* \rangle > \frac{9\beta_k}{\sqrt{N_k}}$, then

$$
\begin{aligned}
\langle x_* - x', \bar{\theta}_k \rangle &= \langle x_* - x', \theta_* \rangle + \langle x_* - x', \bar{\theta}_k - \theta_* \rangle \\
&\geqslant \frac{9\beta_k}{\sqrt{N_k}} - 2 \Big| \sup_{x \in \widehat{\mathcal{X}}_k, \theta, \theta' \in K, \|\theta - \theta'\|_{X_k^\top X_k} \leqslant \gamma(K, \delta/k^2)} \langle x, \theta - \theta' \rangle \Big| \\
&\geqslant \frac{9\beta_k}{\sqrt{N_k}} - 2 \cdot \frac{4}{3} \Big| \sup_{x \in \widehat{\mathcal{X}}_k, \theta, \theta' \in K, \|\theta - \theta'\|_{N_k \cdot (\sum_i \lambda_{k,i} x_i x_i^\top)} \leqslant \gamma(K, \delta/k^2)} \langle x, \theta - \theta' \rangle \Big| \\
&= \frac{9\beta_k}{\sqrt{N_k}} - \frac{8}{3} \Big| \sup_{x \in \widehat{\mathcal{X}}_k, \theta, \theta' \in K, \sqrt{N_k}\|\theta - \theta'\|_{(\sum_i \lambda_{k,i} x_i x_i^\top)} \leqslant \gamma(K, \delta/k^2)} \langle x, \theta - \theta' \rangle \Big| \\
&\geqslant \frac{9\beta_k}{\sqrt{N_k}} - \frac{8}{3} \frac{\beta_k}{\sqrt{N_k}} \\
&> \frac{6\beta_k}{\sqrt{N_k}}.
\end{aligned}
$$

where the second step follows from the event $\Omega$, the third step follows from Proposition 8.4.2, and the fifth step follows from the definition of $\beta_k$.

Hence the loss cannot be more than $\frac{9\beta_k}{\sqrt{N_k}}$. Notice that when $k > \log \frac{9}{\nu}$, $\frac{9\beta_k}{\sqrt{N_k}} \leqslant 9 \cdot 2^{-k} < \nu$. Therefore if at one step the algorithm incurs more than $\nu$ loss, then it must happens before stage $\log \frac{9}{\nu}$.

We then argue that $\frac{\beta_k}{\sqrt{N_k}} > 2^{-k-1}$. This is because $\frac{\beta_k}{\sqrt{N_k/4}} \geqslant 2^{-k}$, otherwise we would have set $N_k \leftarrow N_k/2$.

Let $\beta = \max_{1 \leqslant k \leqslant \log \frac{9}{\nu}} \beta_k$, then the total loss is no more than

$$\begin{aligned} T\nu + \sum_{k=1}^{\log \frac{9}{\nu}} \frac{9\beta_k}{\sqrt{N_k}} \cdot (N_k + 1620d) &\leqslant T\nu + \sum_{k=1}^{\log \frac{9}{\nu}} 18\beta_k^2 \cdot 2^k + \sum_{k=1}^{\log \frac{9}{\nu}} 9 \cdot 2^{-k} \cdot 1620d \\ &\leqslant T\nu + \frac{324\beta^2}{\nu} + 14580d. \end{aligned}$$

By setting $\nu = 18\beta/\sqrt{T}$, we obtain the claimed result. □

We can also talk about the sample complexity of Algorithm 6.

**Theorem 8.4.4.** *Let $\Delta_{\min} = \min_{x' \in \mathcal{X}, x' \neq x_*} \langle x_* - x', \theta_* \rangle$. Then the worst case sample complexity of Algorithm 6 is $O(\frac{\beta^2}{\Delta_{\min}^2} + d \log \Delta_{\min}^{-1})$.*

*Proof.* We first argue that Algorithm 6 halts before $k = \log \frac{9}{\Delta_{\min}}$. Indeed, from the proof of Theorem 8.4.3, we have for any $x' \in \widehat{\mathcal{X}}_{k+1}$, $\langle x_* - x', \theta_* \rangle \leqslant \frac{9\beta_k}{\sqrt{N_k}}$. But when $k > \log \frac{9}{\Delta_{\min}}$, we have

$$\langle x_* - x', \theta_* \rangle \leqslant \frac{9\beta_k}{\sqrt{N_k}} \leqslant 9 \cdot 2^{-k} < \Delta_{\min}$$

which means we must have $x' = x_*$, namely now Algorithm 6 has to report the best result.

From the proof of Theorem 8.4.3 we also see that $N_k < (2^{k+1}\beta_k)^2$. Therefore the total sample complexity is

$$\sum_{k=1}^{\log \frac{9}{\Delta_{\min}}} (N_k + 1620d) \leqslant \sum_{k=1}^{\log \frac{9}{\Delta_{\min}}} \left( (2^{k+1}\beta_k)^2 + 1620d \right),$$

which is upper bounded by $O(\frac{\beta^2}{\Delta_{\min}^2} + d \log \Delta_{\min}^{-1})$. □

In Algorithm 6, there is one data-dependent parameter $\beta_k$. From Theorem 8.4.3, a small $\beta_k$ leads to a good regret bound. We have the following upper bound for $\beta_k$ for general convex set $K$.

**Lemma 8.4.5.** *For any integer $k$, recall that $\beta_k$ is induced from Eq.* (8.2)*, we have*

$$\beta_k \leqslant \Theta(\sqrt{d} \cdot \left(\sqrt{d} + \sqrt{\log(k^2/\delta)}\right)).$$

*Proof.* We first argue that for fixed $A \in \mathbb{R}^{d\times d}$, $t_*(K, A) \leqslant t_*(\mathbb{R}^d, A)$, where $t_*(\cdot,\cdot)$ satisfies the condition in Lemma 8.4.1. For brevity let $t_1 = t_*(K, A)$ and $t_2 = t_*(\mathbb{R}^d, A)$. Clearly we have

$$\mathop{\mathbf{E}}_{\xi\sim\mathcal{N}(0,I_d)}\left[\sup_{\theta,\theta'\in\mathbb{R}^d:\|\theta-\theta'\|_{X^\top X}\leqslant t_2} \langle (X^\top X)^{1/2}(\theta-\theta'), \xi\rangle\right] \leqslant t_2^2/2.$$

On the other hand, since $K \subset \mathbb{R}^d$, we have

$$\begin{aligned}
&\mathop{\mathbf{E}}_{\xi\sim\mathcal{N}(0,I_d)}\left[\sup_{\theta,\theta'\in K:\|\theta-\theta'\|_{X^\top X}\leqslant t_2} \langle (X^\top X)^{1/2}(\theta-\theta'), \xi\rangle\right] \\
\leqslant &\mathop{\mathbf{E}}_{\xi\sim\mathcal{N}(0,I_d)}\left[\sup_{\theta,\theta'\in\mathbb{R}^d:\|\theta-\theta'\|_{X^\top X}\leqslant t_2} \langle (X^\top X)^{1/2}(\theta-\theta'), \xi\rangle\right].
\end{aligned}$$

Hence we have $t_1 \leqslant t_2$.

Next we estimate $t_2$. Since

$$\mathop{\mathbf{E}}_{\xi\sim\mathcal{N}(0,I_d)}\left[\sup_{\theta,\theta'\in\mathbb{R}^d:\|\theta-\theta'\|_{X^\top X}\leqslant t_2} \langle (X^\top X)^{1/2}(\theta-\theta'), \xi\rangle\right] = \Theta(t_2\sqrt{d}),$$

we can choose $t_2$ to be $\Theta(\sqrt{d})$.

Putting everything together, we have

$$\begin{aligned}
&\min_{\lambda\in\Delta_{\mathcal{X}}}\max_{x\in\mathcal{X}} \sup_{\theta,\theta'\in K:\sqrt{N_k}\|\theta-\theta'\|_{(\sum_i \lambda_i x_i x_i^\top)}\leqslant\gamma(K,\delta/k^2)} \sqrt{N_k}\langle x, \theta-\theta'\rangle \\
\leqslant &\min_{\lambda\in\Delta_{\mathcal{X}}}\max_{x\in\mathcal{X}} \sup_{\theta,\theta'\in\mathbb{R}^d:\sqrt{N_k}\|\theta-\theta'\|_{(\sum_i \lambda_i x_i x_i^\top)}\leqslant\gamma(\mathbb{R}^d,\delta/k^2)} \sqrt{N_k}\langle x, \theta-\theta'\rangle \\
= &\min_{\lambda\in\Delta_{\mathcal{X}}}\max_{x\in\mathcal{X}} \sqrt{N_k}\cdot\|x\|_{(\sqrt{N_k}\sum_i \lambda_i x_i x_i^\top)^{-1}}\cdot\gamma(\mathbb{R}^d,\delta/k^2) \\
\leqslant &\Theta(\sqrt{N_k}\cdot\min_{\lambda\in\Delta_{\mathcal{X}}}\max_{x\in\mathcal{X}}\|x\|_{(\sqrt{N_k}\sum_i \lambda_i x_i x_i^\top)^{-1}}\cdot\left(\sqrt{d}+\sqrt{\log(k^2/\delta)}\right)) \\
= &\Theta(\sqrt{d}\cdot\left(\sqrt{d}+\sqrt{\log(k^2/\delta)}\right)).
\end{aligned}$$

where the last step follows from the Kiefer-Wolfowitz theorem for G-optimal design [111].

□

Note this implies a regret bound of $O(d\sqrt{T})$, which matches the state-of-art result.

### *8.5 Specific Examples*

In this section we compute $\beta$ for different cases. Since $\beta$ is a function of $t_*$, we also compute a variety of values of $t_*$.

Lemma 8.4.5 provides a universal upper bound for $\beta_k$. However, the geometry structure of $K$ is not taken into consideration, which inspires us to work on the following result.

**Lemma 8.5.1.** *Fix $A \in \mathbb{R}^{d\times d}$. Let $t_* = t_*(K, A)$ satisfy the condition in Lemma 8.4.1. Then*

$$t_* \leqslant \min\{2\sqrt{\operatorname{rank}(A)}, 2\sqrt{\|A^{1/2}\| w(K)}\},$$

*where $\|\cdot\|$ is the spectrum norm of matrices, and $w(K) := \mathbf{E}_{\xi\sim\mathcal{N}(0,I_d)}[\sup_{x\in K}\langle x, \xi\rangle]$ is the Gaussian mean width of convex set $K$.*

Before we proceed to the proof, we stress that an upper bound of $t_*$ implies an upper bound of $\beta$, as it is always true that $\beta_k \leqslant \sqrt{d}\cdot(t_* + \sqrt{2\log\frac{2k^2}{\delta}})$. Moreover, $\beta$ can be smaller than this bound.

*Proof.* Fix $t > 0$. We first upper bound

$$\mathop{\mathbf{E}}_{\xi\sim\mathcal{N}(0,I_d)}\left[\sup_{\theta,\theta'\in K:\|\theta-\theta'\|_A\leqslant t}\langle A^{1/2}(\theta-\theta'), \xi\rangle\right]$$

as a function of $t$, then give an estimation of $t_*$.

For any $\xi \in \mathcal{N}(0, I_d)$, we have

$$\sup_{\theta,\theta'\in K:\|\theta-\theta'\|_A\leqslant t}\langle A^{1/2}(\theta-\theta'), \xi\rangle \leqslant \min\{\sup_{\theta,\theta'\in K}\langle A^{1/2}(\theta-\theta'), \xi\rangle, \sup_{\|\theta-\theta'\|_A\leqslant t}\langle A^{1/2}(\theta-\theta'), \xi\rangle\}.$$

Therefore

$$\begin{aligned}
&\mathop{\mathbf{E}}_{\xi\sim\mathcal{N}(0,I_d)}\left[\sup_{\theta,\theta'\in K:\|\theta-\theta'\|_A\leqslant t}\langle A^{1/2}(\theta-\theta'),\xi\rangle\right]\\
\leqslant\ &\mathop{\mathbf{E}}_{\xi\sim\mathcal{N}(0,I_d)}\left[\min\{\sup_{\theta,\theta'\in K}\langle A^{1/2}(\theta-\theta'),\xi\rangle,\sup_{\|\theta-\theta'\|_A\leqslant t}\langle A^{1/2}(\theta-\theta'),\xi\rangle\}\right]\\
\leqslant\ &\min\{\mathop{\mathbf{E}}_{\xi\sim\mathcal{N}(0,I_d)}\left[\sup_{\theta,\theta'\in K}\langle A^{1/2}(\theta-\theta'),\xi\rangle\right],\mathop{\mathbf{E}}_{\xi\sim\mathcal{N}(0,I_d)}\left[\sup_{\|\theta-\theta'\|_A\leqslant t}\langle A^{1/2}(\theta-\theta'),\xi\rangle\right]\}.
\end{aligned}$$

We consider the terms on the right hand side separately. Notice that $\|\theta-\theta'\|_A\leqslant t$ implies that $\|A^{1/2}(\theta-\theta')\|_2\leqslant t$. Let $P\in\mathbb{R}^{d\times d}$ be the orthogonal projection onto the subspace spanned by the columns of $A^{1/2}$. Then

$$\begin{aligned}
\mathop{\mathbf{E}}_{\xi\sim\mathcal{N}(0,I_d)}\left[\sup_{\|\theta-\theta'\|_A\leqslant t}\langle A^{1/2}(\theta-\theta'),\xi\rangle\right]&=\mathop{\mathbf{E}}_{\xi\sim\mathcal{N}(0,I_d)}\left[\sup_{\|\theta-\theta'\|_A\leqslant t}\langle A^{1/2}(\theta-\theta'),P\xi\rangle\right]\\
&\leqslant\mathop{\mathbf{E}}_{\xi\sim\mathcal{N}(0,I_d)}[t\|P\xi\|_2]\\
&\leqslant t\sqrt{\operatorname{rank}(A)}.
\end{aligned}$$

Next we have

$$\mathop{\mathbf{E}}_{\xi\sim\mathcal{N}(0,I_d)}\left[\sup_{\theta,\theta'\in K}\langle A^{1/2}(\theta-\theta'),\xi\rangle\right]=w(A^{1/2}(K-K))\leqslant 2\|A^{1/2}\|w(K),$$

where the last step follows from Proposition 7.5.2 and Exercise 7.5.4 of [149].

Now we have

$$\mathop{\mathbf{E}}_{\xi\sim\mathcal{N}(0,I_d)}\left[\sup_{\theta,\theta'\in K:\|\theta-\theta'\|_A\leqslant t}\langle A^{1/2}(\theta-\theta'),\xi\rangle\right]\leqslant\min\{t\sqrt{d},2\|A^{1/2}\|w(K)\}.$$

By setting $\min\{t\sqrt{\operatorname{rank}(A)},2\|A^{1/2}\|w(K)\}=\frac{t^2}{2}$, we conclude that

$$t_*(K,A)\leqslant\min\{2\sqrt{\operatorname{rank}(A)},2\sqrt{\|A^{1/2}\|w(K)}\}.\quad\square$$

Lemma 8.5.1 provides some insight on $t_*$, as it is closely related to the Gaussian mean width of the set $K$. $w(K)$ can vary greatly for different convex set $K$. In the case of unit

$\ell_p$ ball in $\mathbb{R}^d$, we have $w(K) = \Theta(\log d)$ when $p = 1$, $w(K) = \Theta(\sqrt{d})$ when $p = 2$, and $w(K) = \Theta(d)$ when $p = \infty$.

In the rest of this section, we will examine some more specific examples of the constrained set $K$, and obtain more accurate bound on $\beta_k$.

*Linear subspaces*

Let $K$ be a linear subspace of dimension $p$. Let $P \in \mathbb{R}^{d\times d}$ be the projection matrix from $\mathbb{R}^d$ onto $K$. Notice that $P^2 = P$ and $P$ is symmetric. Moreover, $\mathrm{rank}(P) = p$. Therefore, any element in $K$ can be represented by $Py$ for some $y \in \mathbb{R}^d$. Fix integer $k > 0$. We compute $\beta_k$ explicitly.

For any $\lambda \in \Delta_{\mathcal{X}}$, we have

$$\begin{aligned}
&\max_{x\in\mathcal{X}} \sup_{\theta,\theta'\in K:\sqrt{N_k}\|\theta-\theta'\|_{(\sum_i \lambda_i x_i x_i^\top)}\leqslant\gamma(K,\delta/k^2)} \sqrt{N_k}\langle x, \theta-\theta'\rangle \\
=& \max_{x\in\mathcal{X}} \sup_{y,y'\in\mathbb{R}^d:\sqrt{N_k}\|Py-Py'\|_{(\sum_i \lambda_i x_i x_i^\top)}\leqslant\gamma(K,\delta/k^2)} \sqrt{N_k}\langle x, Py-Py'\rangle \\
=& \max_{x\in\mathcal{X}} \sup_{y,y'\in\mathbb{R}^d:\sqrt{N_k}\|Py-Py'\|_{(\sum_i \lambda_i x_i x_i^\top)}\leqslant\gamma(K,\delta/k^2)} \sqrt{N_k}\langle Px, Py-Py'\rangle \\
\leqslant& \max_{x\in\mathcal{X}} \sup_{y,y'\in\mathbb{R}^d:\sqrt{N_k}\|Py-Py'\|_{(\sum_i \lambda_i x_i x_i^\top)}\leqslant\gamma(K,\delta/k^2)} \sqrt{N_k}\|Px\|_{(\sum_i \lambda_i x_i x_i^\top)^{-1}} \cdot \|Py-Py'\|_{(\sum_i \lambda_i x_i x_i^\top)} \\
=& \max_{x\in\mathcal{X}} \|Px\|_{(\sum_i \lambda_i x_i x_i^\top)^{-1}} \cdot \gamma(K,\delta/k^2).
\end{aligned}$$

where the third step follows from Cauchy-Swartz inequality.

We then argue that $\min_{\lambda\in\Delta_{\mathcal{X}}} \max_{x\in\mathcal{X}} \|Px\|_{(\sum_i \lambda_i x_i x_i^\top)^{-1}} \leqslant \sqrt{p}$. Notice that

$$\begin{aligned}
\sum_i \lambda_i \|Px_i\|^2_{(\sum_i \lambda_i x_i x_i^\top)^{-1}} &= \sum_i \lambda_i x_i^\top P^\top (\sum_i \lambda_i x_i x_i^\top)^{-1} P x_i \\
&= \sum_i \mathsf{Tr}[\lambda_i P x_i x_i^\top P^\top (\sum_i \lambda_i x_i x_i^\top)^{-1}] \\
&= \mathsf{Tr}[(\sum_i \lambda_i P x_i x_i^\top P^\top)(\sum_i \lambda_i x_i x_i^\top)^{-1}].
\end{aligned}$$

Because $\mathrm{rank}((\sum_i \lambda_i P x_i x_i^\top P^\top)) \leqslant \mathrm{rank}(P) = p$, and $(\sum_i \lambda_i x_i x_i^\top) \succeq (\sum_i \lambda_i P x_i x_i^\top P^\top)$, we have that

$$\mathsf{Tr}[(\sum_i \lambda_i P x_i x_i^\top P^\top)(\sum_i \lambda_i x_i x_i^\top)^{-1}] \leqslant p.$$

which proves the claimed result.

Finally, since $\gamma(K, \delta/k^2) = \gamma(\mathbb{R}^p, \delta/k^2) = \sqrt{p} + \sqrt{2\log(2k^2/\delta)}$, we can compute $\beta_k$ as in the following proposition:

**Proposition 8.5.2.** *Let $K$ be a linear subspace of dimension $p$. Then we have*

$$\beta_k \leqslant \sqrt{p} \cdot (\sqrt{p} + 2\sqrt{\log(2k^2/\delta)}).$$

*As a corollary, Algorithm 6 has a regret bound of $O\left((p + \sqrt{p \log \frac{\log T}{\delta}}) \cdot \sqrt{T} + d\right)$, and a worst case sample complexity of $O(\frac{(p+\sqrt{p \log \frac{\log T}{\delta}})^2}{\Delta_{\min}^2} + d \log \Delta_{\min}^{-1})$.*

*Ellipsoids*

Let $R \in \mathbb{R}^{d \times d}$ be a positive-semidefinite (PSD) matrix, and $K = \{x \in \mathbb{R} : x^\top R x \leqslant 1\}$. We have the following upper bound on $t_*(K, A)$.

**Proposition 8.5.3.** *Fix $A \in \mathbb{R}^{d \times d}$. Let $t_* = t_*(K, A)$ satisfy the condition in Lemma 8.4.1. Then*

$$t_* \leqslant \min_{\alpha \geqslant 0} 4\sqrt{(\alpha + 1)\mathsf{Tr}[A(4A + \alpha R)^{-1}]}.$$

*Proof.* Similar to the proof of Lemma 8.5.1, we start with upper bounding the term

$$\sup_{\theta, \theta' \in K : \|\theta - \theta'\|_A \leqslant t} \langle A^{1/2}(\theta - \theta'), \xi \rangle$$

for fixed $\xi \in \mathbb{R}^d$. Since $K$ is convex and centrally symmetric, there always exists an optimal solution of the form $\theta' = -\theta$. Hence

$$\sup_{\theta, \theta' \in K : \|\theta - \theta'\|_A \leqslant t} \langle A^{1/2}(\theta - \theta'), \xi \rangle = \sup_{\theta^\top R \theta \leqslant 1, \|2\theta\|_A \leqslant t} \langle 2A^{1/2}\theta, \xi \rangle.$$

Let us consider the Lagrangian multipliers of the RHS, which is

$$L(\theta, \lambda, \nu) := \langle 2A^{1/2}\theta, \xi \rangle - \lambda(\theta^\top R\theta - 1) - \nu(4\theta^\top A\theta - t^2),$$

By setting $\frac{\partial L}{\partial \theta} = 0$. we have

$$\theta = (4\nu A + \lambda R)^{-1} A^{1/2}\xi.$$

Therefore for fixed $\lambda \geqslant 0$ and $\nu \geqslant 0$,

$$\begin{aligned}
&\max_\theta L(\theta, \lambda, \nu) \\
&= \lambda + t^2\nu + \xi^\top A^{1/2}(4\nu A + \lambda R)^{-1}(2(4\nu A + \lambda R) - \lambda R - 4\nu A)(4\nu A + \lambda R)^{-1}A^{1/2}\xi \\
&= \lambda + t^2\nu + \xi^\top A^{1/2}(4\nu A + \lambda R)^{-1}A^{1/2}\xi \\
&\geqslant 2\sqrt{(\lambda/\nu + t^2)\xi^\top A^{1/2}(4A + \lambda/\nu R)^{-1}A^{1/2}\xi}.
\end{aligned}$$

Let $\alpha = \lambda/\nu$, then we can conclude that

$$\sup_{\theta,\theta' \in K: \|\theta - \theta'\|_A \leqslant t} \langle A^{1/2}(\theta - \theta'), \xi \rangle = \min_{\alpha \geqslant 0} 2\sqrt{(\alpha + t^2)\xi^\top A^{1/2}(4A + \alpha R)^{-1}A^{1/2}\xi}.$$

Now we have

$$\begin{aligned}
&\mathop{\mathbf{E}}_{\xi \sim \mathcal{N}(0, I_d)} \left[ \sup_{\theta,\theta' \in K: \|\theta - \theta'\|_A \leqslant t} \langle A^{1/2}(\theta - \theta'), \xi \rangle \right] \\
=\ &\mathop{\mathbf{E}}_{\xi \sim \mathcal{N}(0, I_d)} \left[ \min_{\alpha \geqslant 0} 2\sqrt{(\alpha + t^2)\xi^\top A^{1/2}(4A + \alpha R)^{-1}A^{1/2}\xi} \right] \\
\leqslant\ &\min_{\alpha \geqslant 0} \mathop{\mathbf{E}}_{\xi \sim \mathcal{N}(0, I_d)} \left[ 2\sqrt{(\alpha + t^2)\xi^\top A^{1/2}(4A + \alpha R)^{-1}A^{1/2}\xi} \right] \\
\leqslant\ &\min_{\alpha \geqslant 0} 2\sqrt{\mathop{\mathbf{E}}_{\xi \sim \mathcal{N}(0, I_d)} \left[ (\alpha + t^2)\xi^\top A^{1/2}(4A + \alpha R)^{-1}A^{1/2}\xi \right]} \\
=\ &\min_{\alpha \geqslant 0} 2\sqrt{(\alpha + t^2)\mathsf{Tr}[A(4A + \alpha R)^{-1}]}
\end{aligned}$$

where the third step follows from Cauchy-Schwartz inequality, and the last step follows from the cyclic property of trace.

To complete the proof, for fixed $\alpha$, we want to choose $t_*$ so that

$$2\sqrt{(\alpha + t_*^2)\mathsf{Tr}[A(4A + \alpha R)^{-1}]} \leqslant t_*^2/2.$$

It thus suffices to pick up $t_* = \{4\sqrt{(\alpha+1)\mathsf{Tr}[A(4A+\alpha R)^{-1}]}$. □

We can also estimate the corresponding $\beta_k$ value.

**Proposition 8.5.4.**

$$\beta_k \leqslant \min_{\lambda \in \Delta_{\mathcal{X}}} \max_{x \in \mathcal{X}} \min_{\alpha \geqslant 0} 2\|x\|_{\left(\frac{4}{N_k\alpha + \gamma^2(K,\delta/k^2)}(\sum_i \lambda_i x_i x_i^\top) + \frac{\alpha}{N_k\alpha+\gamma^2(K,\delta/k^2)}R\right)^{-1}}$$

Before we proceed to the proof, let us first get some better sense of the implication of Proposition 8.5.4. Recall from Proposition 8.5.3, we have $\gamma(K, \delta/k^2) = t_* + \sqrt{2\log(2k^2/\delta)} \leqslant 2(\sqrt{d} + \sqrt{2\log(2k^2/\delta)})$. Hence when $\alpha = 0$, we have $\beta_k \leqslant 2\sqrt{d}(\sqrt{d} + \sqrt{2\log(2k^2/\delta)})$; And when $\alpha \to \infty$, we have $\beta_k \leqslant \sqrt{N_k}\max_{x\in\mathcal{X}} \|x\|_{R^{-1}}$.

*Proof of Proposition 8.5.4.* Fix $\lambda \in \Delta_{\mathcal{X}}$ and $x \in \mathcal{X}$. We can again set $\theta' = -\theta$ and compute Lagrangian multipliers for the program

$$\sup_{\theta,\theta' \in K: \|\theta - \theta'\|_{(\sum_i \lambda_i x_i x_i^\top)} \leqslant t} \langle x, \theta - \theta' \rangle,$$

which is

$$L(\theta, \rho, \nu) := \langle 2\theta, x\rangle - \rho(\theta^\top R\theta - 1) - \nu(4\theta^\top(\sum_i \lambda_i x_i x_i^\top)\theta - t^2),$$

By setting $\frac{\partial L}{\partial \theta} = 0$. we have

$$\theta = (4\nu(\sum_i \lambda_i x_i x_i^\top) + \rho R)^{-1} x.$$

Therefore for fixed $\rho \geqslant 0$ and $\nu \geqslant 0$,

$$\begin{aligned}
&\max_{\theta} L(\theta, \rho, \nu) \\
&= \rho + t^2\nu + x^\top (4\nu(\sum_i \lambda_i x_i x_i^\top) + \rho R)^{-1}(4\nu(\sum_i \lambda_i x_i x_i^\top) + \rho R)(4\nu(\sum_i \lambda_i x_i x_i^\top) + \rho R)^{-1}x \\
&= \rho + t^2\nu + x^\top (4\nu(\sum_i \lambda_i x_i x_i^\top) + \rho R)^{-1}x \\
&\geqslant 2\sqrt{(\rho/\nu + t^2)x^\top (4(\sum_i \lambda_i x_i x_i^\top) + \rho/\nu R)^{-1}x}.
\end{aligned}$$

Let $\alpha = \rho/\nu$, then we can conclude that

$$\begin{aligned}
&\sup_{\theta,\theta' \in K: \sqrt{N_k}\|\theta-\theta'\|_{(\sum_i \lambda_i x_i x_i^\top)} \leqslant \gamma(K, \delta/k^2)} \sqrt{N_k}\langle x, \theta - \theta' \rangle \\
&= \min_{\alpha \geqslant 0} 2\sqrt{N_k}\sqrt{(\alpha + \frac{\gamma^2(K, \delta/k^2)}{N_k})x^\top (4(\sum_i \lambda_i x_i x_i^\top) + \alpha R)^{-1}x} \\
&= \min_{\alpha \geqslant 0} 2\|x\|_{\left(\frac{4}{N_k\alpha + \gamma^2(K,\delta/k^2)}(\sum_i \lambda_i x_i x_i^\top) + \frac{\alpha}{N_k\alpha + \gamma^2(K,\delta/k^2)}R\right)^{-1}},
\end{aligned}$$

which completes the proof.

□

*Convex Polytopes*

Let $C \in \mathbb{R}^{m \times d}, b \in \mathbb{R}^m$, and $K = \{x \in \mathbb{R} : Cx \leqslant b\}$. For simplicity we assume $K$ is centrally symmetric, by forcing that whenever we have a constraint $c^\top x \leqslant b'$, we also have a constraint $-c^\top x \leqslant b'$. We have the following upper bound on $t_*(K, A)$.

**Proposition 8.5.5.** *Fix $A \in \mathbb{R}^{d \times d}$. Let $t_* = t_*(K, A)$ satisfy the condition in Lemma 8.4.1. Then $t_*$ can be the smallest positive number that satisfies*

$$\mathop{\mathbf{E}}_{\xi \sim \mathcal{N}(0, I_d)} \left[ \min_{\lambda \in \mathbb{R}^m_{\geqslant 0}} \left( \langle \lambda, b \rangle + t\|\xi - \frac{A^{-1/2}C^\top \lambda}{2}\|_2 \right) \right] \leqslant t_*^2/2$$

*Proof.* We again apply the Lagrangian multipliers method, which is

$$L(\theta, \lambda, \nu) := \langle 2A^{1/2}\theta, \xi \rangle - \langle \lambda, C\theta - b \rangle - \nu(4\theta^\top A\theta - t^2).$$

By setting $\frac{\partial L}{\partial \theta} = 0$. we have

$$\theta = (8\nu A)^{-1}(2A^{1/2}\xi - C^\top \lambda).$$

Therefore for fixed $\lambda \in \mathbb{R}^m_{\geqslant 0}$ and $\nu \geqslant 0$,

$$\begin{aligned}
\max_\theta L(\theta, \lambda, \nu) &= \langle \lambda, b \rangle + t^2\nu + \frac{\|\eta\|_2^2}{4\nu} + \frac{\lambda^\top C A^{-1} C^\top \lambda}{16\nu} - 2\eta^\top A^{1/2}(8\nu A)^{-1} C^\top \lambda \\
&= \langle \lambda, b \rangle + t^2\nu + (4\nu)^{-1}\|\xi - \frac{A^{-1/2}C^\top \lambda}{2}\|_2^2 \\
&\geqslant \langle \lambda, b \rangle + t\|\xi - \frac{A^{-1/2}C^\top \lambda}{2}\|_2.
\end{aligned}$$

Hence

$$\sup_{\theta,\theta' \in K : \|\theta - \theta'\|_A \leqslant t} \langle A^{1/2}(\theta - \theta'), \xi \rangle = \min_{\lambda \in \mathbb{R}^m_{\geqslant 0}} (\langle \lambda, b \rangle + t\|\xi - \frac{A^{-1/2}C^\top \lambda}{2}\|_2).$$

This gives us

$$\begin{aligned}
&\mathop{\mathbf{E}}_{\xi \sim \mathcal{N}(0, I_d)} \left[ \sup_{\theta,\theta' \in K : \|\theta - \theta'\|_A \leqslant t} \langle A^{1/2}(\theta - \theta'), \xi \rangle \right] \\
= &\mathop{\mathbf{E}}_{\xi \sim \mathcal{N}(0, I_d)} \left[ \min_{\lambda \in \mathbb{R}^m_{\geqslant 0}} (\langle \lambda, b \rangle + t\|\xi - \frac{A^{-1/2}C^\top \lambda}{2}\|_2) \right] \\
\leqslant &\min_{\lambda \in \mathbb{R}^m_{\geqslant 0}} \mathop{\mathbf{E}}_{\xi \sim \mathcal{N}(0, I_d)} \left[ \langle \lambda, b \rangle + t\|\xi - \frac{A^{-1/2}C^\top \lambda}{2}\|_2 \right].
\end{aligned}$$

which completes the proof. □

### *8.6 Conclusion and Open Problems*

In this chapter we discuss how to apply the structure of constraints to speed up the linear bandits problem. As a concrete example, we observe improved performance in the case of linear subspaces.

The regret and sample complexity of Algorithm 6 is closely related to the data dependent parameter $\beta$. It would be interesting to work out tighter bounds of $\beta$ for other

convex sets besides linear subspaces. Lemma 8.5.1 suggests that $t_*$ may be related to the Gaussian mean width of $K$. Since $\beta$ and $t_*$ are closely relevant, can we upper bound $\beta$ as a function of the Gaussian width of $K$ and the geometry of $\mathcal{X}$?

# Chapter 9

# CONCLUSION

In this thesis, we have presented results about algorithms and complexity for several supervised learning problems. In Chapter 3, we proved strong time-space tradeoffs for general discrete problems of learning from uniform random samples. In particular, our lower bound holds for learning problems where (1) the size of the sample space and the size of the concept space can be different and (2) the labels can be more than binary. One important application of Chapter 3 is a lower bound for learning polynomials on prime fields over random evaluations, which requires understanding the distribution of the bias of Reed-Muller codes. Chapter 4 gave a sharp bound for this problem. In Chapter 5, we showed a generic PAC-learning algorithm with small space usage, by taking reduction from decision trees. Chapter 6 concerned the total least squares problem. We exhibited a novel algorithm with strong guarantees in both theory and experiments. In Chapter 7, we studied the problem of computing the John Ellipsoid of convex polytopes, which is equivalent to the problem of D-optimal design. We developed a simple algorithm based on convexity, as well as advances in randomized linear algebra, and its experimental evaluation outperformed all existing algorithms significantly. Finally, in Chapter 8, we looked into the problem of linear bandits and optimal experimental design with the presence of constraints. We obtained the first regret bound that takes advantage of the geometry properties of the constraints.

Learning theory is an extremely broad field, and this thesis only touched a very small part of it. There are many interesting open problems, some of which have already been mentioned in the individual chapters. We now discuss some interesting problems that are not covered.

In Chapter 3 we studied the samples-memory lower bound in the setting of online learning from uniform random samples. Many works [113, 114, 84, 102, 56, 12] have focused on this model. One important feature of these lower bounds is that each new sample is "fresh", namely it is independent from previous samples. From this perspective, we can say that these lower bounds hold for algorithms that can be modeled by *read-once branching programs*. From streaming algorithms, we know that the sample complexity may drastically drop if we take multiple passes of the data stream. Take parity learning as an example. Because we know that Gaussian elimination can be done in the circuit class **NC**, we can simulate the running of the circuit in the online learning model by iterating over $n$ samples for $n^{O(\mathsf{poly}(\log n))}$ passes. For the space usage, we need $O(\mathsf{poly}(\log n))$ to track where we are in the circuit, and $n$ bits to output. Clearly more passes are helpful, and the question is, with $k$ passes, what kind of time-space tradeoff can we prove? Garg, Raz, Tal [58] made a first attempt on this problem, by showing that when $k = 2$, either $2^{\Omega(\sqrt{n})}$ samples or $\Omega(n^{1.5})$ space is required for parity learning. It is not clear if their result is tight, and it would be interesting if we can prove for general $k$.

Another interesting question is time-space tradeoffs for continuous problems. The continuous analog of parity learning is simply solving random linear systems. But with method like gradient descent, on $d$ variables, an $\varepsilon$-close answer can be easily obtained with $O(d \log(1/\varepsilon))$ samples and $O(d \log(1/\varepsilon))$ space [135]. However, Sharan, Sidford and Valiant [120] showed that similar tradeoffs still exist: any successful learning algorithm requires either $\Omega(d \log\log(1/\varepsilon))$ samples or $\Omega(d^2)$ space. There is still room to improve the lower bound on the sample complexity from $\Omega(d \log\log(1/\varepsilon))$ to $\Omega(d \log(1/\varepsilon))$. Moreover, for problems with $|\mathcal{X}| \neq |\mathcal{C}|$, it will be of great interest to derive similar types of lower bounds.

Finally, in Chapter 5 we presented one algorithm that refutes the possibility of using statistical query dimension to characterize bounded space learnability when $|\mathcal{X}| \neq |\mathcal{C}|^{\Theta(1)}$. Is there a simple characterization that holds in this range of parameters?

# BIBLIOGRAPHY

[1] Emmanuel Abbe, Amir Shpilka, and Avi Wigderson. Reed-Muller codes for random erasures and errors. *IEEE Transactions on Information Theory*, 61(10):5229–5252, 2015.

[2] Zeyuan Allen-Zhu, Yuanzhi Li, Aarti Singh, and Yining Wang. Near-optimal design of experiments via regret minimization. In *International Conference on Machine Learning*, pages 126–135. `https://arxiv.org/pdf/1711.05174`, 2017.

[3] Zeyuan Allen-Zhu, Yuanzhi Li, Aarti Singh, and Yining Wang. Near-optimal discrete optimization for experimental design: A regret minimization approach. *Mathematical Programming*, pages 1–40, 2020.

[4] Alexandr Andoni, Chengyu Lin, Ying Sheng, Peilin Zhong, and Ruiqi Zhong. Subspace embedding and linear regression with Orlicz norm. In *International Conference on Machine Learning*, pages 224–233. `https://arxiv.org/pdf/1806.06430`, 2018.

[5] Kurt M Anstreicher. Improved complexity for maximum volume inscribed ellipsoids. *SIAM Journal on Optimization*, 13(2):309–320, 2002.

[6] Corwin L Atwood. Optimal and efficient designs of experiments. *The Annals of Mathematical Statistics*, pages 1570–1602, 1969.

[7] Haim Avron, Kenneth L. Clarkson, and David P. Woodruff. Sharper bounds for regularized data fitting. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017, August 16-18, 2017, Berkeley, CA, USA*, pages 27:1–27:22, 2017.

[8] Paul Beame, Shayan Oveis Gharan, and Xin Yang. On the bias of reed–muller codes over odd prime fields. *SIAM Journal on Discrete Mathematics*, 34(2):1232–1247, 2020.

[9] Paul Beame, Shayan Oveis Gharan, and Xin Yang. Time-space tradeoffs for learning from small test spaces: Learning low degree polynomial functions. Technical Report TR17-120, Electronic Colloquium on Computational Complexity (ECCC), 2017.

[10] Paul Beame, Shayan Oveis Gharan, and Xin Yang. On the bias of Reed-Muller codes over odd prime fields. *arXiv preprint arXiv:1806.06973v1*, 2018.

[11] Paul Beame, Shayan Oveis Gharan, and Xin Yang. Time-space tradeoffs for learning finite functions from random evaluations, with applications to polynomials. In *Conference On Learning Theory*, pages 843–856, 2018.

[12] Paul Beame, Shayan Oveis Gharan, and Xin Yang. Time-space tradeoffs for learning finite functions from random evaluations, with applications to polynomials. Technical Report TR18-114, Electronic Colloquium on Computational Complexity (ECCC), 2018.

[13] Peter Beelen and Mrinmoy Datta. Generalized Hamming weights of affine Cartesian codes. *Finite Fields and Their Applications*, 51:130–145, 2018.

[14] Pierre Bellec. Sharp oracle inequalities for least squares estimators in shape restricted regression. *The Annals of Statistics*, 46(2):745–780, 2018.

[15] Ido Ben-Eliezer, Rani Hod, and Shachar Lovett. Random low-degree polynomials are hard to approximate. *Computational Complexity*, 21(1):63–81, 2012.

[16] Elwyn R Berlekamp. *Algebraic Coding Theory*. McGraw-Hill, 1968.

[17] Abhishek Bhowmick and Shachar Lovett. Bias vs structure of polynomials in large fields, and applications in effective algebraic geometry and coding theory. *CoRR*, abs/1506.02047, 2015.

[18] Abhishek Bhowmick and Shachar Lovett. The list decoding radius for Reed-Muller codes over small fields. *IEEE Trans. Information Theory*, 64(6):4382–4391, 2018.

[19] Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration inequalities: A nonasymptotic theory of independence*. Oxford university press, 2013.

[20] Christos Boutsidis and David P Woodruff. Optimal CUR matrix decompositions. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 353–362. ACM, `https://arxiv.org/pdf/1405.7910`, 2014.

[21] Christos Boutsidis, David P Woodruff, and Peilin Zhong. Optimal principal component analysis in distributed and streaming models. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 236–249. ACM, `https://arxiv.org/pdf/1504.06729`, 2016.

[22] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, Cambridge, 2004.

[23] Sébastien Bubeck, Nicolo Cesa-Bianchi, and Sham Kakade. Towards minimax policies for online linear optimization with bandit feedback. In *Annual Conference on Learning Theory*, volume 23, pages 41–1. Microtome, 2012.

[24] Michal Černỳ and Milan Hladík. Two complexity results on C-optimality in experimental design. *Computational Optimization and Applications*, 51(3):1397–1408, 2012.

[25] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *Automata, Languages and Programming*, pages 693–703. Springer, 2002.

[26] Sabyasachi Chatterjee and John Lafferty. Adaptive risk bounds in unimodal regression. *Bernoulli*, 25(1):1–25, 2019.

[27] Sourav Chatterjee. A new perspective on least squares under convex constraint. *The Annals of Statistics*, 42(6):2340–2381, 2014.

[28] Yuansi Chen, Raaz Dwivedi, Martin J Wainwright, and Bin Yu. Fast MCMC sampling algorithms on polytopes. *The Journal of Machine Learning Research*, 19(1):2146–2231, 2018.

[29] Kenneth L. Clarkson, Ruosong Wang, and David P Woodruff. Dimensionality reduction for tukey regression. In *ICML*. arXiv preprint arXiv:1904.05543, 2019.

[30] Kenneth L Clarkson and David P Woodruff. Low rank approximation and regression in input sparsity time. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 81–90. ACM, 2013.

[31] Kenneth L Clarkson and David P Woodruff. Input sparsity and hardness for robust subspace approximation. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 310–329. IEEE, 2015.

[32] Kenneth L Clarkson and David P Woodruff. Sketching for M-estimators: A unified approach to robust regression. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 921–939. Society for Industrial and Applied Mathematics, 2015.

[33] Michael B Cohen, Ben Cousins, Yin Tat Lee, and Xin Yang. A near-optimal algorithm for approximating the john ellipsoid. In *COLT*, 2019.

[34] Michael B Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Madalina Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 163–172. ACM, 2015.

[35] Michael B. Cohen, Rasmus Kyng, Gary L. Miller, Jakub W. Pachocki, Richard Peng, Anup B. Rao, and Shen Chen Xu. Solving SDD linear systems in nearly $m\log^{1/2}n$ time. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, STOC '14, 2014.

[36] Michael B Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 181–190. ACM, 2015.

[37] Michael B Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC)*. `https://arxiv.org/pdf/1810.07896.pdf`, 2019.

[38] Michael B Cohen and Richard Peng. $\ell_p$ row sampling by Lewis Weights. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 183–192. ACM, `https://arxiv.org/pdf/1412.0588`, 2015.

[39] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 1–6. ACM, 1987.

[40] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553, 2009.

[41] Samuel I Daitch and Daniel A Spielman. Faster approximate lossy generalized flow via interior point algorithms. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 451–460. ACM, 2008.

[42] S Damla Ahipasaoglu, Peng Sun, and Michael J Todd. Linear convergence of a modified Frank–Wolfe algorithm for computing minimum-volume enclosing ellipsoids. *Optimisation Methods and Software*, 23(1):5–19, 2008.

[43] Rémy Degenne, Pierre Ménard, Xuedong Shang, and Michal Valko. Gamification of pure exploration for linear bandits. In *International Conference on Machine Learning*, pages 2432–2442. PMLR, 2020.

[44] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, 2019.

[45] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017.

[46] Huaian Diao, Rajesh Jayaram, Zhao Song, Wen Sun, and David P. Woodruff. Optimal sketching for kronecker product regression and low rank approximation. In *NeurIPS*, 2019.

[47] Huaian Diao, Zhao Song, Wen Sun, and David Woodruff. Sketching for kronecker product regression and p-splines. In *International Conference on Artificial Intelligence and Statistics*, pages 1299–1308, 2018.

[48] Huaian Diao, Zhao Song, David Woodruff, and Xin Yang. Total least squares regression in input sparsity time. In *Advances in Neural Information Processing Systems*, pages 2482–2493, 2019.

[49] Leonard E. Dickson. *Linear Groups with an Exposition of the Galois Field Theory*. B.G. Trubner, Leipzig, 1901.

[50] Petros Drineas, Malik Magdon-Ismail, Michael W Mahoney, and David P Woodruff. Fast approximation of matrix coherence and statistical leverage. *Journal of Machine Learning Research*, 13(Dec):3475–3506, 2012.

[51] Petros Drineas, Michael W. Mahoney, and S. Muthukrishnan. Sampling algorithms for l2 regression and applications. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, SODA '06, pages 1127–1136, Philadelphia, PA, USA, 2006. Society for Industrial and Applied Mathematics.

[52] Petros Drineas, Michael W. Mahoney, and S. Muthukrishnan. Subspace sampling and relative-error matrix approximation: Column-row-based methods. In *Algorithms - ESA 2006, 14th Annual European Symposium, Zurich, Switzerland, September 11-13, 2006, Proceedings*, pages 304–314, 2006.

[53] Petros Drineas, Michael W Mahoney, S Muthukrishnan, and Tamás Sarlós. Faster least squares approximation. *Numerische mathematik*, 117(2):219–249, 2011.

[54] Tanner Fiez, Lalit Jain, Kevin G Jamieson, and Lillian Ratliff. Sequential experimental design for transductive linear bandits. In *Advances in Neural Information Processing Systems*, pages 10666–10676, 2019.

[55] Shmuel Friedland and Anatoli Torokhti. Generalized rank-constrained matrix approximations. *SIAM Journal on Matrix Analysis and Applications*, 29(2):656–659, 2007.

[56] Sumegha Garg, Ran Raz, and Avishay Tal. Extractor-based time-space lower bounds for learning. In *STOC 2018*, 2017.

[57] Sumegha Garg, Ran Raz, and Avishay Tal. Extractor-based time-space tradeoffs for learning. Technical Report TR17-121, Electronic Colloquium on Computational Complexity (ECCC), 2017.

[58] Sumegha Garg, Ran Raz, and Avishay Tal. Time-space lower bounds for two-pass learning. In *34th Computational Complexity Conference (CCC 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[59] Alon Gonen, Shachar Lovett, and Michal Moshkovitz. Towards a combinatorial characterization of bounded memory learning. *arXiv preprint arXiv:2002.03123*, 2020.

[60] Alon Gonen, Shachar Lovett, and Michal Moshkovitz. Towards a combinatorial characterization of bounded memory learning. In *NeurIPS*, 2020.

[61] Parikshit Gopalan, Adam R Klivans, and David Zuckerman. List-decoding Reed-Muller codes over small fields. In *Proceedings of the Fortieth Annual ACM symposium on Theory of Computing*, pages 265–274. ACM, 2008.

[62] Izrail Solomonovich Gradshteyn and Iosif Moiseevich Ryzhik. *Table of integrals, series, and products*. Academic press, 2014.

[63] Ben Green and Terence Tao. The distribution of polynomials over finite fields, with applications to the gowers norms. *arXiv preprint arXiv:0711.3191*, 2007.

[64] Adam Gustafson and Hariharan Narayanan. John's walk. *arXiv preprint arXiv:1803.02032*, 2018.

[65] David H Gutman and Javier F Peña. A unified framework for bregman proximal methods: subgradient, gradient, and accelerated gradient schemes. *arXiv preprint arXiv:1812.10198*, 2018.

[66] Hulda S. Haraldsdottir, Ben Cousins, Ines Thiele, Ronan M. T. Fleming, and Santosh Vempala. CHRR: coordinate hit-and-run with rounding for uniform sampling of constraint-based models. *Bioinformatics*, 33(11), 1 2017.

[67] Elad Haramaty and Amir Shpilka. On the structure of cubic and quartic polynomials. In *Proceedings of the Forty-Second Annual ACM Symposium on Theory of Computing, STOC 2010, Cambridge, MA, USA*, pages 331–340, 2010.

[68] Elad Hazan and Zohar Karnin. Volumetric spanners: an efficient exploration basis for learning. *The Journal of Machine Learning Research*, 17(1):4062–4095, 2016.

[69] Petra Heijnen and Ruud Pellikaan. Generalized Hamming weights of q-ary Reed-Muller codes. *IEEE Transactions on Information Theory*, 44(1):181–196, 1998.

[70] Kenneth Ireland and Michael Rosen. *A classical introduction to modern number theory*, volume 84. Springer Science & Business Media, 2013.

[71] GJO Jameson. Inequalities for gamma function ratios. *The American Mathematical Monthly*, 120(10):936–940, 2013.

[72] Fritz John. Extremum problems with inequalities as subsidiary conditions. In *Studies and Essays Presented to R. Courant on his 60th Birthday, January 8, 1948*, pages 187–204. Interscience Publishers, Inc., New York, N. Y., 1948.

[73] Ravi Kannan, László Lovász, and Miklós Simonovits. Random walks and an $O^*(n^5)$ volume algorithm for convex bodies. *Random Structures Algorithms*, 11(1):1–50, 1997.

[74] Tadao Kasami and Nobuki Tokura. On the weight structure of Reed-Muller codes. *IEEE Transactions on Information Theory*, 16(6):752–759, 1970.

[75] Tadao Kasami, Nobuki Tokura, and Saburo Azumi. On the weight enumeration of weights less than 2.5$d$ of Reed-Muller codes. *Information and Control*, 30(4):380–395, 1976.

[76] Tali Kaufman, Shachar Lovett, and Ely Porat. Weight distribution and list-decoding size of Reed-Muller codes. *IEEE Trans. Information Theory*, 58(5):2689–2696, 2012.

[77] Michael Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM (JACM)*, 45(6):983–1006, 1998.

[78] Peter Keevash and Benny Sudakov. Set systems with restricted cross-intersections and the minimum rank ofinclusion matrices. *SIAM Journal on Discrete Mathematics*, 18(4):713–727, 2005.

[79] Jonathan A Kelner, Lorenzo Orecchia, Aaron Sidford, and Zeyuan Allen Zhu. A simple, combinatorial algorithm for solving SDD systems in nearly-linear time. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 911–920. ACM, 2013.

[80] L. Khachiyan, S. Tarasov, and I. Ehrlich. The method of inscribed ellipsoids. *Soviet Math. Doklady*, 1988.

[81] Leonid G Khachiyan. Rounding of polytopes in the real number model of computation. *Mathematics of Operations Research*, 21(2):307–320, 1996.

[82] Leonid G Khachiyan and Michael J Todd. On the complexity of approximating the maximal inscribed ellipsoid for a polytope. *Mathematical Programming*, 61(1):137–159, 1993.

[83] Jack Kiefer and Jacob Wolfowitz. The equivalence of two extremum problems. *Canadian Journal of Mathematics*, 12(363-366):234, 1960.

[84] Gillat Kol, Ran Raz, and Avishay Tal. Time-space hardness of learning sparse parities. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 1067–1080, 2017.

[85] Piyush Kumar and E Alper Yildirim. Minimum-volume enclosing ellipsoids and core sets. *Journal of Optimization Theory and Applications*, 126(1):1–21, 2005.

[86] Rasmus Kyng, Yin Tat Lee, Richard Peng, Sushant Sachdeva, and Daniel A Spielman. Sparsified cholesky and multigrid solvers for connection laplacians. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 842–850. ACM, 2016.

[87] Rasmus Kyng and Sushant Sachdeva. Approximate gaussian elimination for laplacians-fast, sparse, and simple. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 573–582. IEEE, 2016.

[88] Jörg Lampe and Heinrich Voss. Solving regularized total least squares problems based on eigenproblems. *Taiwanese Journal of Mathematics*, 14(3A):885–909, 2010.

[89] Jorg Lampe and Heinrich Voss. Large-scale dual regularized total least squares. *Electronic Transactions on Numerical Analysis*, 42:13–40, 2014.

[90] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[91] Beatrice Laurent and Pascal Massart. Adaptive estimation of a quadratic functional by model selection. *Annals of Statistics*, pages 1302–1338, 2000.

[92] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[93] Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in $O(\sqrt{rank})$ iterations and faster algorithms for maximum flow. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 424–433. IEEE, 2014.

[94] Yin Tat Lee, Zhao Song, and Qiuyi Zhang. Solving empirical risk minimization in the current matrix multiplication time. In *COLT*. `https://arxiv.org/pdf/1905.04447`, 2019.

[95] Xingguo Li, Jarvis Haupt, and David Woodruff. Near optimal sketching of low-rank tensor regression. In *Advances in Neural Information Processing Systems*, pages 3466–3476. `https://arxiv.org/pdf/1709.07093`, 2017.

[96] László Lovász and Santosh Vempala. Hit-and-run from a corner. *SIAM J. Comput.*, 35(4):985–1005, 2006.

[97] Haihao Lu, Robert M Freund, and Yurii Nesterov. Relatively smooth convex optimization by first-order methods, and applications. *SIAM Journal on Optimization*, 28(1):333–354, 2018.

[98] Shuai Lu, Sergei V Pereverzev, and Ulrich Tautenhahn. Regularized total least squares: computational aspects and error bounds. *SIAM Journal on Matrix Analysis and Applications*, 31(3):918–941, 2009.

[99] Ivan Markovsky and Sabine Van Huffel. Overview of total least-squares methods. *Signal processing*, 87(10):2283–2302, 2007.

[100] Robert James McEliece. *Linear recurring sequences over finite fields*. PhD thesis, California Institute of Technology, 1967.

[101] Xiangrui Meng and Michael W Mahoney. Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 91–100. ACM, `https://arxiv.org/pdf/1210.3135`, 2013.

[102] Dana Moshkovitz and Michal Moshkovitz. Mixing implies lower bounds for space bounded learning. In *Proceedings of the 30th Conference on Learning Theory, COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017*, pages 1516–1566, 2017.

[103] Dana Moshkovitz and Michal Moshkovitz. Entropy samplers and strong generic lower bounds for space bounded learning. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, pages 28:1–28:20, 2018.

[104] David E Muller. Application of Boolean algebra to switching circuit design and to error detection. *Transactions of the IRE Professional Group on Electronic Computers*, EC-3(3):6–12, 1954.

[105] Jelani Nelson and Huy L Nguyên. OSNAP: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 117–126. IEEE, 2013.

[106] Arkadi Nemirovski. On self-concordant convex–concave functions. *Optimization Methods and Software*, 11(1-4):303–384, 1999.

[107] Yurii Nesterov and Arkadii Nemirovskii. *Interior-point polynomial algorithms in convex programming*, volume 13. Siam, 1994.

[108] Aleksandar Nikolov, Kunal Talwar, and Li Zhang. The geometry of differential privacy: the sparse and approximate cases. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 351–360. ACM, 2013.

[109] Halil Oruç and George M Phillips. Explicit factorization of the Vandermonde matrix. *Linear Algebra and its Applications*, 315(1-3):113–123, 2000.

[110] Eric Price, Zhao Song, and David P. Woodruff. Fast regression with an $\ell_\infty$ guarantee. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, 2017.

[111] Friedrich Pukelsheim. *Optimal design of experiments*. SIAM, 2006.

[112] P. van der Putten and M. van Someren. Coil challenge 2000: The insurance company case. Technical report.

[113] Ran Raz. Fast learning requires good memory: A time-space lower bound for parity learning. In *Proceedings, 57th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2016, New Brunswick, New Jersey, USA*, pages 266–275, October 2016.

[114] Ran Raz. A time-space lower bound for a large class of learning problems. In *Proceedings, 58th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, California, USA*, pages 732–742, October 2017.

[115] Ilya Razenshteyn, Zhao Song, and David P Woodruff. Weighted low rank approximations with provable guarantees. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 250–263. ACM, 2016.

[116] Irving S Reed. A class of multiple-error-correcting codes and the decoding scheme. Technical report, MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB, 1953.

[117] Rosemary A Renaut and Hongbin Guo. Efficient algorithms for solution of regularized total least squares. *SIAM Journal on Matrix Analysis and Applications*, 26(2):457–476, 2004.

[118] Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. Green ai. *arXiv preprint arXiv:1907.10597*, 2019.

[119] Ohad Shamir. Fundamental limits of online and distributed algorithms for statistical learning and estimation. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*, pages 163–171, Montreal, Quebec, Canada, 2014.

[120] Vatsal Sharan, Aaron Sidford, and Gregory Valiant. Memory-sample tradeoffs for linear regression with small error. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 890–901, 2019.

[121] Alexander A Sherstov. Halfspace matrices. *Computational Complexity*, 17(2):149–178, 2008.

[122] Mohit Singh and Weijun Xie. Approximate positive correlated distributions and approximation algorithms for D-optimal design. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2240–2255. Society for Industrial and Applied Mathematics, 2018.

[123] Neil J. A. Sloane and Elwyn R. Berlekamp. Weight enumerator for second-order Reed-Muller codes. *IEEE Trans. Information Theory*, 16(6):745–751, 1970.

[124] Marta Soare. *Sequential resource allocation in linear stochastic bandits*. PhD thesis, Université Lille 1-Sciences et Technologies, 2015.

[125] Marta Soare, Alessandro Lazaric, and Rémi Munos. Best-arm identification in linear bandits. In *Advances in Neural Information Processing Systems*, pages 828–836, 2014.

[126] Zhao Song, Ruosong Wang, Lin F. Yang, Hongyang Zhang, and Peilin Zhong. Efficient symmetric norm regression via linear sketching. In *NeurIPS*, 2019.

[127] Zhao Song, David P. Woodruff, and Huan Zhang. Sublinear time orthogonal tensor decomposition. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems (NIPS) 2016, December 5-10, 2016, Barcelona, Spain*, pages 793–801, 2016.

[128] Zhao Song, David P Woodruff, and Peilin Zhong. Low rank approximation with entrywise $\ell_1$-norm error. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 688–701. ACM, 2017.

[129] Zhao Song, David P Woodruff, and Peilin Zhong. Average case column subset selection for entrywise $\ell_1$-norm loss. In *NeurIPS*, 2019.

[130] Zhao Song, David P Woodruff, and Peilin Zhong. Relative error tensor low rank approximation. In *SODA*. `https://arxiv.org/pdf/1704.08246`, 2019.

[131] Zhao Song, David P Woodruff, and Peilin Zhong. Towards a zero-one law for column subset selection. In *NeurIPS*, 2019.

[132] Zhao Song, Lin F Yang, and Peilin Zhong. Sensitivity sampling over dynamic geometric data streams with applications to $k$-clustering. *arXiv preprint arXiv:1802.00459*, 2018.

[133] Daniel A Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.

[134] Jacob Steinhardt, Gregory Valiant, and Stefan Wager. Memory, communication, and statistical queries. In *Proceedings of the 29th Conference on Learning Theory, COLT 2016, New York, USA*, pages 1490–1516, 2016.

[135] Thomas Strohmer and Roman Vershynin. A randomized solver for linear systems with exponential convergence. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 499–507. Springer, 2006.

[136] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, 2019.

[137] Peng Sun and Robert M Freund. Computation of minimum-volume covering ellipsoids. *Operations Research*, 52(5):690–706, 2004.

[138] Chao Tao, Saúl Blanco, and Yuan Zhou. Best arm identification in linear bandits with linear dimension dependency. In *International Conference on Machine Learning*, pages 4877–4886, 2018.

[139] Mikkel Thorup and Yin Zhang. Tabulation-based 5-independent hashing with applications to linear probing and second moment estimation. *SIAM Journal on Computing*, 41(2):293–331, 2012.

[140] Andrea Tirinzoni, Matteo Pirotta, Marcello Restelli, and Alessandro Lazaric. An asymptotically optimal primal-dual incremental algorithm for contextual linear bandits. *Advances in Neural Information Processing Systems*, 33, 2020.

[141] Michael J. Todd. *Minimum-volume ellipsoids : theory and algorithms*. MOS-SIAM series on optimization. SIAM, Philadelphia, 2016.

[142] Michael J Todd and E Alper Yıldırım. On khachiyan's algorithm for the computation of minimum-volume enclosing ellipsoids. *Discrete Applied Mathematics*, 155(13):1731–1744, 2007.

[143] UCI. Airfoil self-noise. In . `https://archive.ics.uci.edu/ml/datasets/Airfoil+Self-Noise,`.

[144] UCI. Insurance company benchmark (coil 2000) data set. In . `https://archive.ics.uci.edu/ml/datasets/Insurance+Company+Benchmark+%28COIL+2000%29,`.

[145] UCI. Wine quality. In . `https://archive.ics.uci.edu/ml/datasets/Wine+Quality,`.

[146] Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[147] Sabine Van Huffel and Joos Vandewalle. *The total least squares problem: computational aspects and analysis*, volume 9. Siam, 1991.

[148] Santosh Vempala. Geometric random walks: a survey. In *Combinatorial and computational geometry*, volume 52 of *Math. Sci. Res. Inst. Publ.*, pages 577–616. Cambridge Univ. Press, Cambridge, 2005.

[149] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

[150] Yining Wang, Hsiao-Yu Tung, Alexander J Smola, and Anima Anandkumar. Fast and guaranteed tensor decomposition via sketching. In *Advances in Neural Information Processing Systems (NIPS)*, pages 991–999. `https://arxiv.org/pdf/1506.04448`, 2015.

[151] Yining Wang, Adams Wei Yu, and Aarti Singh. On computationally tractable selection of experiments in measurement-constrained regression models. *The Journal of Machine Learning Research*, 18(1):5238–5278, 2017.

[152] Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing (STOC)*, pages 887–898. ACM, 2012.

[153] David P Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157, 2014.

[154] Liyuan Xu, Junya Honda, and Masashi Sugiyama. A fully adaptive algorithm for pure exploration in linear bandits. In *International Conference on Artificial Intelligence and Statistics*, pages 843–851. PMLR, 2018.

[155] Yin Zhang and Liyan Gao. On numerical solution of the maximum volume ellipsoid problem. *SIAM Journal on Optimization*, 14(1):53–76, 2003.

# Appendix A

## A FAST SOLVER FOR THE TOTAL LEAST SQUARES PROBLEM

In this chapter, we provide the matlab code for experiments in Chapter 6.

```
function [hatC,cost]=optimalSelect(A,B,s1,d2,o1,o2)

C=[A,B];
szA=size(A);
n=szA(2);
% step 1: compute the sketch version


if o1==1
    SC=countSketch(C,s1);
elseif o1==2
    SC=slowLeverageScoreSampling(C,s1);
else
    SC=GaussianSampling(C,s1)*C;
end

% step 2: compute leverage score sampling matrix
%D_1 according to columns of S_1 C


CD=C;

% step 3: compute the leverage score sampling matrix
%D_2 according to rows of C D_1
```

```
if o2==1
    DC=countSketch(C,d2);
elseif o2==2
    DC=slowLeverageScoreSampling(C,d2);
else
    DC=GaussianSampling(C,d2)*C;
end

% step 4: solve the small size problem
Z=rankOptimize(DC,DC,SC,n);

hatC=CD*Z*SC;

V=hatC-C;
cost=sum(sum(V.^2));


% compare to optimal solution
% hatX=rankOptimize(B,A,eye(d),d);
end
```

# Appendix B

## A FAST ALGORITHM FOR COMPUTING THE JOHN ELLIPSOID

In this chapter, we provide the matlab code for experiments in Chapter 7.

```
% Compute the John ellipsoid weight for a full rank matrix A
function [w_avg, iter] = FixedPoint(A,tol,w)
m = size(A,1); n = size(A,2);
exactTime = Inf; useJL = false;
if ~exist('tol', 'var'), tol = 0.01; end
if ~exist('w', 'var'), w = ones(m,1)*n/m; end
if issparse(A), A = A(:,colamd(A)); useJL = true; end

w_avg = w;
for iter = 1:ceil(10*log(m/n)/tol)
    tau = computeTau(w, 10+iter); %increase the JLdim

    if max(tau./w) < 1 + tol
        w_avg = w; %use only the current w if we detect it has convgerged
        break;
    end
    w = tau; %update w

    if useJL
        w_avg = (1-1/iter)*w_avg + w/iter;
        if randi(iter) == 1 %with prob. 1/iter, test if we have converged
                        %check conv of running avg
            tau_avg = computeTau(w_avg, 10+iter^2);
            if max(tau_avg./w_avg) < 1 + tol, break; end
        end
```

```
    end
end

% compute tau = diag(B*inv(B'*B)*B') with B = sqrt(W) A
function tau = computeTau(w, JLdim)
    tstart = tic; %record times to decide when to use sketching
    B = spdiags(sqrt(w),0,m,m)*A;
    R = chol(B' * B); % In rare cases, chol fails due to numerical error

    if (nnz(R) > n^2/20)
                A = full(A);
                useJL = false;
        end % use full if dense

    if (exactTime == Inf || ~useJL) % use JL only if it is much faster
        S = R' \ B';
        tau = full(sum(S.^2,1)'); % compute tau exactly
        exactTime = toc(tstart);
    else
        S = B * (R\randn(n, JLdim));
        tau = full(sum(S.^2,2)/JLdim);  % compute tau using JL
        JLTime = toc(tstart);
        if JLTime*(10+iter) > exactTime, useJL = false; end
    end
end
end
```